MULTI-ENTITY BAYESIAN NETWORKS LEARNING FOR PREDICTIVE SITUATION AWARENESS

by

Cheol Young Park
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

_________________________________ Dr. Kathryn B Laskey, Dissertation Director

_________________________________ Dr. Huzefa Rangwala, Committee Member

_________________________________ Dr. Kuo-Chu Chang, Committee Member

_________________________________ Dr. Paulo Costa, Committee Member

_________________________________ Dr. Stephen Nash, Senior Associate Dean

_________________________________ Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering

Date: _____________________________ Fall Semester 2017
George Mason University
Fairfax, VA
Multi-Entity Bayesian Networks Learning for Predictive Situation Awareness

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

by

Cheol Young Park
Master of Science
George Mason University, 2010
Bachelor of Science
Won Kwang Digital University, 2005

Director: Kathryn B Laskey, Professor
Department of Information Sciences and Technology

Fall Semester 2017
George Mason University
Fairfax, VA
Copyright 2017 Cheol Young Park
All Rights Reserved
DEDICATION

This dissertation is dedicated to my father, Rho Hyun Park, for his devotion and my mother, Yong Ju Kwon, in heaven.
ACKNOWLEDGEMENTS

In my PhD program, I had the fortune of having an excellent supervisor, Dr. Kathryn B Laskey. She always encouraged me to meet the challenge. When I achieved a small accomplishment, she continuously presented the next level of challenge. I saw that she was doing the best for what people easily overlook and learned many things from her attitude. I am grateful that my attitude toward the world has grown. I would like to express my deepest gratitude to my dissertation committee, Dr. Huzefa Rangwala, Dr. Kuo-Chu Chang, and Dr. Paulo Costa. They have broadened my scientific thinking and research skills. For my PhD program, many people have helped me directly or indirectly. I would like to thank Shou Matsumoto, Andy Powell, Mark Locher, Rommel Carvalho, Ricardo Carvalho, Young Don Shin, Wei Sun, Aditya Mugali, Seungwon Noh, Charles R Twardy, Todd Martin, James Ha, Won Jae Lee, Hyung Joon Nam, Timothy Park, Thomas Kang, Eugene Ro, and Seung Joon Nam. I am grateful that my dear children, Eui Chan Park and Arie Park, have been together for these times. My thanks are extended to my father-in-law, Hyo Ki Kim, and mother-in-law, Eun Sook Han, for their love to my family. Finally, I would like to thank my wife Ju Mi Kim for all the love.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>xi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xii</td>
</tr>
<tr>
<td>List of Abbreviations and Symbols</td>
<td>xiv</td>
</tr>
<tr>
<td>Abstract</td>
<td>xviii</td>
</tr>
<tr>
<td>1. Chapter One: Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Suitability of MEBN for PSAW</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Problem Statement</td>
<td>10</td>
</tr>
<tr>
<td>1.3 Thesis Statement and Scope</td>
<td>17</td>
</tr>
<tr>
<td>1.4 Contributions</td>
<td>18</td>
</tr>
<tr>
<td>1.5 Organization of the Dissertation</td>
<td>19</td>
</tr>
<tr>
<td>2. Chapter Two: Background</td>
<td>20</td>
</tr>
<tr>
<td>2.1 From Data Fusion to Situation Awareness</td>
<td>20</td>
</tr>
<tr>
<td>2.1.1 Data Fusion</td>
<td>20</td>
</tr>
<tr>
<td>2.1.2 Situation Awareness</td>
<td>23</td>
</tr>
<tr>
<td>2.2 Bayesian Network</td>
<td>26</td>
</tr>
<tr>
<td>2.3 Multi-Entity Bayesian Network</td>
<td>27</td>
</tr>
<tr>
<td>2.3.1 Definition of MEBN</td>
<td>27</td>
</tr>
<tr>
<td>2.3.2 A Script for MEBN</td>
<td>35</td>
</tr>
<tr>
<td>2.4 Data for MEBN Learning from RM</td>
<td>36</td>
</tr>
<tr>
<td>2.4.1 Relational Model</td>
<td>37</td>
</tr>
<tr>
<td>2.5 Uncertainty Modeling Process for Semantic Technology (UMP-ST)</td>
<td>43</td>
</tr>
<tr>
<td>2.6 Conclusion</td>
<td>45</td>
</tr>
<tr>
<td>3. Chapter Three: MEBN-RM Mapping Model</td>
<td>46</td>
</tr>
<tr>
<td>3.1 MEBN-RM</td>
<td>47</td>
</tr>
<tr>
<td>3.2 Entity Mapping</td>
<td>52</td>
</tr>
<tr>
<td>3.3 Resident Node Mapping</td>
<td>52</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.3.1 Predicate</td>
<td>53</td>
</tr>
<tr>
<td>3.3.2 Function</td>
<td>55</td>
</tr>
<tr>
<td>3.4 Relation Schema and MFragment Mapping</td>
<td>56</td>
</tr>
<tr>
<td>3.5 Relational Database Schema and MTheory Mapping</td>
<td>58</td>
</tr>
<tr>
<td>3.6 MEBN-RM Mapping Algorithm</td>
<td>59</td>
</tr>
<tr>
<td>3.7 Conclusion</td>
<td>62</td>
</tr>
<tr>
<td>4. Chapter Four: PSAW-MEBN Reference Model</td>
<td>64</td>
</tr>
<tr>
<td>4.1 Predictive Situation Awareness</td>
<td>66</td>
</tr>
<tr>
<td>4.1.1 Definition of PSAW</td>
<td>66</td>
</tr>
<tr>
<td>4.1.2 Properties of a PSAW Model</td>
<td>71</td>
</tr>
<tr>
<td>4.1.3 Physical and Mental Situations in PSAW</td>
<td>76</td>
</tr>
<tr>
<td>4.1.4 Properties of OODA</td>
<td>78</td>
</tr>
<tr>
<td>4.1.5 PSAW in terms of OODA</td>
<td>82</td>
</tr>
<tr>
<td>4.2 PSAW-MEBN Reference Model</td>
<td>85</td>
</tr>
<tr>
<td>4.2.1 Entities for the Reference Model</td>
<td>89</td>
</tr>
<tr>
<td>4.2.2 Random Variables for the Reference Model</td>
<td>90</td>
</tr>
<tr>
<td>4.2.3 The Five MFragment Groups</td>
<td>95</td>
</tr>
<tr>
<td>4.3 Conclusion</td>
<td>99</td>
</tr>
<tr>
<td>5. Chapter Five: Human-aided MEBN Learning for PSAW</td>
<td>100</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>101</td>
</tr>
<tr>
<td>5.1.1 Illustrative Running Example of Relational Data for MEBN Learning</td>
<td>101</td>
</tr>
<tr>
<td>5.1.2 Elements of MEBN Learning from Relational Data</td>
<td>103</td>
</tr>
<tr>
<td>5.2 A Process for Human-Aided MEBN Learning for PSAW</td>
<td>104</td>
</tr>
<tr>
<td>5.2.1 Analyze Requirements</td>
<td>105</td>
</tr>
<tr>
<td>5.2.2 Define World Model</td>
<td>108</td>
</tr>
<tr>
<td>5.2.3 Construct Reasoning Model</td>
<td>116</td>
</tr>
<tr>
<td>5.2.4 Test Reasoning Model</td>
<td>144</td>
</tr>
<tr>
<td>5.3 Conclusion</td>
<td>145</td>
</tr>
<tr>
<td>6. Chapter SIX: Applying HMLP - Use Cases and Experiment</td>
<td>151</td>
</tr>
<tr>
<td>6.1 Use Case 1 - HERALD</td>
<td>151</td>
</tr>
<tr>
<td>6.1.1 Critical Infrastructure Defense Situation</td>
<td>152</td>
</tr>
<tr>
<td>6.1.2 HERALD System</td>
<td>155</td>
</tr>
</tbody>
</table>
6.1.3 Discussion of HMLP Performance ................................................................. 158
6.2 Use Case 2 - PROGNOS ....................................................................................... 161
   6.2.1 Introduction .................................................................................................... 162
   6.2.2 PROGNOS PO via HMLP ............................................................................. 163
   6.2.3 Comparing UMP-ST and HMLP ................................................................. 170
6.3 Use Case 3 - Smart Manufacturing ....................................................................... 172
   6.3.1 Introduction .................................................................................................... 173
   6.3.2 MSAW-MEBN Model for Predictive Situation Awareness ......................... 175
   6.3.3 Discussion ....................................................................................................... 179
6.4 Experimental Comparison between UMP-ST and HMLP .................................... 179
   6.4.1 Preparation Process ......................................................................................... 181
   6.4.2 Execution Process ........................................................................................... 182
   6.4.3 Evaluation Process .......................................................................................... 186
   6.4.4 Comparison Results ........................................................................................ 188
7. Chapter SEVEN: Conclusion and Future Work ..................................................... 193
   7.1 MEBN-RM Mapping Model .............................................................................. 193
   7.2 The PSAW-MEBN Reference Model ............................................................... 194
   7.3 Human-aided MEBN Learning for PSAW ....................................................... 194
   7.4 Future Research Questions ................................................................................ 195
      7.4.1 Aggregating Influence Problem .................................................................. 195
      7.4.2 Dynamic Model Learning .......................................................................... 196
      7.4.3 Continuous Random Variable Learning ...................................................... 196
      7.4.4 Learning from Incomplete Data ................................................................... 197
      7.4.5 Learning for Hidden Variables .................................................................... 198
      7.4.6 Incremental MEBN Learning ..................................................................... 199
   7.6 Conclusion ......................................................................................................... 199
Appendix .................................................................................................................. 201
   Appendix A. Bayesian Network Learning .............................................................. 203
      A.1 BN Parameter Learning ................................................................................. 203
   Appendix B. PSAW Questions .............................................................................. 211
   Appendix C. Metrics for Prediction Accuracy ......................................................... 213
   Appendix D. Local Probability Description Language (LPDL) .............................. 215
Appendix E. Conditional Probability Script Language (CPSL) ........................................ 217
Appendix F. An HMLP Tool .......................................................................................... 224
Appendix G. Use Case 1: Human-Aided MEBN Learning for HERALD and Test Results ........................................ 226
  G.1 Analyze Requirements .......................................................................................... 226
  G.2 Define World Model .......................................................................................... 228
  G.2 Construct Reasoning Model .............................................................................. 238
  G.4 Test Reasoning Model .......................................................................................... 241
Appendix H. Use Case 1: HERALD Scenario Simulator ............................................. 256
  H.1 Environmental Factors, Spatial objects, Way Points, and Time ...................... 256
  H.2 Simulation Entities ............................................................................................ 258
  H.3 Sensor Systems in the Simulation ........................................................................ 260
  H.4 Ground Truth Dataset ...................................................................................... 263
Appendix I. Use Case 1: HERALD MTheory ................................................................. 266
  I.1 SensorOf MFrag ................................................................................................. 268
  I.2 Predecessor MFrag ............................................................................................. 269
  I.3 SensorTemporalProperty MFrag ........................................................................ 269
  I.4 ReportedTarget_MTIRPT MFrag ........................................................................ 269
  I.5 ReportedTarget_IMINTSRPT MFrag ................................................................ 270
  I.6 ReportedTarget_GEOINTRPT MFrag ............................................................... 270
  I.7 GEOINTS_Report MFrag .................................................................................. 270
  I.8 MTI_Report MFrag ............................................................................................ 271
  I.9 IMINTS_Report MFrag ..................................................................................... 271
  I.10 Target MFrag .................................................................................................... 272
  I.11 TargetTemporalProperty MFrag ...................................................................... 272
  I.12 Situation MFrag ................................................................................................. 273
Appendix J. Use Case 1: Confusion Matrix for Categorical Variables of HERALD MTheory .............................................................................................. 274
  J.1 SSBN_g1_t2 ....................................................................................................... 274
  J.2 SSBN_g1_t3 ....................................................................................................... 275
  J.3 SSBN_g1_t4 ....................................................................................................... 275
Appendix K. Use Case 1: SSBN_g1_t4 ........................................................................ 277
Appendix L. Use Case 2: PROGNOS PO via UMP-ST .................................................. 301
# Table of Contents

L.1 Requirements ..................................................................................................... 301  
L.2 Analysis & Design ............................................................................................ 302  
L.3 Implementation ................................................................................................. 304  
L.4 Test .................................................................................................................... 308  

**Appendix M. Use Case 3: Predictive Situation Awareness Model for Smart Manufacturing** ......................................................... 309  
M.1 Smart Manufacturing ....................................................................................... 309  
M.2 Representation for Uncertainty in Manufacturing ........................................... 312  
M.3 Use Case ........................................................................................................... 319  
M.4 Conclusion ....................................................................................................... 325  

**Appendix N. Experimental Comparison of HMLP with UMP-ST** .......................... 326  
N.1 Slides for the Lectures ...................................................................................... 326  
N.2 Data given to Participants ................................................................................. 334  
N.3 MEBN Model Results from Participants .......................................................... 335  

**Appendix O. Hybrid Message Passing with Gaussian Mixture Reduction with Optimal Settings** ................................................................................. 341  
O.1 Introduction ........................................................................................................ 342  
O.2 Preliminaries ..................................................................................................... 348  
  O.2.1 Structure of Hybrid Bayesian Network ......................................................... 348  
  O.2.2 Message Passing Inference for Discrete BN ............................................... 351  
  O.2.3 Message Passing Inference for Continuous BNs ....................................... 354  
  O.2.4 Message Passing Inference for Hybrid BNs ............................................... 355  
  O.2.5 Gaussian Mixture Reduction ....................................................................... 357  
O.3 Extended Hybrid Message Passing Algorithm ..................................................... 358  
O.4 Optimizing the Settings of HMP-GMR .............................................................. 363  
O.5 Experiment ........................................................................................................ 368  
  O.5.1 Scalability of HMP-GMR ............................................................................. 372  
  O.5.2 Accuracy and Efficiency of HMP-GMR ...................................................... 374  
  O.5.3 Optimal Settings for HMP-GMR ................................................................. 378  
O.6 Conclusion ........................................................................................................... 386  

**Appendix P. Test HBN for HMP-GMR** ................................................................... 387  
CLG BN 1 (n = 5) ................................................................................................... 387  
CLG BN 2 (n = 5) ................................................................................................... 387
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3.1 Resident node types on MEBN-RM</td>
<td>53</td>
</tr>
<tr>
<td>Table 3.2 Predicate mapping in MEBN-RM</td>
<td>54</td>
</tr>
<tr>
<td>Table 3.3 Function mapping in MEBN-RM</td>
<td>56</td>
</tr>
<tr>
<td>Table 3.4 Mapping types on MEBN-RM</td>
<td>62</td>
</tr>
<tr>
<td>Table 4.1 Possible inputs/outputs for the four steps in OODA</td>
<td>81</td>
</tr>
<tr>
<td>Table 5.1 Parts of the Situation Identification Relational Database</td>
<td>103</td>
</tr>
<tr>
<td>Table 5.2 Normalized relational dataset from Table 5.1</td>
<td>118</td>
</tr>
<tr>
<td>Table 5.3 Joined dataset</td>
<td>123</td>
</tr>
<tr>
<td>Table 5.4 CSD dataset</td>
<td>134</td>
</tr>
<tr>
<td>Table 5.5 Communicate Relation and Meet Relation</td>
<td>141</td>
</tr>
<tr>
<td>Table 5.6 Converted Relations from Communicate Relation and Meet Relation</td>
<td>142</td>
</tr>
<tr>
<td>Table 5.7 Processing Method for Steps in HMLP</td>
<td>149</td>
</tr>
<tr>
<td>Table 6.1 Comparison between UMP-ST and HMLP in terms of the development periods</td>
<td>161</td>
</tr>
<tr>
<td>Table 6.2 Comparison between the original PROGNOS probabilistic ontology</td>
<td>171</td>
</tr>
<tr>
<td>Table 6.3 Preparation Process</td>
<td>182</td>
</tr>
<tr>
<td>Table 6.4 Execution Process</td>
<td>186</td>
</tr>
<tr>
<td>Table 6.5 Evaluation Process</td>
<td>188</td>
</tr>
<tr>
<td>Table 6.6 Comparison results for total development times and average accuracies</td>
<td>189</td>
</tr>
<tr>
<td>Table 6.7 Comparison results for time-consuming tasks</td>
<td>190</td>
</tr>
<tr>
<td>Table 7.1 Dataset A represents the BN data case. Dataset B represents the MEBN data case</td>
<td>198</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Danger Assessment Problem</td>
<td>3</td>
</tr>
<tr>
<td>Figure 1.2</td>
<td>Bayesian Network of the Danger Assessment Problem</td>
<td>4</td>
</tr>
<tr>
<td>Figure 1.3</td>
<td>Danger Assessment MTheory</td>
<td>7</td>
</tr>
<tr>
<td>Figure 1.4</td>
<td>An example SSBN with instance local distributions derived from the Vehicle Object MFragment given entity information {rgn = {Region1}, obj = {Vehicle1}}</td>
<td>8</td>
</tr>
<tr>
<td>Figure 1.5</td>
<td>Generating Several SSBNs</td>
<td>9</td>
</tr>
<tr>
<td>Figure 1.6</td>
<td>Concept of Research</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Danger MFragment</td>
<td>28</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>SSBN from Danger MFragment (given v1, v2, and v3 as vehicle, and region1_1 as region)</td>
<td>30</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Hybrid SSBN from Danger MFragment</td>
<td>32</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>Example of a Vehicle Identification RDB</td>
<td>38</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>An illustrative example of situation awareness</td>
<td>67</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>OODA Loop</td>
<td>79</td>
</tr>
<tr>
<td>Figure 2.7</td>
<td>What an observer sees</td>
<td>81</td>
</tr>
<tr>
<td>Figure 2.8</td>
<td>A situation of an observer in terms of Interpreted/Actual OODA</td>
<td>83</td>
</tr>
<tr>
<td>Figure 2.9</td>
<td>Illustrative example of a PSAW model</td>
<td>85</td>
</tr>
<tr>
<td>Figure 2.10</td>
<td>An example of an SSBN generated by a PSAW-MTheory</td>
<td>87</td>
</tr>
<tr>
<td>Figure 2.11</td>
<td>Four core kinds of RVs and examples of each</td>
<td>94</td>
</tr>
<tr>
<td>Figure 2.12</td>
<td>Illustrative example of PSAW-MFrags</td>
<td>96</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Schema of a Situation Identification Relational Database</td>
<td>101</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Learning Subjects for MEBN Learning</td>
<td>104</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>Process for Human-Aided MEBN Learning</td>
<td>105</td>
</tr>
<tr>
<td>Figure 5.4</td>
<td>Analyze Requirements</td>
<td>106</td>
</tr>
<tr>
<td>Figure 5.5</td>
<td>Define World Model</td>
<td>110</td>
</tr>
<tr>
<td>Figure 5.6</td>
<td>Construct Reasoning Model</td>
<td>116</td>
</tr>
<tr>
<td>Figure 5.7</td>
<td>Test Reasoning Model</td>
<td>144</td>
</tr>
<tr>
<td>Figure 6.1</td>
<td>Ranges of a target from a critical infrastructure</td>
<td>154</td>
</tr>
<tr>
<td>Figure 6.2</td>
<td>HERALD System in a C4I system</td>
<td>157</td>
</tr>
<tr>
<td>Figure 6.3</td>
<td>Comparison between UMP-ST and HMLP</td>
<td>159</td>
</tr>
<tr>
<td>Figure 6.4</td>
<td>Part of EER Model for a PROGNOS world model</td>
<td>166</td>
</tr>
<tr>
<td>Figure 6.5</td>
<td>Extended PROGNOS probabilistic ontology</td>
<td>168</td>
</tr>
<tr>
<td>Figure 6.6</td>
<td>Extended MSAW-MEBN Model derived using the PSAW-MEBN reference model (Chapter 4)</td>
<td>176</td>
</tr>
<tr>
<td>Figure 6.7</td>
<td>Situation for the simple heating machinery</td>
<td>183</td>
</tr>
</tbody>
</table>
Figure 6.8 Each of training and test data has sensed data and actual data for the simple heating machinery.................................................................................................................. 185
Figure 6.9 Ideal causal relationships between random variables for the simple heating machinery.................................................................................................................. 190
# LIST OF ABBREVIATIONS AND SYMBOLS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activities</td>
<td>Act</td>
</tr>
<tr>
<td>Actors</td>
<td>Atr</td>
</tr>
<tr>
<td>Aggregating Function</td>
<td>AG Function</td>
</tr>
<tr>
<td>Air-Moving Object</td>
<td>AMO</td>
</tr>
<tr>
<td>Area under the Receiver Operating Characteristic Curve</td>
<td>AUC</td>
</tr>
<tr>
<td>Bayesian Network</td>
<td>BN</td>
</tr>
<tr>
<td>Central Processing Unit</td>
<td>CPU</td>
</tr>
<tr>
<td>Child nodes of the node X</td>
<td>Ch(X)</td>
</tr>
<tr>
<td>Class Local Distribution</td>
<td>CLDs</td>
</tr>
<tr>
<td>Class Local Distributions</td>
<td>CLDs</td>
</tr>
<tr>
<td>Class-parenthood condition</td>
<td>CPC</td>
</tr>
<tr>
<td>Class-Sub-Local Distribution</td>
<td>CSD</td>
</tr>
<tr>
<td>Closed-World Assumption</td>
<td>CWA</td>
</tr>
<tr>
<td>Command and control</td>
<td>C2</td>
</tr>
<tr>
<td>Command, Control, Communications, Computers, and Intelligence</td>
<td>C4I</td>
</tr>
<tr>
<td>Communications Intelligence</td>
<td>COMINT</td>
</tr>
<tr>
<td>Conditional Gaussian</td>
<td>CG</td>
</tr>
<tr>
<td>Conditional Linear Gaussian</td>
<td>CLG</td>
</tr>
<tr>
<td>Conditional Nonlinear Gaussian</td>
<td>CNG</td>
</tr>
<tr>
<td>Conditional Probability Script Language</td>
<td>CPSL</td>
</tr>
<tr>
<td>Conditional Probability Table</td>
<td>CPT</td>
</tr>
<tr>
<td>Context node</td>
<td>C</td>
</tr>
<tr>
<td>Continuous Ranked Probability Score</td>
<td>CRPS</td>
</tr>
<tr>
<td>Course Of Actions</td>
<td>COA</td>
</tr>
<tr>
<td>Critical Infrastructure</td>
<td>CI</td>
</tr>
<tr>
<td>Data Fusion</td>
<td>DF</td>
</tr>
<tr>
<td>Detection-Classification-Recognition-Identification</td>
<td>DCRI</td>
</tr>
<tr>
<td>Direct Acyclic Graph</td>
<td>DAG</td>
</tr>
<tr>
<td>Domain of a variable X</td>
<td>Dom(X)</td>
</tr>
<tr>
<td>Domain value, $x_{pq}$, where p is an index of X and q is an index of x</td>
<td>$x$</td>
</tr>
<tr>
<td>Dynamic Bayesian networks</td>
<td>DBNs</td>
</tr>
<tr>
<td>Enhanced Entity–Relationship</td>
<td>EER</td>
</tr>
<tr>
<td>Entity Relation Schema</td>
<td>ERS</td>
</tr>
<tr>
<td>Entity</td>
<td>E</td>
</tr>
<tr>
<td>Entity–Relationship</td>
<td>ER</td>
</tr>
<tr>
<td>First-Order Logic</td>
<td>FOL</td>
</tr>
<tr>
<td>Term</td>
<td>Abbreviation</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Foreign Key</td>
<td>FK</td>
</tr>
<tr>
<td>Fragment graph in an MFrag</td>
<td>F</td>
</tr>
<tr>
<td>Gaussian Mixture Model</td>
<td>GMM</td>
</tr>
<tr>
<td>Gaussian Mixture Reduction</td>
<td>GMR</td>
</tr>
<tr>
<td>Geospatial Intelligence</td>
<td>GEOINT</td>
</tr>
<tr>
<td>Headquarters Effectiveness Assessment Tool</td>
<td>HEAT</td>
</tr>
<tr>
<td>Higher level information fusion</td>
<td>HLIF</td>
</tr>
<tr>
<td>HMP-GMR with Optimal Setting</td>
<td>HMP-GMR-OS</td>
</tr>
<tr>
<td>Human-aided MEBN learning for PSAW</td>
<td>HMLP</td>
</tr>
<tr>
<td>Hybrid Bayesian Network</td>
<td>HBN</td>
</tr>
<tr>
<td>Hybrid Message Passing</td>
<td>HMP</td>
</tr>
<tr>
<td>If the statement S is true, then return 1, else return 0</td>
<td>$\mathbb{I}[S]$</td>
</tr>
<tr>
<td>Imagery Intelligence</td>
<td>IMINT</td>
</tr>
<tr>
<td>Independent and identically distributed</td>
<td>IID</td>
</tr>
<tr>
<td>Input Node</td>
<td>I</td>
</tr>
<tr>
<td>Input Node</td>
<td>IP</td>
</tr>
<tr>
<td>Instance Local Distribution</td>
<td>ILD</td>
</tr>
<tr>
<td>Instance parent condition</td>
<td>IPC</td>
</tr>
<tr>
<td>Instance-Sub-Local Distribution</td>
<td>ISD</td>
</tr>
<tr>
<td>Interpreted Situation</td>
<td>IS</td>
</tr>
<tr>
<td>Joint Directors of Laboratories</td>
<td>JDL</td>
</tr>
<tr>
<td>Junction Tree</td>
<td>JT</td>
</tr>
<tr>
<td>Kullback-Leibler</td>
<td>KL</td>
</tr>
<tr>
<td>Land-Moving Object</td>
<td>LMO</td>
</tr>
<tr>
<td>Likelihood Weighting</td>
<td>LW</td>
</tr>
<tr>
<td>Local Probability Description Language</td>
<td>LPDL</td>
</tr>
<tr>
<td>Local Probability Distribution</td>
<td>LPD</td>
</tr>
<tr>
<td>Locations</td>
<td>Loc</td>
</tr>
<tr>
<td>Lower level information fusion</td>
<td>LLIF</td>
</tr>
<tr>
<td>Maritime Domain Awareness</td>
<td>MDA</td>
</tr>
<tr>
<td>Maximum Likelihood Estimation</td>
<td>MLE</td>
</tr>
<tr>
<td>Meaning of the situation</td>
<td>MS</td>
</tr>
<tr>
<td>Measures of Effectiveness</td>
<td>MOEs</td>
</tr>
<tr>
<td>Measures of Performance</td>
<td>MOPs</td>
</tr>
<tr>
<td>MEBN Fragment</td>
<td>MFragment</td>
</tr>
<tr>
<td>MEBN Theory</td>
<td>MTheory</td>
</tr>
<tr>
<td>MEBN-RM Mapping Model</td>
<td>MEBN-RM</td>
</tr>
<tr>
<td>Message Passing Algorithm with Gaussian Mixture Reduction</td>
<td>HMP-GMR</td>
</tr>
<tr>
<td>MFrag for PSAW</td>
<td>PSAW-MFragment</td>
</tr>
<tr>
<td>MFragment</td>
<td>F</td>
</tr>
<tr>
<td>Miles per Hour</td>
<td>MPH</td>
</tr>
<tr>
<td>MNode</td>
<td>N</td>
</tr>
<tr>
<td>Most probable a posteriori</td>
<td>MAP</td>
</tr>
<tr>
<td>Moving Target Indicator</td>
<td>MTI</td>
</tr>
</tbody>
</table>
MTheory for PSAW ................................................................. PSAW-MTheory
MTheory .................................................................................................................. M
Multi-Entity Bayesian Networks ............................................................... MEBN
Neighbor nodes of the node X ............................................................ Ne(X)
Non-Foreign-Key Attribute ........................................................................... A
Non-Primary Foreign Key ............................................................................. NK
Objects ........................................................................................................ Obj
Observe – Orient – Decide – Act ............................................................. OODA
Observer ......................................................................................................... OR
Observing Condition RV ........................................................................ OC_RV
Observing Condition .................................................................................. OC
Open-World Assumption .............................................................................. OWA
Parameter of a LR ....................................................................................... θR
Parent Configuration .................................................................................... PC
Parent nodes of the node X .......................................................................... Pa(X)
Predictive Situation Awareness ................................................................. PSAW
Primary Key ................................................................................................ PK
PRObabilistic OntoloGies for Net-Centric Operational Systems ............. PROGNOS
Probabilistic Ontology ..................................................................................... PO
Probabilistic OWL version 2 ................................................................. PR-OWL2
Probabilistic OWL ..................................................................................... PR-OWL
Probability ....................................................................................................... P
Random variable ............................................................................................ RV
Region Connection Calculus........................................................................... RCC
Relation Instance .......................................................................................... RI
Relation Schema ............................................................................................. RS
Relational Database Schema ......................................................................... RDBS
Relational Database ........................................................................................ RDB
Relational Model ............................................................................................ RM
Relationship Relation Schema ....................................................................... RRS
Reported Object RV ..................................................................................... RT_RV
Reported Target .............................................................................................. RT
Resident node ..................................................................................................... R
Resident Parent Node ....................................................................................... RP
RV for PSAW ................................................................................................. PSAW-RV
Sea-Moving Object ......................................................................................... SMO
Sensor .............................................................................................................. SR
Situation Assessment ....................................................................................... SA
Situation Awareness ........................................................................................ SAW
Situation RV ................................................................................................. SIT_RV
Situation Specific Bayesian Network ......................................................... SSBN
SSBN or BN ..................................................................................................... B
Stimulus – Hypothesis – Option – Response ..................................................... SHOR
Subject Matter Expert ...................................................................................... SME
Target Object RV .................................................................................................................. TR_RV
Target Situation.................................................................................................................. TS
Target.................................................................................................................................. TR
Technical Performance Measures .................................................................................. TPMs
Time .................................................................................................................................... T
Uncertainty Modeling Process for Semantic Technology ........................................ UMP-ST
Unified Modeling Language .......................................................................................... UML
Universal Time Coordinated ......................................................................................... UTC
Unmanned Aerial Vehicle............................................................................................... UAV
ABSTRACT

MULTI-ENTITY BAYESIAN NETWORKS LEARNING FOR PREDICTIVE SITUATION AWARENESS

Cheol Young Park, Ph.D.

George Mason University, 2017

Dissertation Director: Dr. Kathryn B Laskey

Over the past few decades, machine learning has led to substantial changes in Data Fusion Systems throughout the world. One of the most important application areas for data fusion is to support situation awareness for command and control. Situation Awareness (SAW) is perception of elements in the environment, comprehension of the current situation, and projection of future status. Predictive Situation Awareness (PSAW) emphasizes the ability to make predictions about aspects of a temporally evolving situation. PSAW requires a semantically rich representation to handle a complex real world situation and ability to reason under uncertainty about the situation. Probabilistic ontologies are able to address the requirements of PSAW, by augmenting standard ontologies with support for uncertainty management. PR-OWL (Probabilistic Web Ontology Language), a representation language for probabilistic ontologies, is founded on Multi-Entity Bayesian Networks (MEBN). MEBN combines First-Order Logic with
Bayesian Networks for representing and reasoning about uncertainty in complex, knowledge-rich domains. MEBN goes beyond standard Bayesian networks to enable reasoning about an unknown number of entities interacting with each other in various types of relationships, a key requirement for PSAW. MEBN models have heretofore been constructed manually by a domain expert. However, manual MEBN modeling is labor-intensive and insufficiently agile. To address these problems, an efficient method is needed for MEBN modeling. One of the methods is to use machine learning to learn a probabilistic ontology in whole or in part from data. In the era of Big Data, data-rich environments, characterized by uncertainty and complexity, have become ubiquitous. The larger the data sample is, the more accurate the results of the machine learning approach can be. Therefore, machine learning has potential to improve the quality of MEBN models. In this research, we study a machine learning method from data for MEBN-based probabilistic ontologies for PSAW. Specifically, we introduce a MEBN learning framework to develop a MEBN model from a combination of domain expert's knowledge and data. To support the framework, we present a bridge model between a MEBN model and a relational model, a reference model supporting the design of a MEBN model for PSAW, and a parameter learning algorithm given a MEBN model. The presented methodology is evaluated on three example use cases: (1) a Critical Infrastructure Defense system, (2) a Maritime Domain Awareness system, (3) a Smart Manufacturing system. Finally, we conduct an experiment to compare the MEBN learning framework and the manual MEBN modeling in terms of development efficiency.
CHAPTER ONE: INTRODUCTION

Over the past few decades, machine learning has led to substantial changes in Data Fusion Systems throughout the world [White, 1988; Endsley, 1988; Steinberg et al., 1998; Endsley et al., 2003; Llinas et al., 2004; Linggins et al., 2008]. One of the most important application areas for data fusion is to support Situation Awareness (SAW) to support command and control (C2). Systems to support SAW provide information regarding the present or future situation. This information supports situation assessment (SA) and is exploited for C2 decision making.

According to the most commonly cited definition, SAW is composed of three processes: perception of elements in the environment, comprehension of the current situation, and projection of the future status [Endsley, 1988; Endsley et al., 2003]. Breton and Rousseau classified 26 SAW definitions and identified a set of common elements of SAW [Breton & Rousseau, 2001]. They identified two distinct varieties, which they termed State- and Process-oriented SAW. In their definition, Process-oriented SAW focuses on the link between the situation and the cognitive processes generating SAW, while State-oriented SAW focuses on the link between the situation and an internal representation of elements present in the situation.

Predictive Situation Awareness (PSAW) places special emphasis on the ability to make predictions about aspects of a temporally evolving situation [Costa et al., 2009;
Carvalho et al., 2010]. Traditionally, decision makers are responsible for the higher level information fusion (HLIF) in which they use the results of low-level fusion to estimate and predict the evolving situation. A PSAW system must aggregate state estimates provided by lower level information fusion (LLIF) systems to help users understand key aspects of the aggregated situation and project its likely evolution. In order to do so, the PSAW system needs semantically rich representation to capture attributes of, relationships among, and processes associated with various kinds of entities in a situation, and the ability to reason under uncertainty about the situation.

Ontologies provide common semantics for expressing information about entities and relationships in the domain. Probabilistic ontologies augment standard ontologies with support for uncertainty management [Costa, 2005], providing a representation language rich enough to support PSAW. The probabilistic Web Ontology language (PR-OWL) is a representation language for probabilistic ontologies. PR-OWL 2 extends PR-OWL to provide better integration with OWL ontologies [Carvalho, 2011]. Multi-Entity Bayesian Networks (MEBN) is the logical basis of PR-OWL and PR-OWL 2 for the uncertainty representation and reasoning. MEBN combines First-Order Logic with Bayesian Networks [Pearl, 1988] for representing and reasoning about uncertainty in complex, knowledge-rich domains [Laskey, 2008]. MEBN goes beyond standard Bayesian networks to enable reasoning about an unknown number of entities interacting with each other in various types of relationships.
1.1 Suitability of MEBN for PSAW

In this section, we discuss why MEBN is a good knowledge representation language for Predictive Situation Awareness (PSAW). In order to grasp the idea more easily, we use the following example. Note that this instance is a simplified example used for illustration. A serious PSAW application may involve estimation and prediction about the meaning of a dynamically evolving situation involving many objects using evidence from a large number of sources.

Fig. 1.1 depicts a specific situation of interest for our simplified example. Our goal is to estimate the vehicle type (e.g., Tracked and Wheeled) of one or more target objects and the degree of danger (e.g., High and Low) of a specific region.

The rectangles in Fig. 1.1 mean instances of entities. Fig. 1.1 expresses two relations among entities. An inner rectangle which is shown within an outer rectangle means a part entity of an entity represented by the outer rectangle, so it means
composition or aggregation. The rectangle described by “Communicated = Y” specifies that the entities communicate with each other. Our system has been provided with the following evidence. At Time 1, a weather sensor has reported clear weather for Region 1.1. A geographic information system has reported that Region 1.1 is Off-road terrain.

Two vehicle objects, \( V1 \) and \( V2 \), have been detected by an imaging system, which has reported that the type of \( V2 \) is tracked and has failed to report a type for \( V1 \). A moving target indicator (MTI) sensor indicates that both vehicles are traveling slowly. A communications intelligence (COMINT) report indicates communications between \( V1 \) and \( V2 \). Given this evidence, we want to know the object type of both vehicles and the danger level of the Region 1.1.

We might consider using a Bayesian network (BN) [Pearl, 1988] to fuse these reports from multiple sources and answer the queries of interest. Fig. 1.2 shows a Bayesian network we might use for this problem.

![Figure 1.2 Bayesian Network of the Danger Assessment Problem](image-url)
Each box in the figure depicts a random variable (RV), or node. A label at the top of the box gives a name for the RV, the labels inside the boxes indicate its possible states, and the numbers indicate the probability of the state given our current evidence. For example, the VehicleType_v2 RV denotes the type of V2. It can have value either Wheeled or Tracked. Arcs among RVs in Fig. 1.2 represent direct dependence relationships. For example, ImageTypeReport_rpt1, the type recorded on imaging sensor report rpt1, depends on VehicleType_v2, the actual type of V2, the vehicle being observed by the sensor. Each RV has an associated marginal (if it doesn’t have a parent RV) or conditional (if it has some parent RV) probability distribution. For example, the Terrain-Type_region1_1 RV has a marginal probability distribution, P(TerrainType_region1_1), while the VehicleType_v1 RV has a conditional probability distribution, P(VehicleType_v1 | TerrainType_region1_1). RVs for which we have evidence are shown in gray and probabilities are set to 100% for the value that was actually observed. For example, recall that Region 1.1 was off-road terrain; thus, evidence for OffRoad is applied to the TerrainType_region1_1 RV. The probabilities shown for the RVs shown in yellow are conditional on all the evidence we have acquired, and are computed by an algorithm for updating probabilities via Bayes Rule. Given all the evidence, the model assigns 80% probability that the type of V1 is tracked, 94% probability that the type of V2 is tracked, and 80% probability that the danger level in Region 1.1 is high.

Manual construction of a BN like Fig. 1.2 is feasible, however what about situations containing hundreds of vehicles and reports? For such situations, MEBN allows us to build up a complex BN out of modular pieces. In other words, a MEBN
model, called an *MTheory*¹ which is a semantically rich representation of knowledge about a domain, is composed of *MFrags*, where each M_frag expresses a modular component of the domain knowledge. These MFrags can be instantiated and composed into larger BN models. For example, the *VehicleObject* M_frag in the *Danger Assessment* MTheory in Fig. 1.3 expresses knowledge of how the vehicle type is related to the terrain type. The green pentagons (e.g., *isA*(obj, VEHICLE), *isA*(rgn, REGION), and rgn = *Location*(obj) in the *VehicleObject* M_frag) are context RVs that express conditions under which the M_frag is valid. The context RVs in the *VehicleObject* M_frag mean that obj is a vehicle located in region rgn. The gray trapezoid input RVs (e.g., *TerrainType*(rgn) in the M_frag) have their distributions defined in other MFrags. The yellow oval resident RV (e.g., *VehicleType*(obj) in the M_frag) has its distribution described as the following, depends on the terrain type of rgn.

\[
\text{if some rgn have ( TerrainType = Road ) [} \\
\text{Tracked = .2,} \\
\text{Wheeled = .8} \\
\text{]} \text{ else if some rgn have ( TerrainType = OffRoad ) [} \\
\text{Tracked = .8,} \\
\text{Wheeled = .2} \\
\text{]} \text{ else [} \\
\text{Tracked = .5,} \\
\text{Wheeled = .5} \\
\text{]} \\
\]

The above class local distribution of the *VehicleType*(obj) RV is defined in a language called *Local Probability Description Language* (LPDL) (see Appendix D). The class local distribution specifies that if the terrain type of some rgn is *Road*, then the

¹ An MTheory which is used for PSAW is called a PSAW-MTheory.
conditional probability of the vehicle type of \( \text{obj} \) will be \( P(\text{VehicleType} = \text{Tracked} \mid \text{TerrainType} = \text{Road}) = 0.2 \), if the terrain type of some \( \text{rgn} \) is \( \text{OffRoad} \), then \( P(\text{VehicleType} = \text{Tracked} \mid \text{TerrainType} = \text{OffRoad}) = 0.8 \), and otherwise \( P(\text{VehicleType} = \text{Tracked} \mid \text{TerrainType} = \text{OffRoad}) = 0.5 \).

In the *Danger Assessment* MTheory in Fig. 1.3, there are 7 MFrags: *Speed*, *ImageTypeReport*, *VehicleObject*, *Danger*, *Weather*, *Region*, and *Reference*. This MTheory can generate many different BNs specialized to different situations using the SSBN algorithm [Laskey, 2008], as depicted in Fig. 1.5 below.
The SSBN algorithm generates a graph of a BN for a specific situation, as well as computing probability distributions of RVs in the BN conditional on the evidence. The generated BN is called a *Situation Specific Bayesian Network* (SSBN) and we call the probability distributions of the SSBN *instance local distributions* $L^I$, because they are derived from the class local distributions $L^C$. For example, given entity information, \( \{\text{rgn} = \{\text{Region1}\}, \text{obj} = \{\text{Vehicle1}\}\} \) for the VehicleObject MFragment, the SSBN algorithm can generate the following graph of the BN described by the nodes and arcs, and instance local distributions described by *Conditional Probability Table* (CPT) in Fig. 1.4.

![Figure 1.4 An example SSBN with instance local distributions derived from the Vehicle Object MFragment given entity information \( \{\text{rgn} = \{\text{Region1}\}, \text{obj} = \{\text{Vehicle1}\}\} \)](image)

Fig. 1.5 shows two SSBNs generated from the *Danger Assessment* MTheory. Case 1 represents two vehicles \( \{\text{obj} = \{v1, v2\}\} \) with two reports \( \{\text{rpt} = \{\text{rpt1, rpt2}\}\} \) in a single region \( \{\text{rgn} = \{\text{region1}_1\}\} \) at a single time \( \{t = \{t1\}\} \). Case 2 represents two vehicles with two reports in a single region over 3 time steps. In Case 1, the terrain type of the region, weather type, reported image type for the vehicles, and speed of the vehicles at time 1 are observed. Given the observations, the SSBN of Case 1 can be used
for estimating the type of the vehicles or danger level of the region. In Case 2, given the same observations, the SSBN of Case 2 can be used for predicting the speed of the vehicles at time 2 or 3.

![Figure 1.5 Generating Several SSBNs](image)

As we saw in the above example, MEBN has been applied to situation assessment and predictive situation awareness [Laskey, 2000][Wright et al., 2002][Costa et al., 2005][Costa et al., 2009][Carvalho et al., 2010][Costa et al., 2012]. Thus, its increased expressive power over ordinary BNs is an advantage for situation assessment and PSAW.
1.2 Problem Statement

In previous applications of MEBN to Situation Assessment (SA) and Predictive Situation Awareness (PSAW), the MTheory or MEBN model was constructed manually by a domain expert using a MEBN modeling process such as Uncertainty Modeling Process for Semantic Technology (UMP-ST) [Carvalho et al., 2016]. Manual MEBN modeling is a labor-intensive and insufficiently agile process. Therefore, the research problem addressed by this dissertation is:

*Traditional MEBN modeling for PSAW is labor-intensive and insufficiently agile.*

To address this problem, we propose two supportive methodologies: automation and reference models. For automation, we use a machine learning method in which all or part of a MEBN model is learned from observed data on previous situations. Specifically, for the observed data, we focus on relational data represented in the Relational Model (RM) which is a database model based on first-order predicate logic [Codd, 1969; Codd, 1970] and the most widely used data model in the world. A reference model is an abstract framework to which a developer refers in order to develop a specific model. As such a reference model, we propose a PSAW-MEBN reference model which is a reference model for a PSAW-MTheory which specifies references for MFrags, RVs, relationships of RVs, and entities. The PSAW-MEBN reference model can support the design of a PSAW-MTheory and improve the quality of the PSAW-MTheory. To convert relational databases to MEBN models, we propose a mapping model, called MEBN-RM, which provides a specification for the conversion.

Fig. 1.6 depicts objects of our study; Relational Data, MEBN Model, SSBN for PSAW, and Inferences to support PSAW. A relational dataset is used to learn a MEBN
model. An SSBN is constructed from the MEBN model. The inference results to support PSAW are outputted using the SSBN. This process depicted in Fig. 1.6 should execute smoothly and accurately to achieve enhanced PSAW. Generating the SSBN from the MEBN model was studied by [Laskey, 2008]. In this research, we focus on a MEBN learning framework, which is the main contribution, containing (1) MEBN-RM, (2) a PSAW-MEBN reference model, and (3) a MEBN parameter learning. To achieve the research objective, we decompose the above main research problem into the following specific problem statements:

- **Problem 1** There is no process defining how to use relational data for MEBN Learning for PSAW.
- **Problem 2** There are no guidelines for representing a MEBN model for PSAW.
• **Problem 3** There is no efficient MEBN learning process from relational data.

• **Problem 4** MEBN modeling and reasoning for PSAW which are introduced in this research should be evaluated.

Obviously, the domain of machine learning for PSAW is huge, and many problems need to be addressed. This dissertation focuses on some of the most important challenges with a focus on PSAW.

The starting point of our research is the data used for learning. For purposes of this dissertation, we assume that the data used to learn a MEBN model for PSAW are contained in a single relational database. In a realistic scenario, the data may be distributed among multiple relational databases. In this case, we assume that an integration method [Bricha et al., 1999][Morgenstern, 1999][Schmitt & Saake, 2005] is used to do multiple queries and join the results of queries. The integration method may contain a convention or rule to prevent conflicts among databases (e.g., key uniqueness problem), when a query for the multiple databases occurs. Given the integration method, the situation of the multiple relational databases can be emulated as a single relational database, because the integration method works as if there is only the single relational database. This dissertation does not address how to manage the situation of multiple relational databases, but we start from the single relational database situation and assume that the multiple relational databases situation can be handled by the integrating method.

In general, data used for PSAW may come in a variety of forms such as triple stores or unstructured text repositories. Non-relational databases, called *NoSQL*, are receiving increasing attention [Han et al., 2011]. In the era of Big Data, we may need a
scalable and flexible database to manage the many and varied types of data. However, *NoSQL* is still immature, with theory, standards and tools lagging behind the relational model. In this research, we focus on the *Relational Model* (RM), which is a very well-known and standardized database model, as a source data model to develop a MEBN theory. Future work will consider extensions to NoSQL data.

To use RM, it is necessary to define how to convert elements of RM to elements of MEBN, so a mapping rule between MEBN and RM, called MEBN-RM, is introduced (Problem 1). For MEBN-RM, we find a link between the two languages (RM and MEBN) and define the link.

MEBN has the ability to express a complex structure of many domains as well as the PSAW domain. For Problem 2, we define a reference model for MEBN for PSAW, called a PSAW-MEBN reference model. The PSAW-MEBN reference model enables us to easily develop a MEBN model for PSAW by supporting the design of a MEBN model for PSAW. Such a MEBN model is called a PSAW-MTheory. The PSAW-MEBN reference model should include the elements of the situation such as targets, sensors, activities, locations, time, etc, as well as it being capable of generalization to express a variety of situations for PSAW (e.g., PSAW for naval operation and PSAW for critical infrastructure protection). Also, the model specifies reference MFrags, RVs, and entities which support the design of a PSAW-MTheory designed to reason about PSAW questions (e.g., “How many military vehicles are we going to encounter?”, “How high will the level of danger for an enterprise be?”, and “Where will the event occur?”).
Problems 1 and 2 are about Data and Model in Fig. 1.6, while Problem 3 is about Learning. Greater automation through machine learning may save labor and enhance agility. For this reason, a basic MEBN learning method was suggested [Park et al., 2013a][Park et al., 2013b]. Although technologies for machine learning have improved dramatically, the necessary capabilities to learn a MEBN model from data do not yet exist. The search space for building the model is too large and complex to investigate all possible structures, variables, and parameters. For this reason, we introduce a process for Human-aided MEBN learning for PSAW (HMLP) which is a framework to develop a MEBN model by combining domain expertise with data. HMLP contains (1) MEBN-RM, (2) a PSAW-MEBN reference model, and (3) MEBN learning from data in RM to develop a MEBN model for PSAW efficiently and effectively. For MEBN learning, HMLP relies partially on expert knowledge and insight to reduce the search space. In this research, MEBN learning is to learn an optimal MEBN model which fits well an observed dataset in RM. MEBN learning can be classified into two types: One is MEBN structure learning (e.g., finding optimal M_frag structures) and another is MEBN parameter learning (e.g., finding an optimal set of parameters for class local distributions). In this dissertation, we focus on MEBN parameter learning and leave MEBN structure learning as a future research topic.

Lastly, Problem 4 is to evaluate the usefulness of our methodologies (i.e., MEBN-RM, the PSAW-MEBN reference model, HMLP, and SSBN reasoning for PSAW). Especially, we focus on evaluation for HMLP in terms of (1) efficiency and (2) effectiveness. In this dissertation, efficiency for HMLP is measured by how HMLP
improves the development time for a PSAW-MTheory. Effectiveness for HMLP is assessed by how well the PSAW-MTheory fits a test dataset. For this, we introduce three example use cases: a Critical Infrastructure Defense system, a Maritime Domain Awareness system, and a Smart Manufacturing system. Also, we conduct an experiment to compare HMLP and UMP-ST (the manual MEBN modeling) in terms of development efficiency.

For the first example use case, we aim to develop a PSAW-MTheory for a proof-of-concept software PSAW system, called HERALD, which wards off attacks against critical infrastructure by means of early detection of threatening targets, identification of the targets, estimation of the target’s activities, and prediction of virtual short-term future situations. HERALD contains a PSAW scenario simulator simulating a critical infrastructure defense problem with two attack scenarios (terrorist and UAV attacks). From the simulation, training and test datasets are generated. Using HMLP, we learn the PSAW-MTheory from the training dataset. The PSAW-MTheory is used for reasoning about the simulated situation. The reasoning results, then, are evaluated using the test dataset. Especially, we focus on evaluation for HMLP in terms of efficiency and effectiveness. For efficiency, HMLP should shorten the development period of the PSAW-MTheory. For effectiveness, the PSAW-MTheory should fit well to a training dataset for an inference task in which the PSAW-MTheory will estimate and accurately predict the evolution of the situation.

For the second example use case, we apply HMLP to PRobabilistic OntoloGies for Net-Centric Operational Systems (PROGNOS), a system that supports Maritime
Domain Awareness (MDA). PROGNOS contains a PROGNOS probabilistic ontology which provides semantically aware uncertainty management to support fusion of heterogeneous input and probabilistic assessment of situations to improve MDA. Developing probabilistic ontologies can be greatly facilitated by the use of a modeling framework such as the Uncertainty Modeling Process for Semantic Technology (UMP-ST) [Carvalho et al., 2016]. An example of using UMP-ST was the development of the PROGNOS probabilistic ontology. However, manually developing and maintaining the PROGNOS probabilistic ontology using UMP-ST was inefficient. HMLP may enhance agility in the development of the PROGNOS probabilistic ontology. In this dissertation, HMLP is used to develop an extended PROGNOS probabilistic ontology and we present a comparison between UMP-ST and HMLP in terms of efficiency.

For the third example use case, we apply HMLP to a Predictive Situation Awareness for Manufacturing (or MSAW) system. The MSAW system, requires comprehensive knowledge representation for various manufacturing situations and expeditious reasoning methods to estimate current situations as well as to predict future situations. MSAW models expressed as MEBN, called MSAW-MEBN models, can be constructed using HMLP. In this use case, we introduce how HMLP can be used to develop the MSAW-MEBN models in a short time.

Finally, we conduct an experiment to compare two MEBN development processes (UMP-ST and HMLP) in terms of development time. In this experiment, there are two groups (Group A for UMP-ST and Group B for HMLP). Both groups are required to develop a MEBN model from stakeholder requirements. MEBN models developed by
participants in both groups are compared in terms of accuracy to assess how similar results are derived. The accuracy from HMLP should be the same as or better than the accuracy from UMP-ST. Also, development times are compared in terms of development efficiency to assess how much improvement has been made with the use of HMLP.

1.3 Thesis Statement and Scope

For this dissertation,

_I propose to develop a process and methods for human-assisted learning of Multi-Entity Bayesian Networks for Predictive Situation Awareness using Relational Data._

Through this research, we will attempt to introduce a new framework for MEBN learning from data in RM, especially for PSAW. For the five problems in Section 1.2, the following research products will be produced.

- **Product 1.** A MEBN-RM mapping model
- **Product 2.** A PSAW-MEBN reference model
- **Product 3.** A process for Human-aided MEBN learning for PSAW
- **Product 4.** Three example use cases in which we will show how our methodologies can improve development for some PSAW MTheories

In this dissertation, we establish the following delimitations to the characteristics of data and learning method we are treating. The focus of this dissertation is not to deal with the following. (1) Various types of data structure; How to treat various types of data structure for MEBN Learning (e.g., text, OWL ontology, and First Order Logic)? (2) Multiple distributed data learning; how to learn an MTheory from data in multiple distributed databases? (3) Learning from insufficient evidence; how to learn an MTheory
from not enough observations? (see 7.4.4 Learning from incomplete Data) (4) A small number of positive cases; How to learn parameters and structures from rare events? (5) Aggregating influence problem; How to learn an aggregating function in an aggregating situation where an instance child random variable depends on multiple instance parents which are generated from an identical class random variable? (see Section 7.4.1 Aggregating influence problem).

1.4 Contributions

The presented methodology has the following contributions.

- The research and implementation for the mapping model between MEBN and RM
- The research for the PSAW-MEBN reference model
- The initial research and implementation for MEBN learning
- The development of three proof of concept PSAW systems using HMLP

(1) The presented methodology provides the MEBN-RM mapping model, which is useful in its own right and as a guide for developing mapping models for other relational modeling frameworks. Also, the mapping model could be extended to other statistical relational models (e.g., Markov Logic Networks [Domingos et al., 2007]). (2) The PSAW-MEBN reference model provides guidance for designing a MEBN model for PSAW, supporting more agile design of MEBN models for PSAW. (3) Although concepts of MEBN Learning have been presented previously [Costa, 2005][Laskey, 2008], our research is the first practical research on MEBN learning and the first MEBN learning implementation. Especially, this dissertation treats MEBN parameter learning
and provides basic definitions for MEBN learning. (4) The use cases which will be applied will be useful as a guideline for other PSAW applications on HMLP.

1.5 Organization of the Dissertation

Chapter 2 provides background information on topics including Data Fusion, Situation Awareness, Bayesian Network, Bayesian Network learning, Multi-Entity Bayesian Network, Relational Data, and UMP-ST. Chapter 3 introduces the MEBN-RM mapping model. MEBN-RM contains several mapping rules such as Entity Mapping, Resident Node Mapping, Relation Schema and MFragment Mapping, and Relational Database Schema and MTheory Mapping. In Chapter 4, we describe the PSAW problem and introduce the PSAW-MEBN reference model specifying Reference Entities, Reference Random Variables, and Reference MFrags. Chapter 5 introduces HMLP with a simple illustrative example. In the chapter, a MEBN parameter learning for discrete and continuous variables is presented. Chapter 6 presents three use cases using the methodologies this research presents and an experiment for comparing UMP-ST and HMLP. Chapter 7 presents the conclusion of this research with a summary and future work.
CHAPTER TWO: BACKGROUND

This chapter provides background information about Data Fusion (DF), Situation Awareness (SAW), Bayesian Networks (BN), BN Learning, Multi-Entity Bayesian Networks (MEBN), Relation Model (RM), and Uncertainty Modeling Process for Semantic Technology (UMP-ST). Section 2.1 introduces the JDL Data Fusion Model and SAW to provide initial concepts for PSAW. PSAW, a specialization of SAW that emphasizes predicting an evolving situation, will be discussed in Chapter 4. In Section 2.2, BN, which is a foundation of MEBN, is introduced briefly. In Section 2.3, MEBN as a representation formalism for PSAW is presented with some definitions and an example for MEBN. The methodology which this research proposes uses a relational model (RM) as a data schema for a dataset. In Section 2.4, the relational model is described. HMLP, one of the topics of this research, in Chapter 5 is a modification of UMP-ST, so UMP-ST is introduced in Section 2.5.

2.1 From Data Fusion to Situation Awareness

2.1.1 Data Fusion

In 1985, U.S. Joint Directors of Laboratories (JDL) Data Fusion Group proposed the Data Fusion Model (called JDL model) [White, 1988] which was revised to [Steinberg et al., 1998][Llinas et al., 2004][Liggins, et al., 2008]. JDL defines data fusion as “Data fusion is a process dealing with the association, correlation, and combination of
data and information from single and multiple sources to achieve refined position and identity estimates, and complete and timely assessments of situations and threats as well as their significance. [White, 1991]”. A concise definition for data fusion was presented as “Data fusion is the process of combining data or information to estimate or predict entity states. [Steinberg et al., 1998]”.

Many definitions of data fusion, including the above two definitions, were reviewed and discussed in [Boström et al., 2007]. The process of data fusion means not only combining signals from low-level sensors but also combining knowledge from high-level sources to estimate and predict entity states. The process involves induction, deduction, and abduction, as does the thinking process of a human being. To implement the process, the JDL model provides four level stages as: “Level 0: Signal/feature assessment. Estimation of signal or feature states. Signals and features may be defined as patterns that are inferred from observations or measurements. These may be static or dynamic and may have locatable or causal origins (e.g., an emitter, a weather front, etc.). Level 1: Entity assessment. Estimation of entity parametric and attributive states (i.e., of entities considered as individuals). Level 2: Situation assessment. Estimation of the structures of parts of reality (i.e., of sets of relationships among entities and their implications for the states of the related entities). Level 3: Impact assessment. Estimation of the utility/cost of signal, entity, or situation states, including predicted utility/cost given a system’s alternative courses of action. Level 4: Process assessment. A system’s self-estimation of its performance as compared to desired states and measures of effectiveness (MOEs). [Liggins, et al., 2008]”
In Level 0, signal or feature states of an attribute of an entity are estimated from one or more signal observations (e.g., imagery, electromagnetic, and acoustic data) detected by sensors of various types. For example, from imagery data depicting movement of a vehicle, the size, velocity, shape or color of the moving vehicle can be estimated using a feature extraction method. The feature extraction method detects a pattern of signals or features (e.g., for the size of the moving vehicle, an image range from the moving vehicle in the imagery data can be regarded as a pattern), and it is mapped to a signal or feature state (e.g., small, medium, or large size of the moving vehicle). In Level 1, parametric and attributive states of entities (e.g., identity, location, track, and activity) considered as individuals are estimated from one or more signals, features, and entity states. The state can be considered as the state of a random variable or random vector. Example state variables include temperature, location, functional class, an attribute, or activity state of an entity. For instance, in the Danger Assessment example in Chapter 1, estimating the types, speeds, and locations of the vehicles would be considered Level 1 fusion. In level 2, relationships or situations are estimated from one or more entity states and relationships. For estimation of a situation, an inference method is usually required. For example, Bayesian Networks [Steinberg, 2003] allow predicting a future situation based on current and historical situations. For instance, in the Danger Assessment example, estimating the danger level of the Region 1.1, a result from interactive influences between the moving vehicles, would be considered Level 2 fusion. In level 3, the impact of a signal, entity, or situation for a goal or mission is estimated. Impact assessment for threat assessment involves combining multiple sources of
information to infer a conditional or counterfactual situation. Based on the estimated situation, known plans, and predicted reactions, outcome and cost in terms of one’s goal or mission are estimated or predicted. For instance, in the *Danger Assessment* example, evaluating and predicting consequences of a situation over time (e.g., the combat performance of an attacker or a defender) would be considered Level 3 fusion. In level 4, measures of performance (MOPs) and measures of effectiveness (MOEs) of the system are estimated from the desired set of the system states and responses by performance analysis. Impact assessment of level 3 provides the impact to the goal or mission as technical performance measures (TPMs) in systems engineering. To assure of achievement of a goal, MOPs derived from MOEs and composed of TPMs should be estimated in level 4 [Liggins, et al., 2008].

2.1.2 Situation Awareness
The concept of situation awareness (SAW) was first introduced during World War I [Gilson, 1995]. Although people realized its necessity a long time ago, the aviation industry started using in-depth studies of SAW to facilitate better performance for pilots and air traffic control system [Jensen, 1997]. During the past quarter-century, SAW has become a critical research theme, because of its importance.

The concept of SAW has been applied various domains which involve humans performing tasks in complex and dynamic systems. Examples include aviation [Jensen, 1997], air traffic control [Endsley & Smolensky, 1998], safety control [Salmon et al., 2006], automobile driving [Zheng et al., 2004], and C4I (Command, control, communication, computers, and intelligence) systems [Stanton et al., 2001]. In the
literature related to SAW, Breton and Rousseau used a systematic approach to classify twenty-six SAW definitions [Breton & Rousseau, 2001]. They found a set of elements of SAW. They classified SAW into two top-level categories: State- and Process-oriented SAW. In their definition, Process-oriented SAW focuses on the link between the situation and the cognitive processes generating SAW, while State-oriented SAW focuses on the link between the situation and an internal representation of elements present in the situation. In the definitions of Process-oriented SAW, they include various processes such as perception, comprehension, projection, and action. In the definitions of State-oriented SAW, they classify awareness and knowing as a state of situation. According to the most commonly cited definition, SAW is composed of three processes;

“Level 1: the perception of the elements in the environment within a volume of time and space, Level 2: the comprehension of their meaning, and Level 3: the projection of their status in the near future. [Endsley, 1988; Endsley et al., 2003]”

In Level 1, the states, attributes, and dynamics of relevant elements in the environment should be detected accurately in a purpose of a system which uses SAW [Endsley et al., 2003]. A perception component of the system receives cues or signals of multiple sensors of the component. The sensors are separated from each other at this level. However, the sensors and their signals should be selected for the purpose of the system. In some cases, a critical sensor or signal is not present, so this level must ensure to perceive correct and necessary signals for the purpose of the system, and must compensate appropriately for degraded or missing inputs. Level 2 is to understand what the signals mean in relation to the purpose of the system [Endsley et al., 2003]. For
example, isolated words are perceived (Level 1). Then, by integrating the words in a
certain manner, the individual words provide meaning to the sentence in which they
occur (Level 2). By integrating signals, the signals form information. Then, the combined
information is measured in how the information impacts the purpose of the system. As a
result, a picture of the current situation showing relationships between information and
impacts to the purpose is formed. At this level, understanding situation correctly is
required. Level 3 is to predict future states of the information in relation to the purpose of
the system [Endsley et al., 2003]. For example, as a driver, by approaching a destination,
the driver may estimate the distance between the current and destination location through
time using her knowledge. Level 3 SAW is the estimation or prediction for the states of
information through time. Time plays a significant role of understanding situation. A
snapshot of situations is not enough. For example, we may want to know an answer about
“how much time is available until some event occurs or some action must be taken?”
[Endsley et al., 2003].

[Lambert, 2001] noted the similarity between levels 1 ~ 3 of the JDL model and
Endsley’s SAW model. Both models describe estimation and prediction of a situation as
part of reality. Note, however, that Endsley’s SAW model provides an abstraction of the
processes of perception, comprehension, and projection, but does not provide specific
processes of how to implement perception, comprehension, and projection, while the JDL
model provides somewhat specific processes through levels 0 ~ 4.
2.2 Bayesian Network

A Bayesian Network (BN) [Pearl, 1988] is a probabilistic graphical model that represents a joint distribution on a set of random variables in a compact form that exploits conditional independence relationships among the random variables. The random variables (RVs) are represented as nodes in a directed acyclic graph (DAG) in which a directed edge represents a direct dependency between two nodes and no directed cycles are allowed in the graph. Bayesian Networks have become a powerful tool for representing uncertain knowledge and performing inference under uncertainty. They have been applied in many domains, such as Image Understanding, Data Fusion, Medical Diagnosis, and Fraud Detection, and have become a powerful tool in inference for the real world.

**Definition 2.1 (Bayesian Network)** A Bayesian Network (BN) represents a joint distribution of the set of random variables \(X = \{X_1, X_2, \ldots, X_n\}\), denoted by a pair \([G, \Theta]\), where \(G\) is a directed acyclic graph whose nodes are associated with the random variables and \(\Theta = \{\theta_1, \ldots, \theta_n\}\) is a set of local distributions \(\theta\) for each \(X_i\) given its parents \(\text{Pa}(X_i)\), denoted by \(\theta = P(X_i | \text{Pa}(X_i))\).

A joint distribution represented by a Bayesian network can be decomposed as a product of the local distributions of the Bayesian network using the probability chain rule. The joint distribution \(P\) is given as follows:

\[
P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \text{Pa}(X_i)), \tag{2.1}
\]
where $\text{Pa}(X_i)$ is a set of parents of $X_i$ in $G$.

A Bayesian network can be classified into three categories: (1) a *Discrete Bayesian Network* contains only discrete RVs, (2) a *Continuous Bayesian Network* contains only continuous RVs, and (3) a *Hybrid Bayesian Network* (HBN) contains both discrete and continuous RVs. For example, Fig. 1.4 in Chapter 1 shows the Discrete Bayesian network containing the discrete RVs $\text{VehicleType}_{\text{Vehicle1}}$ and $\text{TerrainType}_{\text{region1}_1}$. If there is a continuous RV $\text{Speed}_{\text{Vehicle1}}$ representing the speed of the vehicle 1 and it depends on the discrete RV $\text{VehicleType}_{\text{Vehicle1}}$, the example BN in Fig. 1.4 in Chapter 1 will be an HBN. The continuous RV $\text{Speed}_{\text{Vehicle1}}$ can be assigned to a real value with a continuous probability distribution (e.g., normal or log-normal distribution). Note that Appendix A introduces BN parameter learning.

### 2.3 Multi-Entity Bayesian Network

In this section, we describe MEBN, a graphical representation for MEBN, and a script form of MEBN. Details can be found in [Laskey, 2008].

#### 2.3.1 Definition of MEBN

MEBN allows compact representation of repeated structure in a joint distribution on a set of random variables. Also, MEBN allows random variables to be defined as templates than can be repeatedly instantiated to construct probabilistic models with repeated structure. MEBN represents domain knowledge using an MTheory, a collection of MFrags (see Fig. 1.3). An MFragment is a fragment of a graphical model that is a template for probabilistic relationships among instances of its random variables. Random variables
(RVs) can contain ordinary variables, which can be instantiated for different domain entities. We can think of an MFragment as a class which can generate instances of BN fragments. These can then be assembled into a Bayesian network, called a *situation-specific Bayesian Network* (SSBN), using an SSBN algorithm [Laskey, 2008]. A given MTheory can be used to construct many different SSBNs for different situations.

**Figure 2.1 Danger MFragment**

In Chapter 1, we introduced the *Danger Assessment* MTheory in Fig. 1.3. There are 7 MFrags: *Speed*, *ImageTypeReport*, *VehicleObject*, *Danger*, *Weather*, *Region*, and *Reference*. Fig. 2.1 shows the *Danger* MFragment (from the *Danger Assessment* MTheory) used for an illustrative example of an MFragment. The *Danger* MFragment represents probabilistic knowledge of how the level of danger of a region is measured depending on the vehicle type of detected objects. For example, if in a region there are many tracked vehicles (e.g., Tanks), the danger level of the region will be high. An MFragment consists of a set of resident
nodes, a set of context nodes, a set of input nodes, an acyclic directed graph for the nodes, and a set of class local distributions (CLD) for the nodes. The context nodes for this MFrag (shown as pentagons in the figure) show that this MFrag applies when a vehicle entity is substituted for the ordinary variable obj, a region entity is substituted for the ordinary variable rgn, and a vehicle obj is located in region rgn. The context node rgn = Location(obj) constrains the values of obj and rgn from the possible instances of vehicle and region. For example, suppose v1 and v2 are vehicles and r1 is a region in which the only v1 is located. The context node rgn = Location(obj) will allow only an instance of (v1, r1) to be selected, but not (v2, r1), because r1 is not the location of v2. Next, we see the input node VehicleType(obj), depicted as a trapezoid. Input nodes are nodes whose distribution is defined in another MFrag. In Fig. 2.1, the node Danger_Level(rgn) is a resident node, which means its distribution is defined in the MFrag of the figure. This node Danger_Level(rgn) might be an input node of some other MFrag, where it would appear as a trapezoid. Like the graph of a BN, the fragment graph shows statistical dependencies. The class local distribution for Danger_Level(rgn) in the figure describes its probability distribution as a function of the input nodes given the instances that satisfy the context nodes. The class local distribution, \( L^C \) can be used to produce an instance local distribution, \( L^I \), in a SSBN. In our example, the argument, rgn, is the region variable. If the situation involves two regions (r1 and r2), then Danger_Level(r1) and Danger_Level(r2) will be instantiated. The class local distribution in Fig. 2.1 is defined in a language called Local Probability Description Language (LPDL), defined in Appendix D. In our example, the probabilities of the states, high and low, of the Danger_Level(rgn)
RV are defined as a function of the values, high and low, of instances \( rgn = Location(obj) \) of the parent nodes that satisfy the context constraints. For the high state in the first if-scope in the LPD definition of Fig. 2.1, the probability value is assigned by the function described by \( 1 - 1 / \text{CARDINALITY}(obj) \). The CARDINALITY function returns the number of instances of \( obj \) satisfying the if-condition. For example, in the LPD expression of Fig. 2.1, if the situation involves three vehicles and two of them are tracked, then the CARDINALITY function will return 2. We see that as the number of tracked vehicles becomes very large, the function, \( 1 - 1 / \text{CARDINALITY}(obj) \), will tend to 1. This means the danger level of the region will be very high.

From this Danger MFragment, diverse situation-specific Bayesian Networks (SSBN) can be generated depending on the specific entities involved in the situation. For example, a single region entity called \( \text{region1}_1 \) and three vehicle entities called \( v1, v2, \) and \( v3 \) will give rise to the SSBN in Fig. 2.2, with the conditional probability table (CPT) for \( \text{Danger Level}_\text{region1}_1 \) as shown. This transformation from the Danger MFragment to the SSBN is called unrolling. The unrolled CPT in Fig. 2.2 is the instance local distribution.

![Figure 2.2 SSBN from Danger MFragment (given v1, v2, and v3 as vehicle, and region1_1 as region)](image-url)
Alternatively, we might model the \textit{Danger\_Level(rgn)} RV as a continuous random variable. For a continuous resident node, the class local distribution is defined by a continuous probability density function. The class local distribution of the continuous resident node can be described by LPDL, also. Suppose its LPDL is

\begin{verbatim}
if some obj have (VehicleType = Tracked) [
    10 * CARDINALITY(obj) + NormalDist(10, 5)
] else [
    NormalDist(10, 5)
]
\end{verbatim}

The meaning of the above LPDL definition is that the degree of the danger in the region is 10 * the number of tracked vehicles plus a normally distributed error with mean 10 and variance 5. If the \textit{Danger} M_frag is unrolled with the same entities with the previous example in Fig. 2.2, then we have the following unrolled hybrid SSBN (containing at least one continuous RV) in Fig. 2.3. Currently, LPDL limits continuous nodes to conditional linear Gaussian (CLG) distributions [Sun et al., 2010], defined as:

\begin{equation}
p(N \mid \text{Pa}(N), CF_j) = \mathcal{N}(m + b_1P_1 + b_2P_2 + \ldots + b_nP_n, \sigma^2),
\end{equation}

where \text{Pa()} is a set of continuous parent resident nodes of the continuous resident node, \textit{N}, having \{\textit{P}_1, \ldots, \textit{P}_n\}, \textit{CF}_j is a \textit{j}-th configuration of the discrete parents of \textit{N}, \textit{m} is a regression intercept, \textit{\sigma}^2 is a conditional variance, and \textit{b}_i is regression coefficient.
From three discrete parent nodes, 8 configurations are generated and for each configuration described by the If-Then statement, the generated normal distributions are assigned by a script shown at the bottom of Fig. 2.3. The generated script is called Conditional Probability Script Language (CPSL) (Appendix E) [Sun et al., 2011].

Using the above MTheory example, we define elements of MTheory more precisely. The following definitions are taken from [Laskey, 2008].

**Definition 2.2 (MFragment)** An MFragment $F$, or MEBN fragment, consists of: (i) a set $C$ of context nodes, which represent conditions under which the distribution defined in the MFragment is valid; (ii) a set $I$ of input nodes, which have their distributions defined elsewhere and condition the distributions defined in the MFragment; (iii) a set $R$ of resident nodes, whose distributions are defined in the MFragment; (iv) an acyclic directed graph $G$.

---

2 Bold italic letters are used to denote sets.
whose nodes are associated with resident and input nodes; and (iv) a set $\mathbf{L}^C$ of class local distributions, in which an element of $\mathbf{L}^C$ is associated with each resident node.

The nodes in an MFragment are different from the nodes in a common Bayesian network. A node in a common BN represents a single random variable, whereas a node in an MFragment represents a collection of RVs: those formed by replacing the ordinary variables with identifiers of entity instances that meet the context conditions. To emphasize the distinction, we call the resident nodes in the MFragment MBEN nodes, or MNodes.

MNodes correspond to predicates (for true/false RVs) or terms (for other RVs) of first-order logic. An MNode is written as a predicate or term followed by a parenthesized list of ordinary variables as arguments.

**Definition 2.3 (MNode)** An MNode, or MEBN Node, is a random variable $N(\mathbf{f})$ corresponding to an $n$-ary function or predicate of first-order logic, a list of $n$ arguments consisting of ordinary variables, a set of mutually exclusive and collectively exhaustive possible values, and an associated class local distribution. The special values `true` and `false` are the possible values for predicates, but may not be possible values for functions. The RVs associated with the MNode are constructed by substituting domain entities for the $n$ arguments of the function or predicate. The class local distribution specifies how to define local distributions for these RVs.

For example, the node `Danger_Level(rgn)` in Fig. 2.1 is an MNode corresponding to the FOL function `Danger_Level(rgn)`. It has two possible values (i.e., `High` and `Low`). This MNode is associated with the class local distribution, $\mathbf{L}^C$ in Fig. 2.1. The MNode is
used as a template for the distributions of instance RVs (e.g., Danger_Level_region1_1 in Fig. 2.2) created when an SSBN is constructed from the M Frag associated with the MNode.

**Definition 2.4 (MTheory)** An *MTheory* *M*, or MEBN Theory, is a collection of MFrags that satisfies conditions given in [Laskey, 2008] ensuring the existence of a unique joint distribution over its random variables.

An MTheory is a collection of MFrags that defines a consistent joint distribution over random variables describing a domain. The MFrags forming an MTheory should be mutually consistent. To ensure consistency, conditions must be satisfied such as no-cycle, bounded causal depth, unique home MFrags, and recursive specification condition [Laskey, 2008]. No-cycle means that the generated SSBN will contain no directed cycles. Bounded causal depth means that depth from a root node to a leaf node of an instance SSBN should be finite. Unique home MFrags means that each random variable has its distribution defined in a single MFragment, called its home MFragment. Recursive specification means that MEBN provides a means for defining the distribution for an RV depending on an ordered ordinary variable from previous instances of the RV.

For a common Bayesian Network or SSBN, any node in the SSBN has a fixed number of parent nodes. An assignment of value to each parent node *i* is called a *parent configuration* *PCi*. A node can have a set of the parent configurations *PC* = {*PC1*, *PC2*, …, *PCn*}. For example, the node *Danger_Level_region1_1* (Fig. 2.2) can have a parent configuration *PC* = {*VehicleType_v1* = *Tracked*, *VehicleType_v2* = *Tracked*, *VehicleType_v3* = *Wheeled*} or simply {*T*, *T*, *W*}. Also, the node
Danger Level region1_1 (Fig. 2.2) can have a set of 8 parent configurations $\mathbf{PC} = \{PC_1 = \{T, T, T\}, PC_2 = \{T, T, W\}, PC_3 = \{T, W, T\}, PC_4 = \{T, W, W\}, PC_5 = \{W, T, T\}, PC_6 = \{W, T, W\}, PC_7 = \{W, W, T\}, PC_8 = \{W, W, W\}\}$ as shown the CPT in Fig. 2.2.

MEBN contains an Isa random variable (RV) which is a special RV representing the type of an entity as defined by the following. An Isa RV is commonly used for a context node, which is used to specify constraints under which the local distributions apply.

**Definition 2.5 (Isa random variable)** An *IsA random variable*, $\text{IsA}(ov, tp)$, is an RV corresponding to a 2-argument FOL predicate. The IsA RV has value *true* when its second argument $tp$ is filled by the type of its first argument $ov$ and *false* otherwise.

For example, $\text{Isa}(p, \text{PERSON})$ is an Isa RV. It contains two arguments; $p$ as $ov$ and PERSON as $tp$.

### 2.3.2 A Script for MEBN

Fig. 1.3 shows a graphical representation for an MTheory. In this section, we introduce a script representing an MTheory. This script is useful to manage contents of an MTheory. The *Danger Assessment* MTheory in Fig. 1.3 can be represented by the following script (MTheory 2.1).

**MTheory 2.1: Part of Script MTheory for Danger Assessment**

```
1 [F: ImageTypeReport
2   [C: Isa (obj, Vehicle)] [C: Isa (rgn, Region)] [C: Isa (rpt, Report)]
3   [C: rgn = Location (obj)] [C: obj = ReportedObject (rpt)]
4   [R: ImageTypeReport (rpt)
5     [IP: WeatherType (rgn)]
6     [IP: VehicleType (obj)]
7   ]
8 ]
9 [F: Weather
```
The script contains several predefined single letters (F, C, R, IP, RP, and L). The single letters, F, C, and R denote an MFragment, a context node, and a resident node, respectively. For a resident node (e.g., Y) in an MFragment, a resident parent (RP) node (e.g., X), which is defined in the MFragment, is denoted as RP (e.g., [R: Y [RP: X]]). For an input node, we use a single letter IP. Each node can contain a CLD denoted as L. For example, suppose that there is a CLD type called WeatherCLD. If the resident node WeatherType in Line 11 uses the CLD type WeatherCLD, the resident node WeatherType can be represented as [R: WeatherType (rgn) [L: WeatherCLD]].

2.4 Data for MEBN Learning from RM
The data used to support PSAW may be represented in a variety of forms. For example, data could be text, relational data, OWL ontologies, BNs, and others. Depending on the type of data, the method of MEBN learning would be different. For each type of data, a matching process between elements of MEBN and elements of data structure would need to be defined. For instance, if relational data is used, then its data elements such as attributes, value domain, table, key and others should be converted to MEBN elements such as random variables, states, Mfragments, entities, etc. In this research, we focus on relational data for source data of MEBN Learning, because it is considered as the most commonly used data model. There is more discussion about this in Chapter 3.
2.4.1 Relational Model

In 1969, Edgar F. Codd proposed the *Relational Model* (RM) as a database model based on first-order predicate logic [Codd, 1969; Codd, 1970]. The RM is the most popular database model. A *relational database* (RDB) is a database that uses the RM to describe and organize data. In the RM, data are organized as a collection of relations. A *relation* is an abstract definition of a class of entities or a relationship that can hold between classes of entities. An *instance of a relation* is depicted as a table in which each column is an attribute of the relation and each row, also called a *tuple*, contains the value of each attribute for an individual entity of the class represented by the relation. An entry in the table, called a *cell*, is the value of the attribute associated with the column for the entity associated with the row. A *key* for a relation is one or more attributes that uniquely identify a particular domain entity or row. A *primary key* uniquely identifies the individual entities in the relation. A *foreign key* points to the primary key in another relation. The *cardinality* of a relation is the number of rows in the table, i.e., the number of unique entities of the type represented by the relation. The *degree* of the relation is the number of columns in the table, i.e., the number of attributes of entities of the type represented by the relation.
Fig. 2.4 shows an illustrative example of an RDB. In the example RDB, there are three relations: `Vehicle`, `Region`, and `VehicleLocation`. We could imagine different situations, each with different vehicles, regions, etc. Each particular situation, like the one depicted in Fig. 2.4, corresponds to an instance of this relational model. The instance is represented as a table for each of the relations as shown Fig. 2.4, where the columns represent attributes of the relation and the rows represent the attribute values for specific entities. For example, the `Vehicle` relation has two attributes: `VehicleID`, which uniquely identifies each individual vehicle, and `VehicleType`, which indicates whether the vehicle is tracked or wheeled. The `VehicleLocation` relation has three attributes: `LocatingVehicleID`, `LocatingTimeID`, and `Location`. The `LocatingVehicleID` attribute in the `VehicleLocation` relation is a foreign key pointing to the primary key of the `Vehicle`
relation. A row of the \textit{VehicleLocation} relation represents a vehicle being located in a region at a point in time. Attributes that are part of the primary key of the relation (e.g., \textit{LocatingVehicleID} and \textit{LocatingTimeID} in the \textit{Location} relation) are denoted by bold, italicized, and underlined letters, while foreign keys which are not part of the primary key of a relation in which the foreign keys are used (e.g., \textit{Location} in the \textit{VehicleLocation} relation) are denoted by underlined letters.

\textbf{Definition 2.6 (Key)} A \textit{key} of a relation schema is a set of one or more attributes that uniquely identify a row of the relation.

\textbf{Definition 2.7 (Foreign key)} A \textit{foreign key}, FK, of a relation schema, RS\[A_1, A_2, ..., A_n]\], is a subset of the attributes \(\{A_1, A_2, ..., A_n\}\) that uniquely identifies a row of another relation.

The \textit{VehicleLocation} relation of Fig. 2.4 has two foreign keys (i.e., \textit{LocatingVehicleID/Vehicle} and \textit{Location/Region}). Here, we use the “/” symbol followed by the relation name to indicate the relation to which the foreign key points, i.e., \textit{LocatingVehicleID} foreign key refers to the \textit{Vehicle} relation. A relation schema containing a foreign key is called a \textit{target relation schema} for the foreign key, while a relation schema which is referred by the foreign key is called a \textit{home relation schema}. For example, the \textit{VehicleLocation} relation schema is the target relation schema for the \textit{LocatingVehicleID} attribute, while the \textit{Vehicle} relation schema is the home relation schema of the attribute. If the \textit{target and home relation schema} are same, the foreign key and primary key are same. In this case, the foreign key is called a \textit{recursive foreign key}. 
Definition 2.8 (Primary key) A primary key, PK, of a relation schema \(RS[A_1, A_2, ..., A_n]\) is a selected subset of the attributes \(\{A_1, A_2, ..., A_n\}\) that uniquely identifies each tuple in the RS.

The \(VehicleLocation\) relation of Fig. 2.4 has a primary key composed of two attributes (i.e., \(LocatingVehicleID/Vehicle\) and \(LocatingTimeID\)). Each tuple of the \(VehicleLocation\) relation is uniquely identified by these two arguments. Note that the attribute \(Location/Region\) in the relation is not used as the primary key in the target relation, but it uniquely identifies each tuple in its home relation (i.e., the \(Region\) relation); therefore, it is a foreign key. This kind of key is called a Non-Primary Foreign Key.

Definition 2.9 (Non-Primary Foreign Key) A Non-Primary Foreign Key, NK, is a Foreign Key that is not used for a primary key in a target relation.

The \(ContainingRegion\) attribute of the \(Region\) relation of Fig. 2.4 is another example of a non-primary foreign key since the home relation schema of the \(ContainingRegion\) attribute is the \(Region\) relation and it is not used for the primary key of the target relation.

Definition 2.10 (Non-Foreign-Key Attribute) A Non-Foreign-Key Attribute, A, is an attribute which is not a foreign key.

For example, in Fig. 2.4, the \(VehicleType\) and \(TerrainType\) attribute are non-foreign-key attributes since they are not foreign keys.

The \(VehicleLocation\) relation represents the region in which an entity is located at a point in time. The relations and their attributes – that is, a set of empty tables – is called the schema for the database. A populated set of tables such as Fig. 2.4 is called an
instance of the schema. It is clear that many different instances of this schema are possible, each corresponding to a different situation with different vehicles, different regions, and different assignments of vehicles to regions.

**Definition 2.11 (Relation schema)** A relation schema, RS\[A_1:D_1, A_2:D_2, ..., A_n:D_n\], is a set of pairs $A_i:D_i$, where $A_1\neq...\neq A_n$ are attribute names, and $D_i$ is a set called the domain for attribute $i$.

For example, the relation schema of Vehicle in Fig. 2.4 is $[\text{VehicleID}:\{v1, v2, \ldots\}, \text{VehicleType}:\{\text{wheeled, tracked}\}]$, where VehicleID and VehicleType are attributes with domains $\{v1, v2, \ldots\}$ and $\{\text{wheeled, tracked}\}$, respectively. Note that we denote the domain of the attribute by inserting the colon, “:”, between the name of the attribute and the name of the domain. This ancillary information can be omitted for brevity (e.g., $[\text{VehicleID}, \text{VehicleType}]$). As another example, the relation schema of the VehicleLocation relation is VehicleLocation$[\text{LocatingVehicleID}:\{v1, v2, \ldots\}, \text{LocatingTimeID}:\{t1, t2, \ldots\}, \text{Location}:\{r1, r1_1, r1_2, \ldots\}]$, where LocatingVehicleID, LocatingTimeID, and Location are attributes, with domains $\{v1, v2, \ldots\}$, $\{t1, t2, \ldots\}$, and $\{r1, r1_1, r1_2, \ldots\}$, respectively.

**Definition 2.12 (Entity relation schema)** An entity relation schema, ERS, is a relation schema containing a primary key that is not a foreign key and that consists of exactly one attribute.

In Fig. 2.4, the Vehicle and Region relation schema are entity relation schemas. An entity relation schema represents a type of entity. The primary key is a field that holds an identifier that uniquely identifies an instance of the entity type.
Definition 2.13 (Relationship relation schema) A relationship relation schema, RRS, is a relation schema containing a primary key consisting of at least two attributes which are foreign keys pointing to entity relation schemas.

Therefore, a relationship relation represents a relationship between entities of more than two entity types. In Fig. 2.4, the VehicleLocation relation schema is a relationship relation schema, if we assume that there is a Time entity relation schema and the LocatingTimeID attribute points to the Time entity relation schema.

Definition 2.14 (Relation instance) A relation instance, RI of a relation schema, RS[A₁, A₂, ..., Aₙ], is specified by a table with n columns and m rows, \[{d_{11}, d_{21}, ..., d_{n1}}, ..., {d_{1m}, d_{2m}, ..., d_{nm}}\]}, where Aᵢ is an attribute of RS, dᵢⱼ ∈ Dom(Aᵢ) ³. The relation instance represents a set of m specific entities of the class represented by the relation.

For example, in the Vehicle relation, there are six rows (i.e., \{{v₁, wheeled}, {v₂, tracked}, {v₃, tracked}, {v₄, tracked}, {v₅, wheeled}, {v₆, tracked}}). The instance of the Vehicle relation refers to these six rows, or tuples.

Definition 2.15 (Relational database schema) A relational database schema, RDBS[RS₁, RS₂, ..., RSₙ], is a set RSᵢ of relation schemas.

For example, the table headers of Fig. 2.4 describe the relational database schema, RDBS[Vehicle, Region, VehicleLocation].

³ Dom(X) is the domain of the attribute X. This is the set of values that the attribute can take on.
**Definition 2.16 (Relational database)** A relational database, RDB[RI_1:RS_1, RI_2:RS_2, ..., RI_n:RS_n], is a set of pairs RI_i:RS_i, where RS_i denotes a relation schema and RI_i denotes an instance of RS_i.

For example, the tables of Fig. 2.4 describe the relational database RDB[{{v1, wheeled}...{v6, tracked}}:Vehicle, {{r1, off-road, null}...{r2_1_1, road, r2_1}}:Region, {{v1, t1, r1}...{v2, t3, r2_1 }}:VehicleLocation].

In the relational model, normalization is an operation performed on an RDB to make it more manageable by minimizing redundancy of elements and reducing dependency between attributes [Codd, 1970]. Several normal forms have been suggested such as First ~ Fifth normal form and Boyce–Codd Normal Form (BCNF) [Codd, 1970][Codd, 1972][Codd, 1974][Fagin, 1977][Fagin, 1979][Maier, 1983].

**2.5 Uncertainty Modeling Process for Semantic Technology (UMP-ST)**

Traditional ontologies [Smith, 2003] are limited to deterministic knowledge. Probabilistic Ontologies (POs) move beyond this limitation by incorporating formal probabilistic semantics. Probabilistic OWL (PR-OWL) [Costa, 2005] is a probabilistic ontology language that extends OWL with semantics based on Multi-Entity Bayesian Networks (MEBN), a Bayesian probabilistic logic [Laskey, 2008]. PR-OWL has been extended to PR-OWL 2 [Carvalho, 2011], which provides a tighter link between the deterministic and probabilistic aspects of the Ontologies. Developing probabilistic ontologies can be greatly facilitated by the use of a modeling framework such as the UMP-ST [Carvalho et al., 2016]. UMP-ST was applied for construction of PR-OWL 1 &
2 probabilistic ontologies. The UMP-ST process consists of four main disciplines: (1) Requirement, (2) Analysis & Design, (3) Implementation, and (4) Test.

(1) The Requirement discipline defines goals, queries, and evidence for a probabilistic ontology. The goals are objectives to be achieved by reasoning with the probabilistic ontology (e.g., identify a ground target). The queries are specific questions for which the answers help to achieve the objectives. For example, what is the type of the target? The evidence is information used to answer the query (e.g., history of the speed of the target). (2) The Analysis & Design discipline designs entities, attributes for the entities, relationships between the entities, and rules for attributes and relationships to represent uncertainty. These are associated with the goals, queries, and evidence in the Requirement discipline. For example, suppose that a vehicle entity has two attributes, type, and speed. Then an example of a rule might be that if the speed is low, the type is likely to be a tracked vehicle. (3) The Implementation discipline is a step to develop a probabilistic ontology from outputs developed in the Analysis & Design discipline. Entities, attributes, relationships, and rules are mapped to elements of the probabilistic ontology. For example, the attributes type and speed are mapped to random variables type and speed, respectively. The rule for the speed and type is converted to the joint probability for the random variables type and speed. (4) In the Test discipline, the probabilistic ontology developed in the previous step is evaluated to assess its correctness. The correctness can be measured by three approaches: (a) Elicitation Review, in which completeness of the probabilistic ontology addressing requirements are reviewed, (b) Importance Analysis, which is a form of sensitivity analysis that examines the strength of
influence of each random variable on other random variables, and (c) *Case-based Evaluation*, in which various scenarios are defined and used to examine the reasoning implications of the probabilistic ontology [Laskey & Mahoney, 2000].

### 2.6 Conclusion

In this chapter, we have introduced background information about SAW, BN, MEBN, RM, and UMP-ST. This background information will be used for the next chapters. In Chapter 3, a mapping model from RM to MEBN is presented. In Chapter 4, PSAW from SAW is defined. In Chapter 5, HMLP, a modification of UMP-ST, is introduced.
CHAPTER THREE: MEBN-RM MAPPING MODEL

To design a MEBN model effectively, we would like to be able to learn a MEBN theory from data. The Relational Model (RM) is the most popular database model, although recently non-relational databases, called NoSQL, are receiving increasing attention [Han et al., 2011]. In the era of Big Data, we may need a scalable and flexible database to manage the huge and various types of data. However, NoSQL is still immature, with theory, standards and tools lagging behind the relational model. In this research, we focus on the Relational Model, which is a very well-known and standardized database model, as a source data model to develop a MEBN theory. Future work will consider extensions to NoSQL data.

A Relational Database (RDB) uses RM to describe and organize data. Therefore, we consider the problem of learning a MEBN model from data in a relational database. In order to do this, we need a formal model of how data in an RM can be used to estimate parameters and structure of a MEBN model. This requires a formal mapping between RM and MEBN.

An RDB stores data in the form of multiple Relations. A relation is composed of a relation schema and a relation instance. The relation schema contains a set of attributes. Multi-Entity Bayesian Networks (MEBN) is a knowledge representation language based on Bayesian Networks (BN) and First-Order Logic (FOL) for representing and reasoning
about an uncertain and complex world. A MEBN model, called MTheory, consists of a set of MFrags. An MFragment is composed of Context nodes, Input nodes, Resident nodes, a fragment Graph, and a set of Local Distributions. In this research, we introduce the mapping model between a relational database and an MTheory. This mapping is called **MEBN-RM Mapping Model** (or MEBN-RM). MEBN-RM contains four levels of definitions of mapping from elements of a relational database to elements of an MTheory.

In the first level, we define a mapping from a special relation schema of a relational database to an entity in an MTheory. In the second level, we define a mapping from attributes of a relation schema of a relational database to resident nodes of an MFragment. In the third level, a mapping between a relation schema and an MFragment is defined. In the fourth level, a mapping between a relational database and an MTheory is defined. Also, MEBN-RM contains a **MEBN-RM mapping algorithm** which uses the four definitions to develop an MTheory from RM.

### 3.1 MEBN-RM

MEBN-RM is a mapping method that specifies how to convert elements of RM to elements of MEBN. MEBN-RM contains four levels of definitions of mapping from elements of a relational database to elements of an MTheory. In the first level, an entity mapping between an entity relation schema and an entity is defined. In the second level, a resident node mapping is defined. In the third level, a relation and MFragment mapping is defined. In the fourth level, a relational database schema and MTheory mapping is defined. Before discussing these mappings, some ingredients and assumptions are discussed in this section.
The following RDBS from the *Vehicle Identification* RDB in Chapter 2 is used for an illustrative example through Chapter 3.

RDBS 3.1 [Vehicle Identification]

1. VehicleIdentification [  
2.   Vehicle [VehicleID, VehicleType],  
3.       Region [RegionID, TerrainType, ContainingRegion/Region],  
4.      VehicleLocation [LocatingVehicleID/Vehicle, LocatingTimeID, Location/Region],  
5.     Follow [FollowingVehicleID/Vehicle, LeadingVehicleID/Vehicle]  
6. ]

In the example RDBS, there are four relations: *Vehicle*, *Region*, *VehicleLocation*, and *Follow*. The relation *Vehicle* has two attributes: *VehicleID* and *VehicleType*. The relation *Region* has three attributes: *RegionID*, *TerrainType*, and *ContainingRegion*. The relation *VehicleLocation* has three attributes: *LocatingVehicleID*, *LocatingTimeID*, and *Location*. The relation *Follow* has two attributes: *FollowingVehicleID* and *LeadingVehicleID/Vehicle*.

As we saw in Chapter 2, an attribute in a relation can be a **Primary key** (PK), **Non-Foreign-Key Attribute** (NF), or **Non-Primary Foreign Key** (NK). For example, in the relation *Region*, the attribute *RegionID* is PK, the attribute *TerrainType* is NF, and the attribute *ContainingRegion* is NK. Because each of these types of attribute plays a different role in MEBN-RM, we distinguish them from each other.

In Chapter 2, Definition 2.11 defined a relation schema as a set of pairs consisting of an attribute name and a domain (i.e., RS[A1:D1, A2:D2, ..., An:Dn], where Ai is the i-th attribute name and Di is the domain for the attribute i). The attributes [A1, A2, ..., An] can be grouped into three disjoint and exhaustive subsets: PK, NF, and NK, where PK is the
set of attributes in a primary key, NF is the set of non-foreign-key attributes, and NK is the set of attributes in a non-primary foreign key.

A variety of relations can be formed in accordance with the following restrictions on the attributes in these subsets: a PK in a relation can't be empty, however an NF, NK, or both in a relation can be empty. Therefore, a relation can be one of four types: (1) RS[PK] denotes a relation schema containing only a primary key, (2) RS[PK, NF] denotes a relation schema containing only a primary key and non-foreign-key attributes, (3) RS[PK, NK] denotes a relation schema containing only a primary key and non-primary foreign key attributes, and (4) RS[PK, NF, NK] denotes a relation schema containing a primary key, non-foreign-key attributes, and non-primary foreign key attributes. Therefore, we need to define the mapping between RM and MEBN for each of these four types of relation.

We start by assuming that all relations in the RDB are in at least first normal form [Date, 2012]. Therefore, no relation may contain multiple values in a row (and domain) of an attribute of the relation. In accord with the formalism of MEBN, we introduce a new kind of normalization for MEBN-RM, which we call *Entity-Relationship Normalization*.

An MTheory developed from an RDB represents entities. We would like to derive these entities from the RDB. We can do this by defining an entity type in MEBN for each entity relation. This entity type can then be referenced in another relation by using the primary key of the entity relation as a foreign key in a referring relation. For example, we
can identify an entity of type *Vehicle* corresponding to the *Vehicle* relation from Fig. 2.4, and use the primary key *VehicleID* to refer to a specific vehicle instance.

In order for this method to produce a clearly defined mapping, we must make sure that all entity types we wish to represent in the MEBN model are represented as entity relations. As an example of a problem that can occur if this practice is not followed, consider an example of a relationship relation that contains a primary key consisting of two attributes that are not foreign keys. For example, we might represent patrol assignments using a *PatrolAssignment* relation having attributes *PatrolDriver*, *PatrolNavigator*, *PatrolVehicle/Vehicle*, and *PatrolRegion/Region*. The latter two attributes, the vehicle used and the region patrolled, are foreign keys pointing to the *Vehicle* and *Region* relations, respectively. The first two refer to the driver and navigator. These refer to soldiers. If we used this two-attribute primary key to define an entity type, we might erroneously create two different types, when the intention was that both would be filled by an entity of type *Soldier*. To address this issue, we would create a *Soldier* relation with its own primary key, and redefine *PatrolDriver* and *PatrolNavigator* as foreign keys pointing to the *Soldier* relation (i.e., *PatrolDriver/Soldier*, *PatrolNavigator/Soldier*, *PatrolVehicle/Vehicle*, and *PatrolRegion/Region*).

To formalize this idea, we define *Entity-Relationship Normalization* to ensure that each entity instance is uniquely identified and to clarify which attributes in a relation correspond to entities in MEBN.
Definition 3.1 (Entity-Relationship Normalization) Entity-Relationship Normal Form

if either its primary key is a single attribute which is not a foreign key, or its primary key contains two or more attributes, all of which are foreign keys.

For example, the relation VehicleLocation contains a primary key consisting of the attributes LocatingVehicleID and LocatingTimeID. LocatingVehicleID is a foreign key, while LocatingTimeID is not. Therefore, in the Entity-Relationship normalization, a new relation for LocatingTimeID should be added (e.g., the relation Time) and the attribute LocatingTimeID should be changed to a foreign key pointing to the new relation Time. As a result of this transformation, there are three relations (Vehicle, Region, and Time) in which a primary key for each of them consists of a single attribute. These relations are used to identify entities in a MEBN model. Thus, there are three entities; Vehicle, Region, and Time.

As an another example, let us suppose that there is a relation VehicleSize [VehicleID/Vehicle, Size]. The relation VehicleSize has a primary key which is a single foreign key (i.e., VehicleID/Vehicle). The Entity-Relationship Normalization does not allow such a relation. In other words, the primary key should not be a foreign key. Therefore, the relation VehicleSize merges with the relation Vehicle. Thus, this results in the relation Vehicle [VehicleID, VehicleType, Size].

Note that MEBN-RM provides a conversion from a relation schema (RS) to a partial MFrag containing a set of context and resident nodes. Full conversion from a relation instance to a complete MFrag (i.e., context nodes, resident nodes, input nodes, a directed acyclic graph, and local distributions) requires augmenting MEBN-RM with
either a human modeler or a machine learning algorithm. Hence, an MFragment in the following sections means a partial MFragment.

### 3.2 Entity Mapping

In MEBN, an entity is a unique kind of thing which exists distinctly and independently, and can be instantiated as an object in the world. For example, from a person entity, various person instances can be defined (e.g., John and Mathew). In RM, an entity relation, a relation containing a primary key consisting of exactly one attribute, represents a kind of thing that exists uniquely and independently. In MEBN-RM, an entity relation or a non-foreign-key attribute can be mapped to an entity in MEBN as defined by the following.

**Definition 3.2 (ERS to Entity Mapping)** An *ERS to entity mapping* is a mapping in which an entity relation schema, ERS, in an RDBS in Entity-Relationship Normal Form is mapped to an entity, E, denoted by $ERS \mapsto E$.\(^4\)

For example, the entity relation schema (Definition 2.12) *Vehicle* can be mapped to an entity VEHICLE. In MEBN-RM, entities are written as strings of uppercase letters.

### 3.3 Resident Node Mapping

In MFrags, a resident node can be described as *Function* or *Predicate* of FOL. MEBN allows the modeler to specify a probability distribution for the truth-value of a predicate or the value of a function. Formulas are not probabilistic and are defined by built-in MFrags [Laskey, 2008]. In this section, we describe the correspondence between functions and predicates in FOL and relations in RM.

---

\(^4\) A $\mapsto$ B means A is mapped to B.
Table 3.1 Resident node types on MEBN-RM

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Predicate</td>
<td>Follow(followingvehicleid, leadingvehicleid)</td>
</tr>
<tr>
<td>2</td>
<td>Function</td>
<td>Vehicletype(vehicleid), Terraintype(regionid), Containingregion(regionid), Location(loctatingvehicleid, locatingtimeid)</td>
</tr>
</tbody>
</table>

Table 3.1 shows the two types of the resident node with examples from the RDBS VehicleIdentification. These are discussed in the next subsection.

### 3.3.1 Predicate

In FOL, a predicate represents a true/false statement about entities in the domain. It is expressed by a predicate symbol followed by a list of arguments. For example, Follow\((x, y)\) is a predicate that expresses whether a following vehicle indicated by the argument \(x\) is following a leading vehicle indicated by the argument \(y\).

In MEBN, this predicate corresponds to a Boolean RV with possible values true and false. In RM, we can express a predicate as a relation schema in which the attributes are arguments of the predicate, and the rows of the table represent the arguments for which the predicate is true [Date, 2012]. For example, the relation Follow\([FollowingVehicleID, LeadingVehicleID]\) can be mapped to a predicate, Follow\((followingvehicleid, leadingvehicleid)\). The arguments of this predicate are identical to the set of attributes of the relation to which the predicate refers. A predicate from a relation can map to only a true value, because RM doesn't provide a false value for the predicate. For example, suppose that there is a dataset for the relation Follow\((\{v1, v2\}, \{v2, v3\})\).
This dataset can be mapped to a set of propositions of the predicate \( \{ \text{Follow} (v1, v2) = \text{true}, \text{Follow} (v2, v3) = \text{true} \} \).

Table 3.2 defines the relationship between elements of RM and elements of MEBN for the predicate.

<table>
<thead>
<tr>
<th>RM</th>
<th>MEBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of relation</td>
<td>Name of Predicate</td>
</tr>
<tr>
<td>Key</td>
<td>Arguments for Predicate</td>
</tr>
<tr>
<td>Presence of a tuple</td>
<td>\textit{true} value</td>
</tr>
<tr>
<td>Absence of a tuple</td>
<td>\textit{false} value</td>
</tr>
</tbody>
</table>

The name of a relation is used for the name of the predicate corresponding to the relation. The attributes of the relation correspond to the arguments of the predicate in sequence. A given tuple can be either present or absent in the RDB. If the tuple is present, a true value for the corresponding predicate can be assigned in the MEBN representation. If the tuple is absent, a false value for the corresponding predicate can be assigned in the MEBN representation.\(^5\) Now, we introduce a predicate resident node mapping.

**Definition 3.3 (Predicate resident node Mapping)** A *predicate resident node mapping* is a mapping in which a primary key, \( \mathbf{PK} \), of a relationship relation schema, \( \text{RRS}[\mathbf{PK}] \), is

---

\(^5\) This convention is used when adopting the closed world assumption, which asserts that all positive cases of a relation are represented in the database, so that absence of an instance implies the corresponding predicate is false. Dropping the closed world assumption could be handled by adding a *TruthValue* attribute, with values *True* and *False*. With this representation, all cases not appearing in the database would have unknown truth-value.
mapped to a resident node, \( R \), denoted by \( \text{RRS}[PK] \mapsto R[\text{RRS}(A_1, A_2, ..., A_n)] = \{\text{true, false}\} \), where \( PK = \{K_1, K_2, ..., K_n\} \) and \( K_i \mapsto A_i \).

For example, the relation schema, \( \text{Follow} \) \([\text{FollowingVehicleID, LeadingVehicleID}] \) can be mapped to a predicate resident node denoted by \( R[\text{Follow}(\text{followingvehicleid, leadingvehicleid})] \). In Section 5.2.3.4.3, we discuss how to get true/false values from relationship relations. Briefly, existing cases in relationship relations are used for true values, while non-existing cases in relationship relations are used for false values under closed-world assumption (CWA). CWA means that if a case does not appear in the database, it is assumed to be false. The predicate node mapping is discussed in Section 5.2.2.1 in detail.

### 3.3.2 Function

In FOL, a function is a mapping from domain entities called inputs to a value called the output. For example, the function \( \text{VEHICLETYPE(vehicleid)} \) is a function that maps its argument to \textit{wheeled} if it is a wheeled vehicle and \textit{tracked} if it is a tracked vehicle. In RM, a function is represented by a Non-Foreign-Key Attribute (NF) or Non-Primary Foreign Key (NK) of a relation, because both functionally depend on a Primary Key (PK). Thus, a function of a relation maps to its argument \( (s) \), the primary key\( (s) \) for the relation, to the output, which is the value of the domain of the attribute in the relation.

Table 3.3 defines the relationship between elements of RM and elements of MEBN for function. We define a mapping between an element of A or NK of RM, and a function of a resident node of MEBN formally.
Table 3.3 Function mapping in MEBN-RM

<table>
<thead>
<tr>
<th>NF or NK of RM</th>
<th>Resident Node of MEBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Foreign-Key Attribute/Non-Primary Foreign Key</td>
<td>Function</td>
</tr>
<tr>
<td>Primary Key</td>
<td>Arguments of Function</td>
</tr>
<tr>
<td>Domain of Attribute</td>
<td>Domain of Function</td>
</tr>
</tbody>
</table>

**Definition 3.4 (Function resident node Mapping)** A *function resident node mapping* is a mapping in which an attribute, $A$, of a relation schema, $RS$, and a primary key, $PK$, of the $RS$ is mapped to a resident node, $R$, of an MFrag, denoted by $RS[PK, A] \mapsto R[A(K_1, K_2, ..., K_n)] = \text{Dom}(A)$, where $PK = \{K_1, K_2, ..., K_n\}$.

For example, the argument of the function $\text{VEHICLETYPE}(vehicleid)$ is the primary key of the relation $Vehicle$, and the output is the value (either tracked or wheeled) of the attribute $\text{VehicleType}$. In other words, $\text{Vehicle}[\text{VehicleID}, \text{VehicleType}] \mapsto R[\text{VEHICLETYPE}(vehicleid)] = \text{Dom}(\text{VehicleType})$.

**3.4 Relation Schema and MFrag Mapping**

In the previous section, we discussed the mapping between the elements of the relation schema and the elements of the MFrag. In this section, we discuss the mapping between a relation schema and a partial MFrag. It is called *RS-MFrag Mapping*. For ERS and RRS, we define the RS-MFrag mapping formally.

**Definition 3.5 (RS-MFrag Mapping)** An *RS-MFrag Mapping* is a mapping in which a relation schema, $RS$ is mapped to a partial MFrag, $F$, denoted by $RS[PK, O] \mapsto F[C, R]$. Here, $C$ denotes a set of context nodes (Definition 2.2), $R$ denotes a set of resident nodes.
(Definition 2.2), \( \mathbf{PK} = \{K_1, K_2, \ldots, K_n\} \) is the primary key, and \( \mathbf{O} = \{O_1, O_2, \ldots, O_m\} \), where \( O_i \) is the \( i \)-th NF or NK attribute. The mapping satisfies the following conditions:

a) If the RS is an ERS and \( |\mathbf{O}| > 0 \), the \( \mathbf{PK} \) and \( \mathbf{O} \) of the ERS are mapped to the \( C \) and \( R \) of the F, respectively. This is denoted by \( \text{ERS}[\mathbf{PK}, \mathbf{O}] \mapsto \text{F}[C_1[\text{IsA}(K_1, E(K_1))], R_1[O_1(K_1)], \ldots, R_m[O_m(K_1)]] \).

b) If the RS is an RRS and \( |\mathbf{O}| > 0 \), the \( \mathbf{PK} \) and \( \mathbf{O} \) of the RRS are mapped to the \( C \) and \( R \) of the F, respectively. This is denoted by \( \text{RRS}[\mathbf{PK}, \mathbf{O}] \mapsto \text{F}[C_1[\text{IsA}(K_1, E(K_1))], \ldots, C_n[\text{IsA}(K_n, E(K_n))], R_1[O_1(\mathbf{PK})], \ldots, R_m[O_m(\mathbf{PK})]] \).

c) If the RS is an RRS and \( |\mathbf{O}| = 0 \), the \( \mathbf{PK} \) and RRS are mapped to the \( C \) and \( R \) of the F, respectively. This is denoted by \( \text{RRS}[\mathbf{PK}] \mapsto \text{F}[C_1[\text{IsA}(K_1, E(K_1))], \ldots, C_n[\text{IsA}(K_n, E(K_n))], R_1[\text{RRS}(\mathbf{PK})]] \).

Case (a) is that the RS is an ERS and it has at least one attribute which is not used for the primary key. In this case, the single attribute \( K_1 \) in \( \mathbf{PK} \) is used to create the IsA context node, and each attribute in \( \mathbf{O} \) is mapped to each resident node respectively using the function resident node mapping. For example, the relation \( \text{Vehicle[VehicleID/Vehicle, VehicleClass]} \) becomes a partial MFrag denoted by 

\[ \text{F}[C[\text{IsA(vehicleid, VEHICLE)}], R[\text{VehicleClass(vehicleid)}]] \]

Case (b) is that the RS is an RRS and it has an attribute which is not used for the primary key. Each attribute \( K_i \) in \( \mathbf{PK} \) is used to create its respective IsA context node and each attribute in \( \mathbf{O} \) is mapped to a resident node using the function resident node mapping. For example, the relation \( \text{VehicleLocation[LocatingVehicleID/Vehicle,} \)

---

\(^6\) \( E(X) \) is the entity type which the attribute \( X \) points to.
LocatingTimeID/Time, Location/Region] becomes a partial MFrag denoted by 
F[C[IsA(locatingvehicleid, VEHICLE), IsA(locatingtimeid, TIME), 
R[Location(locatingvehicleid, locatingtimeid)]].

Case (c) is that the RS is the RRS and has no attributes other than the primary key. In this case, each attribute $K_i$ in $PK$ is used to create the IsA context node and the relation is mapped to a predicate resident node, $RRS(K_1, K_2, \ldots, K_n)$, using the Predicate resident node Mapping. For example, the relation Follow [FollowingVehicleID/Vehicle, 
LeadingVehicleID/Vehicle] becomes a partial MFrag denoted by 
F[C[IsA(followingvehicleid, VEHICLE), IsA(leadingvehicleid, VEHICLE), 
R[Follow(followingvehicleid, leadingvehicleid)]].

3.5 Relational Database Schema and MTheory Mapping

In the previous section, we discussed the mapping between a relation schema and partial MFrag. In this section, we discuss the mapping between a relational database schema (RDBS) and MTheory (M). It is called RDBS-MTheory Mapping. Basically, the mapping produces one MTheory from one relational database schema, denoted by the following.

Definition 3.6 (RDBS-MTheory Mapping) An RDBS-MTheory Mapping is a mapping in which a relational database schema RDBS is mapped to an MTheory M (Definition 2.4), denoted by RDBS[RS_1, RS_2, \ldots, RS_n] \mapsto M[F_1, F_2, \ldots, F_n], where RS_i is a relation schema in the RDBS, F_i is a partial MFrag in the M, and n is the number of the relation schemas in the RDBS and the number of the partial MFrags in the M, if the RS-MFrag Mapping between RS_i and F_i is able to be used.
For example, the *Vehicle Identification* RDBS can be directly an MTheory using the RDBS-MTheory mapping. The relations *Vehicle*, *Region*, and *VehicleLocation* in the RDBS are converted to partial MFrags *Vehicle*, *Region*, and *VehicleLocation*. The following section presents a mapping algorithm using MEBN-RM, which is a process to develop an MTheory from data in RM and contains specific sub-steps.

### 3.6 MEBN-RM Mapping Algorithm

In the previous sections, we discussed the mapping definitions for entities, resident nodes, MFrags, and MTheories. This section presents a *MEBN-RM mapping algorithm* which performs the *RDBS-MTheory Mapping* in Definition 3.6 and specifies how to convert an MTheory from a relational database schema using the MEBN-RM definitions.

---

**Algorithm 3.1: MEBN-RM Mapping**

**Procedure** `MEBN-RM_Mapping ( RDBS // relational database schema )`

1. $M$ ← create a default MTheory
2. $M$.name ← get a schema name using `RDBS`
3. for $i = 1, \ldots$ until $n$
4.   $rs_i$ ← get an $i$-th relation schema from `RDBS`
5.   `RM-MFrag_Mapping(rs_i, M)`
6. return $M$

**Procedure** `RM-MFrag_Mapping ( rs, // relation schema M // default MTheory )`

7. if $rs = ERS$ then
8.   $M.E$ ← create an entity type from $rs$ using `ERS to Entity Mapping`
9. if $rs = ERS \land |rs.O| > 0$ then
10.  $M.F$ ← $F$ ← create an MFragment for $rs$
11.  $F.C$ ← create IsA context nodes from the entity types $M.E$ associated with $M.F$
12.  $F.R$ ← create resident nodes from $rs$ using `Function resident node Mapping`
13. else if $rs = RRS$ then
14.  $M.F$ ← $F$ ← create an MFragment for $rs$
For the MEBN-RM mapping algorithm, we assume that (1) the relational database schema are normalized by the Entity-Relationship Normalization (Definition 3.1), and (2) the list of relation schemas in the relational database schema are sorted by the entity relation schemas (ERS) first and the relationship relation schemas (RRS) second. For the algorithm, let $M$ be an MTheory, $M.E$ be a set of entity types of $M$, $M.F$ be a set of MFrags, $F.C$ be a set of context nodes in an MFragment $F$, $F.R$ be a set of resident nodes of $F$, $RDBS$ be a relational database schema, $rs$ be a relation schema in $RDBS$, and $rs.O$ be a set of attributes for NF and/or NK of $rs$.

Inputs of this algorithm are a relational database schema $RDBS$. (1) The algorithm starts with creating a default MTheory $M$. (2) The name of $RDBS$ is used to create the name of $M$. (3)(4) All relation schema are investigated from a first relation schema $rs_1$ to a last relation schema $rs_n$. (5) For an $i$-th relation schema, the algorithm performs the procedure $RM$-$MFragment Mapping$ defined in Definition 3.5. (7) If the $i$-th relation schema is ERS, (8) the $ERS$ to $Entity Mapping$ (Definition 3.2) is performed. (9) If the $i$-th relation schema $rs$ is ERS and there is an attribute O for NF or NK, then (10) an MFragment $F$ for the $rs$ is created and added to the set of MFrags of $M$, (11) Isa context nodes are created from the entity types $M.E$ associated with $F$ and added to the set of context nodes of $F$, and (12) resident nodes are created from $rs$ using the $Function resident node Mapping$.
Mapping (Definition 3.4) and added into the set of resident nodes of $F$. (13) If the $i$-th relation schema $rs$ is RRS, then performs (14) to (19). (14) An MFragment $F$ for the $rs$ is created and added to the set of MFrags of $M$. (15) Isa context nodes are created from the entity types $M.E$ associated with $F$ and added to the set of context nodes of $F$. (16) If there is no attribute $O$ for NF or NK, (17) the *Predicate resident node Mapping* (Definition 3.3) for $rs$ is performed. (18) If there is an attribute $O$ for NF or NK, (19) the *Function resident node Mapping* (Definition 3.4) for $rs$ is performed. (6) The algorithm results in the MTheory $M$.

The following MTheory shows a result for the RDBS 3.1 using the MEBN-RM Mapping algorithm.

---

### MTheory 3.1: Vehicle Identification

1. [F: Vehicle]
   
   2. [C: Isa (vehicleid, VEHICLE)]
   
   3. [R: VehicleType (vehicleid)]

5. [F: Region]
   
   6. [C: Isa (regionid, REGION)]
   
   7. [R: TerrainType (regionid)]

9. [R: ContainingRegion (regionid)]

10. [F: VehicleLocation]

11. [C: Isa (locatingvehicleid, VEHICLE)]

12. [C: Isa (locatingtimeid, TIME)]

13. [R: Location (locatingvehicleid, locatingtimeid)]

15. [F: Follow]

16. [C: Isa (followingvehicleid, VEHICLE)]

17. [C: Isa (leadingvehicleid, VEHICLE)]

18. [R: Follow (followingvehicleid, leadingvehicleid)]

19. [ ]
For example, the relation Region in RDBS 3.1 is an ERS, which is mapped to the entity REGION and used to create the Isa context node in Line 6. The relation Region contains an attribute TerrainType which is converted to a resident node Terraintype as a Function in Line 7. Also, the relation Region contains a Non-Primary Foreign Key ContainingRegion which is mapped to a resident node Containingregion as a Function in Line 8.

### 3.7 Conclusion

In this research, we presented MEBN-RM formalizing conversion from a relational database schema in RM to an MTheory in MEBN syntactically. To do this, MEBN-RM contained the four levels of the mappings between elements of the relational database schema and MTheory. Table 3.4 summarizes the mappings which this research presents.

<table>
<thead>
<tr>
<th>RM</th>
<th>Mapping Types</th>
<th>MEBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERS</td>
<td>Definition 3.2 ERS to Entity Mapping</td>
<td>Entity</td>
</tr>
<tr>
<td>RRS</td>
<td>Definition 3.3 Predicate resident node Mapping</td>
<td>Predicate resident node</td>
</tr>
<tr>
<td>Non-foreign-key attribute, non-primary foreign key</td>
<td>Definition 3.4 Function resident node Mapping</td>
<td>Function resident node</td>
</tr>
<tr>
<td>RS</td>
<td>Definition 3.5 RS-MFrag Mapping</td>
<td>MFragment</td>
</tr>
<tr>
<td>RDBS</td>
<td>Definition 3.6 RDBS-MTheory Mapping</td>
<td>MTheory</td>
</tr>
</tbody>
</table>
MEBN-RM is a foundation of designing a MEBN model from a relational database, so, using MEBN-RM, the modeler (Human or Machine) can design the MEBN model seamlessly. The idea behind MEBN-RM may be used to develop other mapping models for different types of database (e.g., ontology, graph, and event database) as an example mapping model.
CHAPTER FOUR: PSAW-MEBN REFERENCE MODEL

PSAW requires reasoning about multiple sensors and targets over time. Furthermore, the number of entities and the relationships among them may be uncertain. For this reason, PSAW requires an expressive language for representing and reasoning about situations. Multi-Entity Bayesian Networks (MEBN) [Laskey, 2008] is such an expressive language, and has been applied to PSAW systems [Laskey, 2000; Wright et al., 2002; Costa et al., 2005; Costa et al., 2009; Carvalho et al., 2010; Costa et al., 2012; Suzic, 2005].

Guidance for modelers is available in the form of a reference architecture [Haberlin et al., 2013] and a structured process [Carvalho, 2011] for constructing semantically rich models for reasoning under uncertainty in complex environments. Nevertheless, the process of constructing a MEBN model for a new application remains challenging. Different applications of MEBN to PSAW tend to have similar goals and common model elements. Therefore, even guided by the reference architecture and the structured modeling process, designing a new PSAW-MTheory from the ground up is inefficient. Incorporating the common model elements into a reference model, “an abstract framework for understanding significant relationships among the entities of some environment [MacKenzie et al., 2006]”, promises to significantly reduce the development time and cost, and also result in a more well-formed model.
A generic framework for plan recognition using MEBN has been introduced [Suzic, 2005]. The framework focuses on estimating plans of multi-agent organizations through time using an ontology that supports MEBN and contains four main template random variables (i.e., utility, plan, state, and observation) to represent a situation for agents in the multi-agent organizations. In the framework, an agent aims to optimize its utility (e.g., gain and loss). Also, the agents may have plans for various situations to fulfill the objectives of the agents. According to the plans, there are several states for the agents and the states may be observed. However, the Suzic framework does not provide specific elements to match the elements of MEBN (e.g., MFrags, RVs, LPD function, and entity). In both manual MEBN modeling and automatic MEBN learning, it is necessary to know what kinds of MFrags, RVs, LPD functions, and entities are needed for PSAW. Thus, pre-defined candidates for these model elements can make the model building process more efficient.

This chapter defines a PSAW-MEBN reference model. The model specifies reference MFrags, RVs, and entities which support the design of a MEBN model for PSAW. Such a MEBN model is called a PSAW-MEBN model. The PSAW-MEBN model is designed to reason about PSAW questions (see Section 4.1) and can be supported by the reference model, called a PSAW-MEBN reference model (see Section 4.2). We introduce what PSAW is (in Section 4.1.1), analyze what properties of PSAW are (in Section 4.1.2 and Section 4.1.3), discuss what OODA is (in Section 4.1.4), and present how to interpret PSAW in terms of OODA (in Section 4.1.5) to investigate possible elements of PSAW from OODA. In Section 4.2, the PSAW-MEBN reference
model is presented with the reference entities (in Section 4.2.1), the reference RVs (in Section 4.2.2), and the reference MFrags (in Section 4.2.3).

4.1 Predictive Situation Awareness

4.1.1 Definition of PSAW

There are several definitions of a situation in the literature. Merriam-Webster Dictionary [Merriam-Webster's situation] defines it as "the way in which something is placed in relation to its surroundings". Wordnet Dictionary [Wordnet's situation] defines it as "the general state of things; the combination of circumstances at a given time;". Devlin [1995] defined a situation as a structured part of reality which is individuated by an agent. Sowa [2011] defined a situation as "a region of space-time that bounds the range of immediate perception, action, interaction, and communication of one or more agents". Devlin [2006] discussed the definition from Barwise & Perry [1981] in which a situation consists of "objects having properties and standing in relations to one another [Barwise & Perry, 1981]". A situation can be described as relations of objects existing in a certain circumstance at a given time, reflecting a partial world, and can be perceived by an agent.

Fig. 4.1 shows an illustrative example for Situation awareness (SAW). A person observes the target situation in the world, in which two vehicles are moving. In the target situation, the first vehicle is actually a wheeled vehicle, while the second vehicle is actually a tracked vehicle. The person makes a picture of the interpreted situation in the person's mind, in which two wheeled vehicles are moving. The interpreted situation in the person's awareness and the target situation in the world are different, because of various
internal and external factors (e.g., the person's knowledge about the vehicle, the person's eyesight, and the observing environment between the person and the vehicles). An overall situation containing the person, the environment of the person, the interpreted situation, and the target situation can be called a \textit{perceived situation}.

![Diagram of situation awareness](image)

\textbf{Figure 4.1 An illustrative example of situation awareness}

According to Devlin [1995], a human agent \textit{individuates} (picks out, or recognizes) objects within a situation and recognizes properties of and relationships among these objects. Accordingly, situation awareness occurs when \textit{an observer recognizes properties of and relations among the interpreted objects through a process of observing a situation in the world}. The observer is a subject in the perceived situation, who observes the situation, or the parts of the situation that can be observed, using sensors of the observer. The observer interprets the observed aspects of the situation to produce an interpreted situation. Fig. 4.1 shows a perceived situation in which the eye in
the figure is a sensor of an observer, the two small vehicles in the target situation are the observed, and the two wheeled vehicles (the left two vehicles in the figure) in the interpreted situation for the observer are the interpreted (or reported) objects.

Predictive Situation Awareness (PSAW) emphasizes the ability to make predictions about aspects of a temporally evolving situation. Decision makers have performed PSAW using higher-level information fusion in which they use the results of low-level fusion to estimate and predict the evolving situation over time. For performing PSAW, a model representing an evolving situation is necessary. In the military domain, a predictive model is defined as the following.

“Predictive model. A model in which the values of future states can be predicted or are hypothesized; for example, a model that predicts weather patterns based on the current value of temperature, humidity, wind speed, and so on at various locations [DoD 5000.59-M, 2011]”.

A predictive model for PSAW (or a PSAW model) should be flexible enough to represent a variety of complex situations to capture attributes of, relationships among, and processes associated with various kinds of entities in a situation. Also, the PSAW model requires a representation treating uncertainty.

“Military situations are inherently uncertain, and the available data are inevitably noisy and incomplete. It is essential to be able to represent and reason with uncertainty [Costa et al., 2009]”.

To identify the PSAW model more precisely, we use the following as our working definition of PSAW.
**Definition 4.1 (Predictive Situation Awareness)** Predictive Situation Awareness (PSAW) is the ability to estimate and predict a possible situation involving multiple actors \((Atr)\) and/or objects \((Obj)\) in different locations \((Loc)\), in which actors may trigger events \((Evt)\) or activities \((Act)\) occurring over time \((T)\), and where the meaning of the situation \((MS)\) is revealed by integrating previous knowledge with evidence from multiple sources.

Actors \((Atr)\) are entities which can generate events \((Evt)\) or activities \((Act)\), whereas objects \((Obj)\) can be involved in events or activities but cannot actively initiate them. For example, vehicles \(V1\) and \(V2\) can be modeled as actors, and they may be able to perform \(Move, Defend,\) or \(Attack\) activities. (In a richer representation, we might model the vehicles as objects and their drivers as actors.) On the other hand, a region or stone is an object and cannot perform an activity. Events and activities happen at a given time or during a time interval \((T)\). For example, two vehicles \(V1\) and \(V2\) may perform certain activities at time \(T1\). The meaning of the situation \((MS)\) means why those activities stimulated by the actors are occurring.

The elements (e.g., actor, event, and activity) in PSAW have not been defined universally and various definitions exist. In the followings, we define each element of PSAW used for this research.

**Definition 4.2 (Actor)** An actor \((Atr)\) is an entity which can generate events or activities.

**Definition 4.3 (Event)** An event \((Evt)\) is something occurring at a given time.

**Definition 4.4 (Activity)** An activity \((Act)\) is the something that an actor does during a time interval.
**Definition 4.5 (Meaning of the Situation)** The meaning of a situation (MS) is an interpretation or an explanation by an observer of a situation.

**Definition 4.6 (Observer)** An observer (OR) is an actor who can be aware of objects through certain senses, interpret the observations, and produce a meaning of a situation (MS).

**Definition 4.7 (Sensor)** A sensor (SR) is an object which transmits an input event to output event for a certain purpose.

**Definition 4.8 (Target)** A target (TR) is something which exists in spatial and temporal spaces and is observed in a target situation by observers.

**Definition 4.9 ( Reported Target)** A reported (or interpreted) target (RT) is something which is transformed from an observed object by observers and exists in an interpreted situation.

**Definition 4.10 (Observing Condition)** An observing condition (OC) is a condition which affects an observing process performed by an observer.

For example, vehicles $V1$ and $V2$ can be modeled as actors, and they may be able to perform *Move*, *Defend*, or *Attack* activity, and generate various events at time $T1$. If the two vehicles’ activities are approaching to a certain target, the meaning of the situation (MS) for the two vehicles may be an attack situation which is interpreted by an observer. The meaning of the situation is identical with Devlin's [1995] type of the situation.

The observing process operates in various ways according to the context and type of the observer. The observing condition can be classified into two categories (Internal
Condition and External Condition). The internal condition is associated with the observer (e.g., the observer's observing capability). The external condition concerns the relationship between the observer and the observed objects (e.g., range and weather).

**Definition 4.11 (Target Situation)** The target situation (TS) is a situation which contains the relations of the observed objects and is observed by an observer.

**Definition 4.12 (Interpreted Situation)** The interpreted (IS) situation is a situation which contains the relations of the interpreted objects and is interpreted by an observer.

To perform PSAW, we may want to answer various questions. (e.g., "How many military vehicles are going to encounter?", "How high is the current level of danger to the Enterprise?", "Where is the target located?", and "What is the enemy doing and why?" [Waltz & Buede, 1986][Costa, 2005][Dorion et al., 2008]). In our research, questions such as the ones listed above are called **PSAW questions**. The purpose of PSAW is to answer some or all of these PSAW questions. Note that Appendix B presents a list of PSAW Questions.

### 4.1.2 Properties of a PSAW Model

In this section, we introduce some properties of a PSAW model which will be used for constructing a MEBN model for PSAW, called a **PSAW-MTheory**, addressing the PSAW questions in Section 4.1.1. Generally, when a model is constructed, the following properties for the model can be considered [Sterman, 1991]: (1) A model boundary, (2) exogenous variables, (3) endogenous variables, (4) a time horizon, and (5) a time granularity. The model boundary identifies the exogenous variables and the endogenous variables from the point of view of the user of the model. If the model is a
dynamic model in which some states of the model change over time, the time horizon (i.e., the time period the model is representing) and time granularity (i.e., the length of time step) should be identified. These are key modeling properties, which can be used for developing the properties of the PSAW model.

### 4.1.2.1 Time and Space

Time and space are basic elements for PSAW. Time can be considered as discrete or continuous. If time is modeled as discrete, time can be represented as a time stamp (e.g., \( t_1, t_2, \) and \( t_3 \)). If time is modeled as continuous, time can be represented as a continuous value with units (e.g., 5.132 seconds). Dynamic Bayesian networks (DBNs) [Murphy, 1998] follow the discrete time model, while Continuous Time Bayesian Networks [Nodelman et al., 2002] use the continuous time model. The continuous time model provides more precise estimation and prediction, however computation of the continuous time model is difficult. In this research, the PSAW-MTheory is based on the discrete time model.

A discrete time in the discrete time model is a time stamp or time slice. A time stamp consists of a starting time and a time period. Time stamps can be equally spaced, which means that the time periods for all time stamps are equal. If time stamps are not equally spaced, then the time periods for all \( t_i \) should be described in some way. This approach makes it more complex to treat each time period. Identifying the time period is important and it influences the PSAW model. For example, if a time period for a model is modeled as a long-term (i.e., low precision), it may be difficult to represent events occurring in the short-term with any precision. The time period can be chosen by

72
consulting historical data, expert opinions, and/or sensor capability. In this research, we assume a single and fixed time period (e.g., a year, a month, a week, and a day) and leave the problem of a multi-resolution temporal model, in which varying time periods can be associated with each other, for future work [Bettini et al., 1998][Bettini et al., 1998 May]. If there are two more time stamps, various relationships between the time stamps can be held. Allen [1983] proposed thirteen relations; equality($t_i$, $t_j$), six relationships (precedes($t_i$, $t_j$), meets($t_i$, $t_j$), overlaps($t_i$, $t_j$), starts($t_i$, $t_j$), finishes($t_i$, $t_j$), during($t_i$, $t_j$)), and other six relationships with reversed arguments from the first six relationships.

Space can be represented as an $n$-dimensional Euclidean space. A region in space can be a part of Euclidean space. In PSAW, a situation is associated with regions. An object in the situation occupies its own region contained in an overall region of the situation. Randell et al., [1992] defined Region Connection Calculus (RCC) which specified several connections between regions. RCC contains 8 types of connection: (1) DC($x$, $y$) $x$ is disconnected from $y$, (2) EC($x$, $y$) $x$ is externally connected to $y$, (3) PO($x$, $y$) $x$ partially overlaps $y$, (4) EQ($x$, $y$) $x$ is identical with $y$, (5) TPP($x$, $y$) $x$ is a tangential proper part of $y$, (6) NTPP($x$, $y$) $x$ is a non-tangential proper part of $y$, (7) TPPi($x$, $y$) $x$ is a tangential proper part inverse of $y$, and (8) NTPPi($x$, $y$) $x$ is a non-tangential proper part inverse of $y$ [Randell et al., 1992]. Using RCC, objects in a situation can be represented more clearly and spatial relationships between objects can be reasoned.

Another issue about space and time is whether they are a composite entity or they are mutually incompatible entities. Grenon & Smith [2004] proposed a separation between space and time as named continuant and occurrent, respectively. A continuant
continuously exists over time (e.g., objects) and an occurrent changes continuant things over time (e.g., event, processes, and activities). These can be combined into a composite entity (e.g., an object whose shape changes over time).

### 4.1.2.2 Object, Actor, and Non-Actor

An object has at least one attribute (e.g., type, size, shape). An object can influence another object (e.g., rain causes the target’s speed to be slow) or be related to another object (e.g., the target is behind the tree). Relationships can be static or may change over time (i.e., dynamic relationships). A PSAW model should be able to represent objects, attributes, and static or dynamic relationships (e.g., the target behind the tree at time 1 moved from the tree at time 2).

An object can be classified into two categories (Actor and Non-actor). An actor can engage in an activity or initiate an event. Once the activity or event happens, the actor may have a goal, purpose, mission, plan, preference, and/or belief. These mental attributes of the actor should be considered in the PSAW model. We will discuss this in Section 4.1.3.

### 4.1.2.3 Groups of Objects

In PSAW, a situation is about more than just individual objects. Thus, the situation can contain a group of individual objects. The group can be represented by relationships of the individual objects. For example, countries $A$ and $B$ may be grouped because they are in an alliance. Many objects can be grouped according to a certain relationship. The relationship combining objects is determined by an observer’s perspective. An example is a road on which several types of cars are running. These cars
can be grouped by the direction, the speed, or the type. This relationship depends on the intentions of the observer and isn’t constrained by space. For example, one person in Asia and another person in Europe can be grouped, if they communicate with each other and/or they work for the same organization. Once some objects are grouped, a property of the group can be observed (e.g., the group’s direction or speed). As the single object case, these properties of a group can change over time. Also, the objects in the group can stay or leave the group over time. Furthermore, some groups can be grouped at a higher level. For example, a military company (higher level group) can consist of three platoons (sub-groups).

4.1.2.4 The Observer, the Sensor, and the Observed

The observer observes the observed through sensors at a certain time. Observations may be intermittent because of observing problems, capabilities of the observer, or concealment by the observed. For this reason, a tracking method estimating an actual property of the observed is necessary. Furthermore, to understand some activities or properties of the observed, the intention or mission of the observed should be identified, so that the observer can estimate the current states of the observed more accurately as well as predict the future states of the observed.

According to the capability of an observer, the observation quality varies. The quality can be regarded as the performance of the observer. This performance should be represented in a PSAW model. At some times, several observers may observe one target and the observers’ performances may be different, so the PSAW model should have a means of increasing estimation quality by combining multiple observations. In some
cases, one observer can detect several objects and these objects should be distinguished and grouped.

4.1.2.5 Types of Uncertainty

Uncertainty about a situation can be classified into five categories [Wright et al., 2001][Laskey et al., 2001][D’Ambrosio et al, 2001]: (1) Existence uncertainty, (2) Type uncertainty, (3) Reference uncertainty, (4) Attribute uncertainty, and (5) Relationship uncertainty. Existence uncertainty is uncertainty about whether an object exists or not. For example, an observation for an object may indicate the actual existence of the object or can be a false alarm. Type uncertainty means uncertainty about what type of object it is. For example, it may be uncertain whether a target is a vehicle or a human. If an object has an attribute that is filled by another entity, then reference uncertainty is uncertainty about which entity fills the slot. An example of reference uncertainty is uncertainty about which object a sensor is looking at. Attribute uncertainty is uncertainty about properties of an object (e.g., the speed of the target). Relationship uncertainty is uncertainty about properties that relate two or more objects (e.g., whether two targets are communicating).

4.1.3 Physical and Mental Situations in PSAW

An observer can observe, interpret, estimate, and predict various situations associated with targets. Commonly, these situations are regarded as physical situations. However, these situations may not be the only physical situations. We can think of what situation of a target which an observer wants to see. For example, a target may act within a decision-making process to achieve certain goals or missions. The decision-making process may influence the actions of the target. The actions may result in some states in
the world. And the states may be observed by an observer observing the target. In view of
the observer, what the observer wants to see can be anything in the situation of the target
(e.g., the goals, the actions, and the states of the target). The situation of the target can be
considered in two categories: a physical and mental situation. If a target is an object (e.g.,
stone) which doesn’t have thinking ability, the physical situation (e.g., a target’s size or
movement) of the object can be considered alone. If a target is an actor (e.g., human)
which can think of something and have an intention, the physical situation of the actor as
well as the mental situation of the actor (e.g., a target’s emotion or intention) should be
dealt with.

For modeling the physical situation, various ontologies for situation have been
researched. For reusability of a situation model, a core-ontology for situation awareness
[Matheus et al., 2003] was suggested, so that it can be applied to designing various
situation models. It provides a schema containing objects, attributes of the objects, and
relationships among the objects to support SAW. Situation ontology for hierarchical
situation modeling was presented [Yau & Liu, 2006]. Situation ontology consists of
context layer for determining elements of context and situation layer for aggregation of
contexts or atomic situations. Situational context ontology designed for SAW focuses on
a person who is in a context and is involved in a situation [Anagnostopoulos et al., 2007].
The situational context ontology is represented in the Web Ontology Language (OWL-
DL) [Baader, 2003] providing useful relations for representation of SAW (e.g., spatial
relations, temporal relations, and restriction relations). SAW ontologies [Matheus et al.,
2005][Yau & Liu, 2006] were surveyed with an evaluation framework [Baumgartner &
Retschitzegger, 2006]. The evaluation framework in the survey measured the SAW ontologies in terms of recommended properties in top-level concept (i.e., object, attribute, relation/role, event, and situation) and specific concepts (i.e., space/time, thematic roles, situation type, and situation object) for SAW. These SAW ontologies can be used for modeling the physical situation.

For modeling a mental situation, we can think of Waltz’s [2003] adversary mental process containing the following steps: (1) Firstly, goals and values of the adversary are initiated by stimuli, (2) Motives are initialized from the goals by combining with some beliefs, (3) Intentions proceed from the motives, (4) Alternative plans are led by the intentions, (5) Chosen plans are decided from the alternative plans, and (6) Action commands following the chosen plans are made. And from the action commands resulted from the entire mental process, actual actions, and state changes in the real world influenced by the actions will appear in the physical domain. However, modeling a mental situation is more complicated than modeling a physical situation as [Waltz, 2003] pointed out that “real-world adversarial systems are complex and not so easily represented by rigid doctrinal and hierarchical models.”

4.1.4 Properties of OODA

For modeling a mental situation, it is necessary to consider what mental processes operate to make decisions by an actor and identify what types of outputs from the mental processes are. In the military domain, the decision-making process has been researched thoroughly, because of the significance of decision-making. Several decision-making models were developed with specific processes. For example, Boyd’s [1976, 1987]

Figure 4.2 OODA Loop

OODA is a high-level concept for decision making, so OODA can be a starting point to consider more specific processes (e.g., C2 model, HEAT, decision-ladder, SHOR, Military Decision-Making Process, and etc). For example, Grant & Kooter [2005] tried to reengineer OODA into a general C2 process by addressing some deficiencies of
OODA (e.g., planning process and learning process). They concluded that OODA could represent a general C2 process. In this research, we employ OODA (Fig. 4.2) as a general mental process, in which an actor may make a decision. According to the situation in a specific domain, the mental process in our research based on OODA may be replaced by the specific processes.

OODA contains four steps (Observe, Orient, Decide, and Act). Each of the four steps in OODA can have various inputs and outputs as shown Table 4.1. For example, in the Observe step, data or signal from every mental/physical situation (e.g., states, activities, and goals) of external systems (e.g., an adversary) as well as internal systems (e.g., a command center or an allied army) in the world are observed in some internal observing guidance or control, and observations derived from data or signal are produced. In the Orient step, observations become information, formed as a model, by reasoning, analysis, and synthesis influenced from knowledge, belief, condition, etc. The Orient step can produce plan and COA (Course Of Actions). Hypotheses or alternatives for models can be decided by the preference of a decision maker in the Decide step. In the Act step, all decided results are implemented, and real activities and states can be operated and produced, respectively. The four steps continue until the end of the life cycle of the actor.

Fig. 4.2 shows a situation in which only one actor operates. We can extend the situation into a multi-actor situation in which many actors interact with each other. The multi-actor situation can be represented as multiple interacting OODAs.
Fig. 4.3 shows an example of a two-actor situation with two-OODAs in which the observer observes all steps in OODA of the observed. In this research, we suppose that every step in OODA of the observed can be observed by an observer performing the Observe step in OODA. The Observe step of an actor $i$ at time $t$ can be defined as a set of four steps of another actor $j$, $\text{Observe}_{i,t} = \{\text{Observe}_{j,t}, \text{Orient}_{j,t}, \text{Decide}_{j,t}, \text{Act}_{j,t}\}$. 

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observe</td>
<td>World Data, Observing Condition, Internal Guidance/Control</td>
</tr>
<tr>
<td>Orient</td>
<td>Knowledge, Belief, Strategies, Experience, Orienting Condition, Observations</td>
</tr>
<tr>
<td>Decide</td>
<td>Vision, Mission, Core Values, Goals, Objectives, Intension, Deciding Condition, Alternative Plans, Alternative COAs, Alternative Responses, Hypotheses</td>
</tr>
<tr>
<td>Act</td>
<td>Internal Guidance/Control, Acting Condition, Chosen Plans, Chosen COAs, Chosen Responses, Chosen Hypotheses</td>
</tr>
</tbody>
</table>

**Figure 4.3 What an observer sees**
For example, the observer $A$ can see the plan and COA of the observed $B$ as well as the activities and status of the observed $B$ at time 1, $\text{Observe}_{A,1} = \{\text{Observe}_{B,1}, \text{Orient}_{B,1}, \text{Decide}_{B,1}, \text{Act}_{B,1}\}$. Furthermore, we can consider that the observed $B$ also can see the observer $A$’s all steps in OODA at the same time 1, $\text{Observe}_{B,1} = \{\text{Observe}_{A,1}, \text{Orient}_{A,1}, \text{Decide}_{A,1}, \text{Act}_{A,1}\}$. And, the observer $A$ can see a situation in which the observed $B$ are seeing the observer $A$ (e.g., in a chess game, a player is pondering about the thought of the opponent about which the opponent is observing the player), $\text{Observe}_{A,1} = \{\text{Observe}_{B,1}, \text{Orient}_{B,1}, \text{Decide}_{B,1}, \text{Act}_{B,1}\}$, where $\text{Observe}_{B,1} = \{\text{Observe}_{A,1}, \text{Orient}_{A,1}, \text{Decide}_{A,1}, \text{Act}_{A,1}\}$. These recursive observations between the observer $A$ and the observed $B$ (e.g., looking an object in a mirror with another mirror in opposite side of the mirror) can be infinite. However, because of a computational limitation of human or machine, the recursive observations between both will stop at a certain point.

In the multi-OODAs, we should consider some properties for the observed and time. Obviously, the observer $A$ can observe itself, $i = j$. For example, we can meditate ourselves. In some cases, the observed $B$ may not be only one actor, but it is a set of actors, $i = \{a_1, a_2, ..., a_n\}$, where $a_k$ is a unit actor for the set. The time, $t$, when the observer $A$ observes the observed $B$ can be a discrete time with a certain time period (e.g., 1 sec or 1 hour) or a continuous time.

4.1.5 PSAW in terms of OODA

A PSAW model may need to represent the four steps in OODA and their inputs/outputs to express various operating aspects for an actor. For example, an observer may observe the inputs and outputs of the OODA process of the observed, estimate
current situations, and predict the future situation for the observed. These observing, estimating, and predicting operations of PSAW correspond to the Observe and Orient steps in OODA. In other words, the observer in the Observe and Orient steps (or PSAW operations) can observe any of inputs/outputs (in Table 4.1) of the Observe, Orient, Decide, and Act steps of the observed.

![Figure 4.4 A situation of an observer in terms of Interpreted/Actual OODA](image)

In the view of an observer, all inputs/outputs in the four steps can be considered as two elements: interpreted elements and actual elements. If the elements are comprehended subjectively, they are interpreted elements (i.e., the two wheeled vehicles in the interpreted situation in Fig. 4.2.) which are derived from the actual elements (i.e.,
the wheeled and tracked vehicles in the actual situation in Fig. 4.2.). In other words, the real type of a target (i.e., actual elements) can be different from the reported type of the target (i.e., interpreted elements).

Fig. 4.4 shows a relationship between the interpreted elements in OODA, called an interpreted OODA, and actual elements in OODA, called an actual OODA. The figure depicts a situation when the observer observes the observed. In our PSAW model, the observer may see all elements of OODA sequence of the observed. For example, a spy investigates the report, organization, strategy, intention, and activities of the spy's opponent. The first OODA sequence in Fig. 4.4 is an OODA sequence for the observer who performs the OODA processes from the Observe step to the Act step. The observer in the Observe step can detect any element in the interpreted OODA (i.e., the second OODA) of the observed. The detected (or reported) element in the interpreted OODA sequence is used to estimate the actual current element and predict the future element in the actual OODA (i.e., the third OODA) by the observer in the Orient step.

Fig. 4.5 shows an illustrative example of the above situation with specific elements of OODA. According to requirements of constructing a PSAW model, elements of OODA are chosen. The elements of OODA (e.g., condition, observations, and plan in Table 4.1) in Fig. 4.5 can be chosen in terms of the modeling properties (i.e., a model boundary, exogenous variables, endogenous variables, a time horizon, and a time granularity). An observer contains 8 elements of the actual OODA (i.e., observing conditions, observations, knowledge, hypothesis, plan, COA, activity, and states). The observing conditions are a set of conditions when the observer observes the observed
(e.g., weather condition). The observations are all elements of the interpreted OODA in the observed. The knowledge in this example model is estimation of all elements of OODAs in the observed. Using the knowledge, the observer estimates possible current situations or predicts future situations. Then the observer decides plans influencing the development of COAs. The observer (now it is an actor following the COAs) executes some activities changing states of the observer, the observed, or the world.

4.2 PSAW-MEBN Reference Model
A PSAW-MEBN reference model specifies reference MFrags, RVs, and entities which support the design of a MEBN model for PSAW. Such a MEBN model is called a PSAW-MTheory. The PSAW-MTheory is designed to reason about the PSAW questions.
from Section 4.1.1 and can be supported by the reference model, called a *PSAW-MEBN reference model*. The PSAW-MEBN reference model is based on our discussion in the previous sections and provides a common semantics for a PSAW-MTheory, which can be used for different applications. Constructing an MTheory can be depending on various purposes as discussed in [Costa, 2005].

“Ultimately, the approach to be taken when building an MTheory will depend on many factors, including the model’s purpose, the background and preferences of the model’s stakeholders, the need to interface with external systems. etc. [Costa, 2005].”

The reference model should be useful to construct a PSAW-MTheory in a new application, so it should enable a PSAW-MTheory modeler to construct the PSAW-MTheory efficiently and effectively. Therefore, the reference model should be designed sufficiently to define precisely a PSAW-MTheory and flexibly to be used for various applications. We develop the PSAW-MEBN reference model to satisfy these two criteria (i.e., *sufficient detail* and *flexibility*) by adopting both of the use of expert knowledge and the design concept of modularity.

To make the reference model flexible and in sufficient detail regarding the characteristics of PSAW, the PSAW-MEBN reference model is designed with the following issues in which a PSAW-MTheory modeler may have difficulty. (1) How does the MTheory address the questions of PSAW? (2) What type of entities and RVs can be chosen to represent hypotheses for PSAW? (3) How can MFrags be decomposed for efficiently exploring the hypotheses? (4) What should be the relationships between RVs in the MFrags for to support efficient and accurate inference?
Fig. 4.6 depicts an illustrative example of an SSBN representing PSAW.

This SSBN example has are five levels: Situation Identification, Mission Identification, Activity Identification, Object Identification, and Detection (note that these levels are not necessarily general). At the detection level, there are two types of RVs; a Sensor Type and Report RV. The Sensor Type RV describes the type of a sensor, s, which is detecting a target, g, at time t, and is producing a report, r. The SensorType_g1_s1_t1 RV means the type of the sensor s1 detecting the target g1 at time t1. The Report_r1_g1_s3_t1 RV means the report r1 of the target g1 detected by the sensor s3 at time t1. At the object identification level, the reports of the target from the sensors at the detection level are integrated into an ObjectState RV. At the object identification level,
there are three ObjectState RVs. The first is for the target g1 at time t1. The second is for the target g1 at time t2. The third is for the target g1 at time t3. From these ObjectState RVs at times 1~3, an Activity RV for the target g1 is estimated. Note that the Activity RV occurs during times 1~3. This duration is specified by t123 in the Activity_g1_t123 RV. As another example, at the activity identification level, the Activity_g1_t67 RV means the activity runs during times 6 and 7. At the mission identification level, a Mission RV is integrated from the Activity RVs at the activity identification level. The Mission_g1_t1234567 RV represents the mission of the target g1 during times 1 through 7 covering the duration of the activities belonging to the mission. At the situation identification level, two Situation RVs with distinct time durations (i.e., times 1~9 and times 10~13) are depicted. Some or all Mission RVs can influence a Situation RV. For example, the Situation_g123_t123456789 RV is influenced by the Mission_g1_t1234567, Mission_g2_t1234567, and Mission_g3_t123456789 RV. This SSBN example is very simplified. In Section 4.1.2 and 4.1.3, we discussed several ontologies such as Space [Randell et al., 1992], Time [Allen, 1983], Situation [Yau & Liu, 2006][Baader, 2003], and SAW [Matheus et al., 2003][Anagnostopoulos et al., 2007]. These ontologies can be used for the development of an SSBN or an MTheory for PSAW. For example, the time relation precedes(t_i, t_j) [Allen, 1983] can be used for an RV representing whether a time t_i finishes before a time t_j starts. The space relation EC(x, y) [Randell et al., 1992], x is externally connected to y, can be used for an RV representing whether two regions x and y adjoin each other and they do not overlap. However, although in a simple case (e.g., two entities are associated), using such a relationship may be easy, it becomes quite
complicated in situations where more than two entities are associated. Also, combining probabilistic information about multiple overlapping relationships may create very challenging issues for representation and reasoning. In this dissertation, we focus on the simple case and leave such complicated situations as a future research topic.

The five levels of Fig. 4.6 can address the PSAW questions from Section 4.1.1. For example, the object identification level can answer questions such as “what type is the target?” The activity identification level can answer questions such as “what are the next activities performed by the (grouped) targets?” The mission identification level can answer questions such as “what is the meaning of the situation?” The situation identification level can answer questions such as “how much danger is there?”.

The SSBN example illustrates how an SSBN for PSAW can be composed. In the following sections, we discuss how an MTheory for PSAW can be composed. In Section 4.2.1, special entity types for the reference model are presented. In Section 4.2.2, RV types based on the entity types are presented. In Section 4.2.3, MFrags for the reference model are discussed.

4.2.1 Entities for the Reference Model
The examples in Fig. 4.1 and Fig. 4.6 allow us to identify some fundamental elements for PSAW. In this research, the observer, the sensor, the target, the reported target, and the time are determined as five fundamental elements. These elements can be represented as entities in a PSAW-MTheory. In PSAW, a target is meaningful, when it can be identified in terms of what it is and what it does. An activity operated by nothing or non-target is meaningless. And also a target which just exists without any operation (or
changes) forever is meaningless. In PSAW, we presume that there is always an observer observing (or being related to) the target through sensors over/at the time. If these fundamental elements don’t exist, we can’t be aware of a situation.

In MEBN, an entity type is a unique kind of thing, and can be instantiated to one or more entity instances which exist distinctly and independently. For example, Person can represent an entity type instantiating person entity instances (e.g., John and Mathew). An entity in the reference model can be classified into five categories, called PSAW-Entities; (1) the time entity $T$, (2) the observer entity $OR$, (3) the sensor entity $SR$, (4) the target entity $TR$, and (5) the reported target entity $RT$. Entities derived from these categories describe a situation in which an observer observes a target through sensors and interprets it as a reported target at a certain time. For example, an omnidirectional radar (as $SR$) detects a vehicle object (as $TR$) and produces a vehicle object report (as $RT$) at time $T1$ (as $T$). A user who uses a PSAW-MTheory can be represented as the observer entity $OR$. Commonly, it is not necessary to represent the observer entity explicitly in a PSAW-MTheory, because we already know about who are using the PSAW-MTheory. However, there are special situations in which a relationship between a target and an observer should be represented in a PSAW-MTheory (e.g., a target takes aim at an observer).

4.2.2 Random Variables for the Reference Model
A Random Variable (RV) represents the uncertainty of a set of values in mutually exclusive (i.e., events cannot occur simultaneously) and collectively exhaustive (i.e., a set of events covers all sample space outcomes). In MEBN, random variables (RVs) (e.g.,
resident, input, and context nodes) can contain ordinary variables (written as strings of lowercase letters) that can be filled by instances of entities (written as strings of uppercase letters). For example, Isa context nodes are used to specify the types of the entities that can fill an ordinary variable. Isa(p, PERSON) is an Isa context node. Only entities of type Person can fill in for p when creating instances of RVs in an MFragment containing the Isa(p, PERSON) node.

In the reference model, random variables containing ordinary variables from entity types T, OR, SR, TR, and/or RT are called PSAW-RVs. The reference model doesn’t limit the number of each of the ordinary variable types in PSAW-RVs. For example, a situation in which two vehicles communicate with each other and are detected by a COMINT sensor at time T1 can be described by a PSAW-RV, specified by Communicated(v1, v2, cmt, t), where v1, v2, cmt, and t are ordinary variables that can be filled in by a first vehicle entity (TR), a second vehicle entity (TR), a COMINT sensor entity (SR), and a time entity (T), respectively. Thus, two target entities (TR) are used in the PSAW-RV.

Each of the five entity types in Section 4.2.1 can have its own attributes. For example, an attribute of the time entity can be used for indicating the clock time associated with a timestamp variable, RealTime(t) (e.g., RealTime(T1) is 12:10:01 UTC for the time entity T1). An attribute for the sensor entity can represent the type of the sensor, SensorType(sr) (e.g., SensorType(sr1) can be an omnidirectional radar). Entities of the five entity types can be related to each other (e.g., SensorCondition(sr1, t1) and IsAlliance(or1, tr1)).
Semantically, the type for a PSAW-RV can depend on the context where an observer, sensor, target, target report, and times are involved. For example, Little & Rogova [2005] suggested main relations for continuant and occurrent things: (1) Topology/mereology relations (e.g., disjoint, joint, overlap, cover, reachable, unreachable, contain, and part of), (2) Direction (e.g., along, towards, east, west, south, north, similar, and opposite), (3) Distance (e.g., far, very far, near, and very near), Size (e.g., small, large, and same), (4) Relationships between time points (e.g., before, at the same time, start, finish, soon, very soon, resulting in, and initiating), and (5) Relationships between time intervals (e.g., disjoint, joint, overlap, inside, and equal). To develop a PSAW-RV, these relations which represent physical aspects can be considered. For the mental situations (Section 4.1.3), the elements of OODA in Table 4.1 can be used. As the elements of OODA in Fig. 4.5, the observing conditions, observations, knowledge, hypothesis, plan, COA, activity, and states can be the PSAW-RVs.

These PSAW-RVs can be classified into four kinds of RV: Observing condition RV ($OC_{RV}$), Reported object RV ($RT_{RV}$), Target object RV ($TR_{RV}$), and Situation RV ($SIT_{RV}$). Fig. 4.7 shows these core kinds of RV ($OC_{RV}$, $RT_{RV}$, $TR_{RV}$, and $SIT_{RV}$) and their instances.

**Definition 4.13 (Observing Condition RV)** An observing condition RV, $OC_{RV}$, is an RV indicating a condition of an observer or a sensor. An observing condition RV can depend on other observing condition RVs, $OC_{RV_i} \rightarrow OC_{RV_j}$, where $I$ is a set of indices for $OC_{RVs}$ and $j \notin I$.

---

7 A → B means B depends on A.
The observing condition RV can be decomposed into two sub-types (An internal condition RV and an external condition RV). An internal condition RV is an RV used for a condition of the internal states of the observer or the sensor. For example, a capability of an omnidirectional radar $sr1$ at time $t1$ can be represented as the external condition RV \text{Capability}($sr1$, $t1$). An external condition RV is an RV used for an observing condition between the observer/sensor and the observed. For example, the range between an omnidirectional radar $sr1$ and a moving target $tr1$ at time $t1$ can be represented as the internal condition RV \text{Range}($sr1$, $tr1$, $t1$).

**Definition 4.14 (Reported Object RV)** A reported (or interpreted) object RV, $RT\_RV$, is an RV used for attributes and/or relations for the reported object from the observer. A reported object RV can depend on a target object RV and/or an observing condition RVs, $TR\_RV \rightarrow RT\_RV$ and/or $OC\_RVs \rightarrow RT\_RV$.

For example, if a speed of the moving target $tr1$ observed by the omnidirectional radar $sr1$ at time $t1$ is interpreted as a reported speed of a reported moving target $rt1$, this situation can be represented as the reported object RV \text{ReportedSpeed}($rt1$, $t1$) depending on the target object RV \text{Speed}($tr1$, $t1$) and the observing condition RV \text{Range}($sr1$, $tr1$, $t1$). The observations for targets can be reported object RVs

**Definition 4.15 (Target Object RV)** A target object RV, $TR\_RV$, is an RV used for actual attributes and/or relations for the target object. A target object RV can depend on other target object RVs, $TR\_RV_i \rightarrow TR\_RV_j$, and/or situation RVs, $SIT\_RVs \rightarrow TR\_RV_j$, where $I$ is a set of indices for $TR\_RV$s and $j \notin I$. 

93
For example, an actual speed $\text{Speed}(tr1, t1)$ of the actual moving target $tr1$ at time $t1$ can depend on the type of the actual moving target $tr1$ $\text{TargetType}(tr1)$. The states, activities, plans, and missions of targets can be target object RVs.

**Definition 4.16 (Situation RV)** A situation RV, $SIT\_RV$, is an RV representing an aspect of a situation. The situation RV can depend on target object RVs, $TR\_RVs \rightarrow SIT\_RV$, and/or other situation RVs, $SIT\_RV_i \rightarrow SIT\_RV_j$, where $I$ is a set of indices for $SIT\_RVs$ and $j \notin I$.

For example, if there is a situation in which two moving targets ($tr1$, $tr2$) are attacking at time $t1$, this situation can be represented as an RV $\text{AttackingSituation}(tr1, tr2, t1)$ which depends on target type RVs $\text{TargetType}(tr1)$ and $\text{TargetType}(tr2)$.

![Figure 4.7 Four core kinds of RVs and examples of each](image-url)
Determining causality between PSAW-RVs is a difficult task because there may be possible unknown or hidden mechanisms influencing the events. The task may rely on experts’ knowledge and/or machine learning. Nevertheless, some restrictions to determine causality between events can be considered. We can assume that between two events (i.e., $A$ and $B$) if the event $A$ occurs before the event $B$, then the event $B$ can’t cause the event $A$. For example, a PSAW-RV Communicated($v1$, $v2$, $cmt$, $t2$) can’t cause to a PSAW-RV Communicated($v1$, $v2$, $cmt$, $t1$), but the opposite may be possible. Fig. 4.7 shows possible causalities (Conditional relationships in the figure). The observing condition RVs and target RVs can cause the reported object RVs. The target RV can cause the situation relation RVs. Fig. 4.7 contains three layers ($Situation$, $Object$, and $Observation$). The observation layer and the situation layer in Fig. 4.7 correspond to the detection and the situation identification in Fig. 4.6, respectively. The object layer in Fig. 4.7 contains the mission identification, the activity identification, and the mission identification object in Fig. 4.6.

4.2.3 The Five M_frag Groups

In MEBN, an M_frag is a fragment of a probabilistic graphical model in which the nodes contain variables that are placeholders for domain entities. M_frams can be instantiated with domain entities and combined into complex models with repeated structure. M_frams for PSAW, called $PSAW-M_frams$, contain PSAW-RVs as defined in Section 4.2.2. The PSAW-MEBN reference model suggests five groups: (1) Reported Object, (2) Observing Condition, (3) Target Object, (4) Situation, and (5) Context. These five groups can be constructed as five types of M_frams: (1) a $Reported$ $Object$ M_frag, (2)
an Observing Condition MFragment, (3) a Target Object MFragment, and (4) a Situation MFragment, and (5) a Context MFragment, respectively. Fig. 4.8 shows an example of PSAW-MFragments for an illustrative PSAW model.

4.2.3.1 Context MFragment
M Fragments in this group define distributions for resident nodes which are used for context nodes in other M Fragments. To construct some PSAW-MFragments (e.g., Target, Observing Conditions, Report, and Situation), contexts for each MFragment may or may not be defined. For example, the observing conditions MFragment has a context node ObserverOf(sr, tr) to determine a sensor entity and a target entity, where the sensor entity is an observer of the target entity. For context MFragments, three common context nodes

Figure 4.8 Illustrative example of PSAW-MFrag
(Predecessor, ObserverOf, and ActualObject) can be considered. We will discuss these RVs in the following MFrags.

### 4.2.3.2 Target M_frag

MFrags in this group define distributions for PSAW-RVs related to relations and attributes of targets. This M_frag contains a resident node Target RV and an input node Target RV, a recursive relationship. These nodes can have target identifiers and time identifiers. The difference between the resident node and the input node is that the input node has a previous time identifier and the resident node has a current time identifier. Therefore, the resident node at the current time is influenced by the input node at the previous time. To determine this time relationship, the context node Predecessor(pre_t, t) is included in the M_frag. The node Predecessor(pre_t, t) means that the time interval pre_t occurs immediately before the time interval t. The node Predecessor has its value set deterministically to represent sequential time steps.

This M_frag illustrates possible resident nodes and possible input nodes related to relations/attributes of a target. In a specific usage, a wide variety of nodes is possible. If the relations/attributes of the target do not change over time, the time identifiers in the resident nodes are omitted, and the input nodes for the resident nodes and the context node Predecessor are not necessary.

### 4.2.3.3 Observing Conditions M_frag

This M_frag represents knowledge about conditions of the sensor. This M_frag contains a resident node Observing Condition RV with the sensor identifiers, the target identifiers, and the time identifiers as its possible values. To determine the relationship
between the sensor identifiers and the target identifiers, the context node $ObserverOf(sr, tr)$ is included in the MFragment. The node $ObserverOf(sr, tr)$ means that a sensor $sr$ is the observer of a target $tr$, and has its value set deterministically. The context node $ObserverOf$ allows a situation in which a sensor observes many targets and a target is observed by many sensors as well.

### 4.2.3.4 Report MFragment

This MFragment represents knowledge about relations of a report object. This MFragment contains a resident node $Report RV$ and two input nodes $Observing Condition RV$ from $Observing Condition MFragment$ and $Target RV$ from $Target MFragment$. The node $Report RV$ has the report identifiers and the time identifiers as its possible values. To determine the relationship between the target and the report object, the context node $tr = ActualObject(rt)$ is included in the MFragment. The context node $tr = ActualObject(rt)$ means that if a target $tr$ is the actual object of a reported target $rt$, the context node is true. This context node makes a relationship between a reported object (e.g., a reported type from a report) and an actual target (e.g., an actual type from an actual target).

### 4.2.3.5 Situation MFragment

This MFragment defines distributions for PSAW-RVs related to a situation for targets. In the example, the situation MFragment contains a resident node $Situation$ and two $Target RV$ input nodes for each target. The resident node $Situation$ can have the target identifiers and the time identifiers. In this MFragment, only two targets are treated, however many targets are designed in such a $Situation MFragment$. This MFragment integrates all or some relations between targets to produce a new aspect from these relations. The situation can vary according to
what the observer related to this MFragment want to see. For example, the situation can be whether two vehicles are in the same group, whether two vehicles are communicating to each other, what the relationship between two vehicles is, whether a causal relationship between two vehicles exists, and how activities from two vehicles are related to each other.

In PSAW, understanding a situation in which targets operate for their own purposes is one of the important issues. Identifying just the type of a target is an insufficient task for PSAW. The meaning of awareness is not to perceive and estimate actual properties of a target, but is to understand, interpret, and explain the relationships between targets. Kokar et al [2009] stated: “The main part of being aware is to be able to answer the question of ‘what’s going on?’". Awareness of a situation is subjective according to an observer, who is aware of the situation. The modeler, who is developing a probabilistic ontology to support PSAW, should define what situation will be considered and explained through all observation from the world.

4.3 Conclusion

Previously, a PSAW-MTheory had to be designed from scratch. Designing PSAW-MEBN models without any reference model is inefficient. To address this issue, this chapter presented a PSAW-MEBN reference model which can address questions of interest for PSAW. The PSAW-MEBN reference model was derived from a concept of PSAW in OODA. For this, we discussed the properties of PSAW and the properties of OODA. We presented the reference entities, the reference RVs, and the reference MFragment for the PSAW-MEBN reference model.
CHAPTER FIVE: HUMAN-AIDED MEBN LEARNING FOR PSAW

If we have data for a domain, it can improve efficiency of developing an MTheory and enhance the quality of the resulting MTheory. Therefore, it is necessary to implement MTheory modeling automation using a machine learning method. MEBN learning is a method to model an MTheory from the domain expert's knowledge as well as data. Although technologies for machine learning have improved dramatically, the necessary capabilities to build a model correctly are still lacking. In complex problems, the search space for building a model is too large and complex to investigate all possible structures, variables, and parameters. For this reason, we propose a MEBN learning framework in the PSAW domain which relies partially on expert's knowledge and insight to reduce the search space.

In Chapter 3, we presented MEBN-RM. Also, we presented the PSAW-MEBN reference model (Chapter 4), which provides support for defining the structure of a PSAW-MTheory. In this Chapter, we introduce a process for Human-Aided MEBN Learning for PSAW (HMLP) which is a framework to assist in solving the learning problem. HMLP uses MEBN-RM and the PSAW-MEBN reference model to develop an MTheory structure. MFrags in the MTheory structure depend on parameters, which are not specified in the reference model. The parameters for RVs can be estimated from data by a parameter learning method called MEBN parameter learning. In this research,
HMLP uses MEBN parameter learning for the MTheory structure. MEBN structure learning for HMLP is deferred to future research.

### 5.1 Introduction

In this section, we introduce an illustrative running example of relational data and define what MEBN learning is composed of.

#### 5.1.1 Illustrative Running Example of Relational Data for MEBN Learning

For an illustrative example for MEBN learning, a situation identification relational database (RDB) is introduced.

![Figure 5.1 Schema of a Situation Identification Relational Database](image)

Fig. 5.1 shows a schema for the situation identification RDB. In the example RDB schema, there are 14 relations: Region, Situation, Location, Time, Speed, Speed_rpt,
**ActualObject, ObserverOf, Vehicle, VehicleType, Predecessor, ReportedVehicle, MTI, and MTI.Condition.** For a detailed description of this RDB, see the RM section in Chapter 2. The relation *Region* is for region information in this situation which can contain a region index (e.g., *region1* and *region2*). The relation *Time* is for time information which is a time stamp representing a time interval (e.g., *t1* and *t2*). The relation *Vehicle* is for vehicle information which is an index of a ground-vehicle (e.g., *v1* and *v2*). The relation *VehicleType* indicates a type of the vehicle (e.g., *Wheeled* and *Tracked*). The relation *MTI* is for a moving target indicator (e.g., *mti1* and *mti2*). An MTI can be in a condition (e.g., *Good* and *Bad*) depending on weather and/or maintenance conditions. The relation *MTI_Condition* indicates the condition of an MTI. The relation *Location* is for a location where a vehicle is located. The relation *Situation* indicates a level of danger to a region at a time (e.g., *Low* and *High*). The relation *ReportedVehicle* indicates a reported vehicle from an MTI. The relation *Speed* indicates an actual speed of a vehicle, while the relation *Speed_rpt* indicates a reported speed of the vehicle from an MTI. The relation *ActualObject* maps a reported vehicle to an actual vehicle. The relation *ObserverOf* indicates that an MTI observes a vehicle. The relation *Predecessor* indicates a temporal order between two-time stamps.

Table 5.1 shows parts of the relations of the situation identification RDB for the schema in Fig. 5.1. As shown Table 5.1, we choose six relations (*Vehicle, Time, Region, VehicleType, Location, and Situation*), which are used for an illustrative example through this chapter. For example, the relation *Vehicle* contains a primary key *VID*. The relation
VehicleType contains a primary key v/Vehicle.VID, which is a foreign key from the primary key VID in the relation Vehicle and an attribute VehicleType.

<table>
<thead>
<tr>
<th>Vehicle</th>
<th>Time</th>
<th>Region</th>
<th>VehicleType</th>
<th>Location</th>
<th>Situation</th>
</tr>
</thead>
<tbody>
<tr>
<td>VID</td>
<td>TID</td>
<td>RID</td>
<td>v/Vehicle. VID</td>
<td>v/Vehicle. VID</td>
<td>v/Vehicle. VID</td>
</tr>
<tr>
<td>v1</td>
<td>t1</td>
<td>rgn1</td>
<td>v1</td>
<td>t1</td>
<td>rgn1</td>
</tr>
<tr>
<td>v2</td>
<td>t2</td>
<td>rgn2</td>
<td>v2</td>
<td>t2</td>
<td>rgn1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

5.1.2 Elements of MEBN Learning from Relational Data
In this section, we introduce various subordinate learning elements in MEBN learning and our scope for MEBN learning. An MTheory can contain MFrags, graphs, context/resident nodes, FOL function or predicate for nodes, CLDs (Class Local Distributions) for nodes, and parameters for CLDs. These elements can be subject to MEBN learning. For example, we can consider (1) parameter learning for a CLD, (2) CLD learning for an MNode, (3) graph learning, (4) context node learning, and (5) MTheory learning.

Fig. 5.2 shows the possible MTheory learning subjects in MEBN learning. In Fig. 5.2, an MRoot is a dummy component which can contain candidate MTheories. Each candidate MTheory can contain many MFrags. Each MFrag is associated with context nodes. Each MFrag can contain many candidate graphs. For each MFrag, the context nodes can be learned and an optimal graph among candidate graphs can be learned. Each graph can contain MNodes. Each MNode can contain many candidate CLDs. Each CLD
can contain many candidate parameters for the CLD. MNode learning is to find an optimal CLD among candidate CLDs in an MNode through finding an optimal parameter from candidate parameters.

Figure 5.2 Learning Subjects for MEBN Learning

A basic MEBN learning method for a relational dataset was suggested [Park et al., 2013a][Park et al., 2013b]. In this research, we focus on MEBN parameter learning given an MTheory structure containing predefined MFrags, Graphs, MNodes, and CLDs.

5.2 A Process for Human-Aided MEBN Learning for PSAW

The process this research presents uses expert knowledge to define the set of possible parameters and structures. The process, called Human-aided MEBN learning for PSAW (HMLP), modifies UMP-ST [Carvalho et al., 2016] to incorporate learning from data. As with UMP-ST, HMLP includes four steps (Fig. 5.3): (1) Analyze Requirements, (2) Define World Model, (3) Construct Reasoning Model, and (4) Test Reasoning Model.
Figure 5.3 Process for Human-Aided MEBN Learning

Initial inputs of the process can be needs and/or missions from stakeholders. In the *Construct Reasoning Model* step, a training dataset can be an input for MEBN learning. In the *Test Reasoning Model* step, a test dataset can be an input for the evaluation of a learned reasoning model. An output of the process is a reasoning model (in our case, a PSAW-MTheory). The following sub-sections describe these four steps.

### 5.2.1 Analyze Requirements

This step is to identify requirements for development of a reasoning model. As with requirements in UMP-ST (Section 2.5), requirements in the HMLP define goals to be achieved, queries to answer, and evidence to be used in answering queries. Also, the requirements should include performance criteria for verification of the reasoning model. These performance criteria are used in the *Test Reasoning Model* step. Before the *Analyze Requirements* step begins, stakeholders provide their initial requirements containing needs, wants, missions, and objectives. These initial requirements may not be defined formally. Therefore, to clarify the initial requirements, operational scenarios are
developed. In other words, the operational scenarios are used to identify the goals, queries, and evidence in the requirements.

This step contains three sub-steps (Fig. 5.4): (1) an Identify Goals step, (2) an Identify Queries/Evidence step, and (3) a Define Performance Criteria step.

5.2.1.1 Identify Goals

The goals represent missions of the reasoning model we are developing. In this step, we can use a set of PSAW questions (see Appendix B), which enables us to grasp some ideas for what questions the reasoning model for PSAW should answer (e.g., does a (grouped) target exist?, what are the environmental conditions?, and what group does the target belong to?). Requirement 5.1 illustrates a goal we will use.

Requirement 5.1

Goal 1: Identify characteristics of a target.
5.2.1.2 Identify Queries/Evidence

The queries are specific questions for which the reasoning model is used to estimate and/or predict answers. The evidence consists of inputs used for reasoning. From these sub-steps, a set of goals, a set of queries for each goal, and a set of evidence for each query are defined. The following shows an illustrative example of defining a requirement.

**Requirement 5.1**

**Goal 1**: Identify characteristics of a target.

**Query 1.1**: What is the speed of the target at a given time?

**Evidence 1.1.1**: A speed report from a sensor.

...

5.2.1.3 Define Performance Criteria

It is necessary to evaluate whether the results for a reasoning model which will be learned from data address performance requirements in terms of reasoning. Criteria for evaluating reasoning performance include: *Speed* (e.g., execution or computation time for reasoning), *Accuracy* (e.g., measuring gap between an actual value and estimation) and *Resource Usage* (e.g., memory or CPU usage). In some situations, execution time for a reasoning model is the most important factor. In other cases, accuracy for a reasoning model may be more important. For example, an initial missile tracking may require high-speed reasoning to estimate the missile trajectory, while matching faces in a security video against a no-fly database may prioritize accuracy over execution time.

The performance criteria in the requirements can be specified in terms of some measure of accuracy (e.g., the Brier score [Brier, 1950] or the continuous ranked
probability score (CRPS) [Gneiting & Raftery, 2007] (these metrics are defined in Appendix C)). For example, we might require that the average of CRPS values between ground truth and estimated results from a reasoning model shall be less than a given threshold.

The performance criteria are determined by stakeholder agreement. Such performance criteria can be acquired through the following approaches: (1) survey, (2) experience, and (3) standard metrics drawn from published literature and standards. (1) Performance criteria can be derived by agreement of stakeholders using survey. (2) Subject matter experts can provide appropriate performance criteria from their experience. (3) Standards or literature can be used to obtain such performance criteria.

5.2.2 Define World Model

The Define World Model step develops a world model consisting of a structure model and rules. The world model describes a target situation of concern that is the subject of PSAW. The structure model can contain entities (e.g., target and sensor), variables (e.g., Speed and DangerLevel), and relations (e.g., location and situation). The rules describe the causal relationships between entities in the structure model (e.g., the type of a target can influence the speed of the target). The causal relationships can contain more specific information such as types of distributions and parameters for the distributions which will be used to develop an initial MTheory in the next step. The structure model and the rules provide a clear idea by which the reasoning model can be formed.
This step decomposes into two sub-steps (Fig. 5.5): (1) a Define Structure Model step and (2) a Define Rules step. The Define Structure Model step defines the structure model from the requirements, domain knowledge and/or existing data schemas. The structure model is used to identify rules. The Define Rules step defines a rule or an influencing relationship between attributes (e.g., $A$ and $B$) in relations for the structure model. The influencing relationship is a relationship between attributes in which there is an unknown causality between the attributes (e.g., $influencing(A, B)$). If we know the causality, the influencing relationship becomes a causal relationship (e.g., $causal(A, B)$). For many parent attributes which influence a child attribute (or variable), a brace is used to indicate a set of parent attributes (e.g., $causal\{A, B\}, C$). The child attribute is called a Target Attribute (or Variable). Also, the set of rules should satisfy the No-cycle condition which means that the generated SSBN will contain no directed cycles (Section 2.3.1).

5.2.2.1 Define Structure Model

The Define Structure Model step uses requirements, domain knowledge and/or existing data schemas to develop a structure model. The structure can be represented in a modeling language (e.g., Entity–Relationship (ER) model, Enhanced Entity–Relationship (EER), Relational Model (RM), or Unified Modeling Language (UML)). The structure model can contain information about entities, attributes, and groups for the entities and the attributes (e.g., a relation in RM).
We need to define how to consider the world model in terms of the closed-world assumption and the open-world assumption. The *closed-world assumption* (CWA) means that data, not known to be true, in a database is considered as false, while in the *open-world assumption* (OWA) it is considered as unknown that can be either true or false [Reiter, 1978]. In the world model, entities, relations, and attributes can be treated according to either CWA or OWA. For example, in CWA, if there is a set of disease entities, we assume the only diseases are the ones represented in the RDB. In OWA, there may be other disease entities in addition to the ones represented in the RDB. Considering CWA or OWA depends on the task and the quality of the data or knowledge. If it is sufficient for the task to assume we know all diseases (although in the real world, it is impossible), CWA can be used. As another example, there are a group of trees in a region and we are trying to identify the type of the trees. However, a method to count the trees performs poorly. In this case, it may make sense that data from such a method is treated according to OWA (although we can identify the type of the trees). Therefore, the determination for CWA or OWA for data or knowledge can depend on how these fit well the real world and on the task. This can be an issue of data quality. If our data or
knowledge fits well to the real world, we may use CWA. If our data or knowledge does not fit well to the real world, we may use OWA. How to measure such a quality? We may need an approach to qualify the fitness by matching between data and the real world. However, the topic of data quality goes beyond our research.

The original formulation of the relational model assumed a closed world [Date, 2007]. Date [2007] discussed the problem of using OWA in the relational model. Under OWA, data, not known to be true, is considered as unknown, which means that we don’t know whether it is true or false. In discussing this approach, Date [2011] stated that it corresponds to a three-valued logic (3VL) containing three truth values (e.g., true, false, and unknown). However, the relational model was not developed for such a logic (but it is based on two-valued logic [Date, 2011]). Date stated that query results under the assumption of the three-valued logic for the relation model can be wrong. “Nulls and 3VL are supposed to be a solution to the “missing information” problem—but I believe I’ve shown that, to the extent they can be considered a “solution” at all, they’re a disastrously bad one. [Date, 2011].”

The discussion from Date was about a database as a repository for storing and querying data, but not for developing statistical models of the data. Our problem is to learn a statistical model from the RDB. Therefore, we are not concerned about getting the right response queries under the three-valued logic, so Date’s criticism does not apply. For our purposes, “unknown” means missing data. Statisticians have developed rigorous, theory based solutions to the “missing information” problem. Incorporating these into MEBN learning from an RDB would give a satisfactory solution to the missing data
problem. For instance, to deal with missing data, we can use an expectation–
maximization (EM) algorithm [Dempster et al., 1977] or multiple imputation method
[Rubin, 1987][Schafer, 1997]. The EM algorithm consists of two steps: the expectation
step and maximization step. In the expectation step, sufficient statistics for missing data
are replaced by expected values using estimation from the current estimates of parameters
and observed data. In the maximization step, estimated sufficient statistics in the
expectation step are used to find new maximized parameters for data. The maximized
parameters used in the next expectation step again to estimate the missing data. These
steps -- expectation and maximization -- iterate until the parameters converge. The
multiple imputation method consists of three steps: imputation, analysis, and pooling. In
the imputation step, \( m \) complete data sets are generated from an approximate posterior
predictive distribution for the missing data conditional on the observed data. In the
analysis step, the completed data sets are analyzed by using standard analysis methods
(e.g., estimating means and standard errors). In the pooling step, the \( m \) analysis results are
combined into a final single result (e.g., a mean and a standard error) reflecting data.

In this research, we follow a limited closed world assumption to maintain
consistency between RM and MEBN in terms of MEBN learning:

[Assumption 1] No Missing Data: Values of all RVs for entities explicitly represented
in the database are known.

[Assumption 2] Boolean RV: For Boolean RVs, if the database does not indicate that
the value is true, it is assumed false.
We make no assumptions about entities that have not yet been represented in the database. The purpose of learning is to define a probability distribution for the attributes and relationships for new entities. Relaxing Assumption 1 and Assumption 2 is a topic for future research.

A requirement specifies a query and evidence for the query. The elements of the requirement are used to define corresponding elements in the structure model. For example, suppose that the requirements specify queries for the attributes Speed (Query 1.1) and Speed Report (Evidence 1.1.1) for a target $g$ at a time $t$. Based on these requirements, we know that these two attributes should be included in the structure model. We can then identify additional attributes related to these attributes by expert knowledge. For example, a TargetType attribute for the target $g$ most likely influences the Speed attribute and the Speed attribute at the previous time probably influences the current Speed attribute. Therefore, these attribute (TargetType and PreviousSpeed) can be included in the structure model.

The PSAW-MEBN reference model in Chapter 4 can be used in this step to identify these possible entities, variables, and relationships. The reference model provides information about possible entities (i.e., $T$, $OR$, $SR$, $TR$, and $RT$) involved in the situation. For example, from these entities and Requirement 5.1 in Section 5.2.1, we can see that the requirement mentions a time ($T$), a sensor ($SR$), and a target ($TR$). However, the requirement misses a reported target entity ($RT$). When defining the structure model, we must add the reported target entity into the structure model.
Also, the PSAW-MEBN reference model provides information about possible groups of MFrags (i.e., *Observing Condition*, *Reported Object*, *Target Object*, *Situation*, and *Context*) (Section 4.2.3). These possible groups are used to identify corresponding relations in the structure model. For example, the following list shows possible relations classified by the PSAW-MEBN reference model from the example RM in Fig.5.1. These relations contain their possible attributes.

**Observing Condition Group**
1. *mti_condition* Relation  
   Attribute: \{MTI\_Condition\}

**Target Object Group**
2. *vehicletype* Relation  
   Attribute: \{VehicleType\}
3. *speed* Relation  
   Attribute: \{Speed, PreviousSpeed\}
4. *location* Relation  
   Attribute: \{Location\}

**Reported Object Group**
5. *speed\_rpt* Relation  
   Attribute: \{Speed\_RPT\}

**Situation Group**
6. *situation* Relation  
   Attribute: \{DangerLevel\}

**Context Group**
7. *actualobject* Relation  
   Attribute: \{ActualObject\}
8. *observerof* Relation
9. *predecessor* Relation

Relations in the groups involve attributes of entities. These attributes can be represented as random variables (or MNodes). For example, the attribute *DangerLevel* in the relation *Situation* can be an MNode *DangerLevel* whose type is a *Situation* RV in the PSAW-MEBN reference model.
5.2.2.2 Define Rules

In the Define Rules step, causal relationships between random variables can be suggested by the PSAW-MEBN reference model. For example, a Reported Object RV (e.g., Speed_RPT) depends on a Target Object RV (e.g., Speed). Also, expert knowledge can provide some causal relationships between RVs. For example, an expert can note that the RV VehicleType most likely influences the RV Speed and an RV PreviousSpeed also likely influences the current RV Speed. These beliefs from expert knowledge become a causal relationship rule as shown in the following.

**Rule 1**: causal({VehicleType, PreviousSpeed}, Speed)

**Rule 2**: causal(VehicleType, DangerLevel)

**Rule 3**: causal({Speed, MTI_Condition}, Speed_RPT)

Rules 1 and 2 are derived from expert knowledge, while Rule 3 is derived from the reference model. Also, in Section 4.2.3.1, the PSAW-MEBN reference model provided knowledge about special context variable types (i.e., context types ActualObject, ObserverOf, and Predecessor) to link entities determined in different MFrags. Thus, the relation *actualobject* is used as the context type *ActualObject*, the relation *observerof* is used as the context type *ObserverOf*, and the relation *predecessor* is used as the context type *Predecessor*.

In the Define Rules step, a (conditional) local distribution for an attribute (e.g., the *speed* attribute) can be defined by expert knowledge. In reality, we can meet a situation in which there is no dataset for a rule and all we have is expert knowledge. For example, a conditional local distribution for the *speed* attribute given the RV *VehicleType* can be identified by a domain expert (e.g., if a vehicle type is wheeled, then the speed of the
vehicle on a road is normally distributed with a mean of 50MPH and a standard deviation of 20MPH). The rules derived in this step are used in the next step to construct an MTheory and the MTheory will be learned by MEBN parameter learning.

In this step, we determine whether data can be obtained for the attribute, and if so, either collect data or identify an existing dataset. We usually divide the data into a training dataset and a test dataset. If no data can be obtained, we use the judgment of domain experts to specify the necessary probability distributions. For example, a belief for the target type attribute can be \( P(\text{Wheeled}) = 0.8 \) and \( P(\text{Tacked}) = 0.2 \). If neither data nor expert judgment is available, we consider whether the attribute is really necessary. For this, we can return the Analyze Requirements step to modify the requirements.

5.2.3 Construct Reasoning Model

The Construct Reasoning Model step develops a reasoning model from a training dataset, a structure model, and rules.
This step decomposes into two sub-steps (Fig. 5.6): (1) a *Map to Reasoning Model* step and (2) a *Learn Reasoning Model* step. The *Map to Reasoning* converts the structure model and rules in the world model to an initial reasoning model. The *Learn Reasoning Model* uses a machine learning method to learn the model from a training dataset.

### 5.2.3.1 Map to Reasoning Model

In the *Map to Reasoning Model* step, MEBN-RM is used as a reference for a mapping rule between RM and MEBN. The relations which are grouped in Section 5.2.2 can be converted to MFrags in an initial MTheory (MTheory 5.1) using the MEBN-RM mapping algorithm in Section 3.6.

#### 5.2.3.1.1 Perform Entity-Relationship Normalization

Before performing MEBN-RM, the relations in Table 5.1 are normalized by the Entity-Relationship Normalization (Definition 3.1). For example, in the relations in Table 5.1, we can notice that the relation *VehicleType* has as its primary key a single foreign key imported from the relation *Vehicle*. They (*Vehicle* and *VehicleType*) can be merged into a relation *Vehicle*. The following table shows the normalized table. Note that after the Entity-Relationship Normalization, any foreign key in a relation comes from a certain entity relation (not relationship relation), which has only one attribute for its primary key, so there is no need to indicate which primary key is used for the entity relation and we can simplify the notation for a foreign key (e.g., *rgn/Region.RID* and *v/Time.TID*). For example, the notation of the foreign key for the vehicle (i.e., *v/Vehicle.VID*) in the relation *Location* (Table 5.1) can be simplified as *v/Vehicle*.  

117
### Table 5.2 Normalized relational dataset from Table 5.1

<table>
<thead>
<tr>
<th>Time</th>
<th>Region</th>
<th>Vehicle</th>
<th>Location</th>
<th>Situation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TID</td>
<td>RID</td>
<td>VID</td>
<td>Vehicle Type</td>
<td>Vehicle t/Time</td>
</tr>
<tr>
<td>t1</td>
<td>rgn1</td>
<td>v1</td>
<td>Wheeled</td>
<td>v1 t1 rgn1</td>
</tr>
<tr>
<td>t2</td>
<td>rgn2</td>
<td>v2</td>
<td>Tracked</td>
<td>v1 t2 rgn1</td>
</tr>
</tbody>
</table>

#### 5.2.3.1.2 Perform MEBN-RM Mapping

The relations in Figure 5.1 can be converted to MFags in an initial MTheory (MTheory 5.1) using the MEBN-RM mapping algorithm (Section 3.6). For example, the relation `mti_condition` is converted to the MFrag 1, F1, `Mti_Condition` described from the lines 1 and 4. Also, the attributes in relations can be resident nodes in the initial MTheory using the resident node mapping defined in MEBN-RM. For example, the attribute `VehicleType` for the vehicle `v` becomes a resident node `VehicleType(v)`. The attribute `Speed` for the vehicle `v` and at the time `t` becomes a resident node `Speed(v, t)`.

**MTheory 5.1: Initial Situation Identification**

```plaintext
F1: Mti_condition
   [C: Isa (v, VEHICLE), Isa (mti, MTI), Isa (t, TIME)]
   [R: MTI_Condition(v, mti, t)]
F2: Vehicle
   [C: Isa (VID, VEHICLE)]
   [R: VehicleType(VID)]
F3: Speed
   [C: Isa (v, VEHICLE), Isa (t, TIME)]
   [R: Speed (v, t)]
F4: Location
   [C: Isa (v, VEHICLE), Isa (t, TIME)]
   [R: Location (v, t)]
```
The initial MTheory, which is directly derived from an RM using MEBN-RM, can be learned using a dataset for each relation associated with the M_frag in the initial MTheory. More specifically, the parameter for the distribution of each RV in the M_frag is learned from a corresponding dataset of the relation for the M_frag. For example, the M_frag Situation is derived from the relation Situation in Table 5.2. The parameter for the distribution of the variable DangerLevel (Line 23 in MTheory 5.1) can be learned from the dataset of the attribute DangerLevel (Table 5.2). An RV (e.g., DangerLevel) in MEBN can contain a default distribution which is used for reasoning, for cases in which none of the conditions associated with parent RVs is valid. In MEBN, the parameter for the default distribution should be learned from a dataset containing such cases.
5.2.3.1.3 Update Reasoning Model using the Rules

The initial MTheory can be updated by the rules defined in Section 5.2.2.2. We have three rules for the three variables Speed, DangerLevel, and Speed_RPT. Each variable is associated with its parent variables (e.g., Pa(DangerLevel) = \{VehicleType\})\(^8\). If the parent variables in the rule for a child variable are in the MFrag where the distribution of the child variable is defined, the dataset for the relation associated with the MFrag is used for learning. For example, suppose that in Table 5.2, there is an attribute VehicleSize in the relation Vehicle. The attribute VehicleSize becomes a variable VehicleSize in the MFragment Vehicle using MEBN-RM. If there is a rule such as a causal(VehicleType, VehicleSize), the dataset in the relation Vehicle is used to learn the parameter for the distribution of the variable VehicleSize. However, if the parent variables in the rule for a child variable are resident in different MFrags where the child variable is not defined, the relations associated with these MFrags for the child variable and the parent variables should be joined to generate a joined dataset containing both datasets for the child and the parents. Then, the dataset for the joined relation from these relations is used for learning. For such a joining, target (or child) variables from a set of rules play an important role. The target variables in the MFragment given their parent variables are learned using the joined relation containing all attributes related to the target variables. In the following, we describe how to join relations in a relational database (e.g., the situation identification relational database). Rule 1 specifies that the probability distribution of the variable Speed depends on the values of variables PreviousSpeed and VehicleType. To learn the parameter for the variable Speed in this situation, it is not enough to use only the

\(^8\) Pa(X) is the set of parent nodes of the node X.
dataset from the relation \textit{Speed}, because the dataset doesn't contain information associated with the variable \textit{VehicleType}. Therefore, relations related to each target variable and its parent variables should be joined. For this purpose, we need to define a joining rule. The variable \textit{PreviousSpeed} indicates a variable \textit{Speed} which happens just before a current time, so the relation \textit{Predecessor}, which indicates a previous time and a current time, is also used for this joining for Rule 1. In other words, the relations \textit{Speed}, \textit{VehicleType}, and \textit{Predecessor} are joined.

\textbf{(1) Join Relations}

Now, let us discuss how to join these relations. A new dataset from the joined relation is called a \textit{joined dataset}. For example, the attributes (e.g., \textit{VehicleType} and \textit{DangerLevel}) which are located in different relations can be joined. Joining in RM is an operation to combine two or more relations. In the example, the relation \textit{Vehicle} can be joined to the relation \textit{Situation} through the relation \textit{Location}, because the relation \textit{Location} contains the attributes \textit{v/Vehicle}, \textit{t/Time}, and \textit{rgn/Region} corresponding to the primary key, \textit{VID}, in the relation \textit{Vehicle} and the primary key, \textit{t/Time} and \textit{rgn/Region}, in the relation \textit{Situation}.

There are several joining rules (e.g., Cartesian Product, Outer Join, Inner Join, and Natural Join) [Date, 2011]. Table 5.3 shows an illustrative example of a joined dataset derived from Table 5.2 using Inner Join. Inner Join produces all tuples from relations as long as there is a match between values in the columns being joined. Table 5.3 shows the result of performing an inner join of the relations \textit{Situation} and \textit{Vehicle} through the relation \textit{Location} and then selecting the columns to be used for learning. The rows (or
tuples) in the relations *Situation* and *Vehicle* are joined when rows of the attributes *v/Vehicle*, *t/Time*, and *Location/Region* in the relation *Location* match rows of the attribute *VID* in the relation *Vehicle* and rows of the attributes *rgn/Region* and *t/Time* in the relation *Situation*. The first column denotes cases for the matched rows. The second column (*Vehicle.VehicleType*) denotes the rows from the attribute *VehicleType* of the relation *Vehicle* in Table 5.2. The third column (*Location.v* and *Vehicle.VID*) denotes the matched rows between the attribute *v* from the relation *Location* and the attribute *VID* from the relation *Vehicle*. The fourth column (*Location.t* and *Situation.t*) denotes the matched rows between the attribute *t* from the relation *Location* and the attribute *t* from the relation *Situation*. The fifth column (*Location.Location* and *Situation.rgn*) denotes the matched rows between the attribute *Location* from the relation *Location* and the attribute *rgn* from the relation *Situation*. The sixth column (*Situation.DangerLevel*) denotes the rows from the attribute *DangerLevel* from the relation *Situation*. 
Table 5.3 shows the joined dataset for the attributes VehicleType and DangerLevel. Now, let us assume that the attribute DangerLevel will be a target variable depending on the variable VehicleType (i.e., Rule 2: causal(VehicleType, DangerLevel)). For each instance of the target variable DangerLevel, Table 5.3 provides relevant information about all the configurations of its parents (i.e., the parent variable VehicleType). For example, there is the value High for the danger level in the situation at Region5 in Time17 (i.e., Cases 17 and 18). The value High is associated with the wheeled Vehicle10 and the wheeled Vehicle11. In other words, two parent instances (i.e., the wheeled Vehicle10 and the wheeled Vehicle11) influence the target instance (i.e., the value High). The following shows a query script⁹ which is an example using Inner Join for Table 5.3.

Table 5.3 Joined dataset

<table>
<thead>
<tr>
<th>Case</th>
<th>VehicleType</th>
<th>Location.Vehicle</th>
<th>Location.VID</th>
<th>Location.Situation</th>
<th>Location.Region</th>
<th>Situation.DangerLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tracked</td>
<td>Vehicle13</td>
<td>Time18</td>
<td>Region6</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Tracked</td>
<td>Vehicle15</td>
<td>Time21</td>
<td>Region7</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Tracked</td>
<td>Vehicle17</td>
<td>Time24</td>
<td>Region8</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Tracked</td>
<td>Vehicle19</td>
<td>Time27</td>
<td>Region9</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Wheeled</td>
<td>Vehicle21</td>
<td>Time30</td>
<td>Region10</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Wheeled</td>
<td>Vehicle23</td>
<td>Time33</td>
<td>Region11</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Wheeled</td>
<td>Vehicle0</td>
<td>Time2</td>
<td>Region0</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Tracked</td>
<td>Vehicle1</td>
<td>Time2</td>
<td>Region0</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Tracked</td>
<td>Vehicle2</td>
<td>Time5</td>
<td>Region1</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Wheeled</td>
<td>Vehicle3</td>
<td>Time5</td>
<td>Region1</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Tracked</td>
<td>Vehicle4</td>
<td>Time8</td>
<td>Region2</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Tracked</td>
<td>Vehicle5</td>
<td>Time8</td>
<td>Region2</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Tracked</td>
<td>Vehicle6</td>
<td>Time11</td>
<td>Region3</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Tracked</td>
<td>Vehicle7</td>
<td>Time11</td>
<td>Region3</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Tracked</td>
<td>Vehicle8</td>
<td>Time14</td>
<td>Region4</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Tracked</td>
<td>Vehicle9</td>
<td>Time14</td>
<td>Region4</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Wheeled</td>
<td>Vehicle10</td>
<td>Time17</td>
<td>Region5</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Wheeled</td>
<td>Vehicle11</td>
<td>Time17</td>
<td>Region5</td>
<td>High</td>
<td></td>
</tr>
</tbody>
</table>

⁹ In this research, we used MySQL, an open-source relational database management system, and Structured Query Language (SQL) supported by MySQL.
**SQL script 5.1:** Joining for Table 5.3

```sql
SELECT Vehicletype, Location.v, Location.t, Location.Location, Dangerlevel
FROM Situation
JOIN Location ON Situation.rgn = Location.Location &&
Situation.t = Location.t
JOIN Vehicle ON Vehicle.VID = Location.v
```

SQL script 5.1 joins the relations *Situation* and *VehicleType* through the relation *Location*. In other words, the rows (or tuples) in the relations *Situation* and *VehicleType* are joined as shown Table 5.3 in which the two attributes (*VehicleType*, *DangerLevel*) are connected through the attributes of the relation *Location*. The joined table shows how the dataset of the attribute *VehicleType* and the dataset of the attribute *DangerLevel* are linked.

We introduced how to join relations according to given rules. In the following, we discuss how to update an MFragment from the given rules. The initial situation identification MTHeory (MTHeory 5.1) was constructed by MEBN-RM. Each MFragment in the initial MTHeory contains resident nodes without any causal relationship between the resident nodes. The given rules enable the resident nodes to specify such causal relationships. Therefore, the MFragment in the initial MTHeory may be changed according to the updated resident nodes with the causal relationships by the given rules. This process contains three steps: Construct input/parent nodes, Construct context nodes, and Refine context nodes.

**(2) Construct input/parent nodes**
A rule denotes a target variable and its parent variables. The joined table for such a given rule contains parents of the resident node (i.e., the target variable) that may be resident in another MFrag and need to be added as input nodes for the resident node. For example, we defined a set of rules in the Define World Model step (e.g., Rule 2: causal(VehicleType, DangerLevel)). In MTheory 5.1, for the target variable DangerLevel in the MFrag Situation, its parent VehicleType is defined in the MFrag Vehicle. The parent variable VehicleType should be an input node in the MFrag Situation. The following MFrag shows the updated result for the MFrag Situation using Rule 2.

MFragment 5.1: Situation

1 [C: Isa (rgn, REGION), Isa (t, TIME)]
2 [C: Isa (VID, VEHICLE)]
3 [R: DangerLevel (rgn, t)
4 [IP: VehicleType (VID)]
5 ]

The primary key for VehicleType is VID associated with the entity VEHICLE, so Isa (v, VEHICLE) is added in the updated MFrag Situation (MFragment 5.1).

(3) Construct context nodes

In this step, additional context nodes (other than Isa context nodes) are added to the updated MFrag. For this, we can use a joining script (e.g., SQL script 5.1) used for joining relations. In SQL script 5.1, there are conditions for joining. (e.g., Situation.rgn = Location.Location, Situation.t = Location.t, and Vehicle.VID = Location.v). These conditions are represented as context nodes. For example, the condition Situation.rgn = Location.Location can be a context node rgn = Location(v, t1), where the ordinary variable rgn comes from the primary key rgn in the relation Situation, the first v comes from the relation Location, and the second t1 comes from the relation Location. Note that
although the primary key $t$ for the attribute $\text{DangerLevel}$ and the primary key $t$ for the attribute $\text{Location}$ are same, they must be given different ordinary variable names in the context nodes, because they refer to different entities. For example, $\text{Location}(v, t)$ for the attribute $\text{Location}$ can be changed to $\text{Location}(v, t1)$. The condition $\text{Situation}.t = \text{Location}.t$ can be a context node $t = t1$, where the first $t$ comes from the relation $\text{Situation}$ associated with the attribute $\text{DangerLevel}$ and the second $t1$ comes from the relation $\text{Location}$ associated with the attribute $\text{Location}$. From the above process, the following script can be developed.

<table>
<thead>
<tr>
<th>MFragment 5.2: Situation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
</tbody>
</table>

The primary key for the attribute $\text{Location}$ are $v$ and $t$, so the $\text{Isa}$ context nodes $\text{Isa}(v, \text{VEHICLE})$ and $\text{Isa}(t1, \text{TIME})$ are added to MFragment 5.2.

(4) Refine context nodes

In MFragment 5.2, we notice that two equal-context nodes (i.e., [C: $t = t1$] in Line 5 and [C: $\text{VID} = v$] in Line 6) indicate conditions that entities must be equal. Consequently, the equal-context node indicates that they are the same entity. The above script can be simplified by removing ordinary variables sharing the same entity and equal-context nodes.
The Learn Reasoning Model step applies MTheory learning from relational data. In this research, we focus on MEBN parameter learning given a training dataset $D$ in RM and an initial MTheory $M$. Before introducing MEBN parameter learning, some definitions are introduced in the following sub-sections.

### 5.2.3.2 Definitions for Class Local Distribution and Instance Local Distribution

We introduced Definition 2.2 (M_frag), Definition 2.3 (M_node), and Definition 2.4 (MTheory) for MEBN in Chapter 2. An MTheory is composed of a set of MFrags $F$ on the MTheory (i.e., $M = \{F_1, F_2, \ldots, F_n\}$) conditions (e.g., no-cycle, bounded causal depth, unique home MFrags, and recursive specification condition [Laskey, 2008]) in Chapter 2. An M_frag $F$ is composed of a set of MNodes $N$ and a graph $G$ for $N$ (i.e., $F = \{N, G\}$). An MNode is composed of a function or predicate of FOL $ff$ and a class local distribution $(L)$ (i.e., $N = \{ff, L\}$).

A CLD specifies how to define local distributions for instantiations of the MNode. The following CLD 5.1 and ILD 5.1 show illustrative examples for a CLD (Class Local Distribution) and an ILD (Instance Local Distribution), respectively (recall that these examples were discussed in Chapter 2). CLD 5.1 defines a distribution for the degree of danger in a region. If there are no tracked vehicles, the default probability distribution described in Line 6 is used. The default probability distribution in a CLD is used for ILDs
generated from the CLD, when no nodes meet the conditions defined in the M_frag for parent nodes.

This CLD is composed of a class parent condition $CPC_i$ and a class-sub-local distribution $CSD_i$. A CPC indicates a condition whether a CSD associated with the CPC is valid. The CSD (class-sub-local distribution) is a sub-probability distribution which specifies how to define a local distribution under a condition in an RV derived from an MNode. For example, the first line in CLD 5.1 is $CPC_1$ which indicates a condition of the first class-sub-local distribution $CSD_1$. In this case, the condition means that “if there is an object whose type is Tracked”. If this is satisfied (i.e., $CPC_1$ is valid), then $CSD_1$ is used. A CPC can be used for a default probability distribution. In such a case, it is called a default CPC specified by $CPC_d$ and also the CSD associated with $CPC_d$ is called a default CSD, $CSD_d$.

CLD 5.1 [Discrete CLD]: DangerLevel($rgn, t$)

1. $CPC_1$: if some $v$ have (VehicleType = Tracked) [ 
2. $CSD_1$: $High = \theta_{1,1}, Low = \theta_{1,2}$ 
3. $CPC_2$: ] else if some $v$ have (VehicleType = Wheeled) [ 
4. $CSD_2$: $High = \theta_{2,1}, Low = \theta_{2,2}$ 
5. $CPC_d$: ] else [ 
6. $CSD_d$: $High = \theta_{d,1}, Low = \theta_{d,2}$ ]

For this case, we assume that the MNode contains two states ($High$ and $Low$) and the discrete parent the RV $VehicleType(v)$ has two states ($Tracked$ and $Wheeled$). The pair of $CSD_1$ and $CSD_1$ (in Line 1 and 2) is for $VehicleType(v) = Tracked$. The pair of $CSD_2$ and $CSD_2$ (in Line 3 and 4) is for $VehicleType(v) = Wheeled$. The pair of $CPC_d$ and $CSD_d$ (in Line 5 and 6) is for a default distribution.
The following ILD 5.1 shows the ILD derived from the above CLD given one region entity \( \text{region1} \) and one vehicle entity \( v1 \). Like the CLD, the ILD is composed of an instance parent condition \( \text{IPC}_i \) and an instance-sub-local distribution \( \text{ISD}_i \). The IPC indicates a condition whether the ISD associated with the IPC is valid. The ISD is a probability distribution which is defined in an ILD of a random variable.

### ILD 5.1: ILD with one region and one vehicle

\[
\begin{align*}
\text{P}(\text{DangerLevel}_\text{region1} | \text{VehicleType}_v1) & \hspace{1cm} \\
\text{IPC}_1: & \text{ if (VehicleType}_v1 \Rightarrow \text{Wheeled ) } \\
\text{ISD}_1: & \text{ High } = \theta_1.; \text{ Low } = \theta_1.; \\
\text{IPC}_2: & \text{ else if (VehicleType}_v1 \Rightarrow \text{Wheeled ) } \\
\text{ISD}_2: & \text{ High } = \theta_2.; \text{ Low } = \theta_2.; \\
\end{align*}
\]

Now, consider a situation in which there is a region containing no vehicles. In this case, the default probability distribution in CLD 5.1 is used for such an ILD (i.e., ILD 5.2), because all conditions associated with parent nodes (i.e., CPC1 and CPC2 in CLD 5.1) are not valid.

### ILD 5.2: Default ILD with one region without any vehicle

\[
\begin{align*}
\text{P}(\text{DangerLevel}_\text{region1}) & = \\
\text{IPC}_1: & \{ \\
\text{ISD}_1: & \text{ High } = \theta_1.; \text{ Low } = \theta_1.; \\
\text{IPC}_2: & \{ \\
\text{ISD}_2: & \text{ High } = \theta_2.; \text{ Low } = \theta_2.; \\
\end{align*}
\]

Now, we introduce the ILD formally.

**Definition 5.1 (Instance Local Distribution)** An instance local distribution \( L^I \) for a random variable \( rv \) in a Bayesian network (Definition 2.1) is a function defining the probability distribution for the random variable \( rv \). It consists of a set of pairs \( (\text{IPC}_i, \text{ISD}_i) \) of an instance parent condition \( \text{IPC}_i \) and an instance-sub-local distribution \( \text{ISD}_i \), and a
rule for mapping an instance parent condition $IPC_i$ into an instance-sub-local distribution $ISD_i$.

An ILD is derived from a CLD given entity information. For example, ILD 5.1 in the above example is derived from CLD 5.1 given the three vehicle entities. Once an ILD is derived from a CLD, the ILD contains a set of pairs $(IPC_i, ISD_i)$. In the following, the CLD is introduced formally.

**Definition 5.2 (Class Local Distribution)** A class local distribution (CLD) $L^C$ (or simply $L$) for an MNode (Definition 2.3) is a function defining uncertainty for the MNode. It consists of a set of pairs $(CPC_i, CSD_i)$ of a class parent condition $CPC_i$ and a class-sub-local distribution $CSD_i$, and a rule for mapping it $(CPC_i, CSD_i)$ into an instance local distribution (ILD) $L^I$.

A class local distribution defines a general rule for specifying distributions for instantiations of its random variables for specific entities. A CLD can refer to a parameterized family of distributions (e.g., normal distribution, categorical distribution). In this case, the CLD definition includes a specification of the parameters. For example, a class local distribution $CLD_1$ can represent a set of normal distributions for CSDs in $CLD_1$ and this $CLD_1$ can be called a normal distribution CLD (i.e., $\text{TYPE}(CLD_1) = \text{Normal Distribution CLD}$). A CSD can contain a set of parameters for its distribution. For example, $CSD_1$ in CLD 5.1 is a distribution containing two parameters $\theta_{1.1}$ and $\theta_{1.2}$. We can think of a parameter function returning a set of parameters from a CSD (i.e., $\Theta(CSD_1) = \{\theta_{1.1}, \theta_{1.2}\}$).
CLDs may be discrete or continuous. According to combination of the CLD types and parent CLD types, there are six categories for a CLD: (1) a discrete CLD with discrete parents, (2) a discrete CLD with continuous parents, (3) a discrete CLD with both discrete and continuous parents, (4) a continuous CLD with discrete parents, (5) a continuous CLD with continuous parents, and (6) a continuous CLD with both discrete and continuous parents.

When a node has discrete parent nodes, influence counts (IC), the number of distinct entities in CPC, can be used to define a CLD. For example, we can think of a CLD for the MNode DangerLevel(rgn, t) described by CLDL as shown the following. CLD 5.2 is the case of a discrete CLD with discrete parents.

| CLD 5.2 [Inverse Cardinality Average]: DangerLevel (rgn, t) |
|-------------|-------------|
| 1 \( \text{CPC}_1: \text{if some } v \text{ have (VehicleType = Tracked) } [\) |
| 2 \( \text{CSD}_1: \text{High} = 1 - \theta/(\text{CARDINALITY}(v) + 1), \text{Low} = 1 - \text{High} \) |
| 3 \( \text{CPC}_d: \text{else } [\) |
| 4 \( \text{CSD}_d: \text{High} = 0.1, \text{Low} = 0.9 \) |

We name CLD 5.2 an Inverse Cardinality Average. Thus, the type of the class local distribution is the inverse cardinality average (i.e., \( \text{TYPE(CLD 5.2)} = \text{Inverse Cardinality Average CLD} \)). CLD 5.2 consists of two CSDs (\( \text{CSD}_1 \) and \( \text{CSD}_d \)). \( \text{CSD}_1 \) contains a parameter \( \theta \), where \( 0 < \theta < 1 \), as shown CLD 5.2. CLD 5.2 represents probabilistic knowledge of how the level of danger of a region is measured depending on the vehicle type of detected objects. For example, if in a region there are many tracked vehicles (e.g., Tanks), the danger level of the region at a certain time will be high. The influence counting (IC) function \( \text{CARDINALITY(obj)} \) returns the number of tracked vehicles from parents nodes. If there are many tracked vehicles, the probability of the
state *High* increases. If there is no tracked vehicles, the default probability distribution (i.e., $CSD_d$) described in Line 4 is used for the CLD of the MNode DangerLevel($rgn$, $t$). Thus, it indicates a situation in peace time.

Here is another CLD example. CLD 5.3 shows the case of the continuous CLD with hybrid parents. For this case, we assume that there is an MNode Range($v$, $t$) which is a parent node of the MNode DangerLevel($rgn$, $t$) and means a range between the region $rgn$ and the vehicle $v$ at a time $t$.

<table>
<thead>
<tr>
<th>CLD 5.3 [Hybrid Cardinality]: DangerLevel($rgn$, $t$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CPC$_1$: if some $v$ have (VehicleType = Tracked) [</td>
</tr>
<tr>
<td>2 CSD$_1$: CARDINALITY($v$) / average( Range ) + NormalDist($\Theta$, 5)</td>
</tr>
<tr>
<td>3 CPC$_d$: ] else [</td>
</tr>
<tr>
<td>4 CSD$_d$: NormalDist(10, 5)</td>
</tr>
<tr>
<td>5 ]</td>
</tr>
</tbody>
</table>

The meaning of CLD 5.3 is that the degree of the danger in the region is the number of tracked vehicles divided by an average of the ranges of vehicles and then plus a normally distributed error with a mean of $\Theta$ and a variance of 5. If there is no tracked vehicles, the default probability distribution, NormalDist(10, 5), described in Lines 4 is used. If there are continuous parents, various numerical *aggregating* (AG) functions (e.g., *average*, *sum*, and *multiply*) can be used. For example, if there are three continuous parents Range1, Range2, and Range3, the numerical aggregating functions average, sum, and multiply will construct three IPDs $IPD_1 = (Range1 + Range2 + Range3)/3$, $IPD_2 = (Range1 + Range2 + Range3)$, and $IPD_3 = (Range1 * Range2 * Range3)$, respectively.

The above CLDs 5.2 and 5.3 are based on an influence counting (IC) function for discrete parents and an aggregating (AG) function for continuous parents. Using such a
function is related to the aggregating influence problem (Section 7.4.1), which treats many instances from a parent RV.

The CLD 5.1 uses a very simple aggregation rule that treats all counts greater than zero as equivalent. In other words, a shared parameter in a CSD is learned from all instances of the parent RV with counts greater than zero. For example, with CLD 5.1, suppose that there are two cases: In Case 1, there is one tracked vehicle. And in Case 2, there are two tracked vehicles. For Case 1, one VehicleType RV is constructed and CSD₁ (Line 1) in CLD 5.1 is used for the parameter of the distribution for the DangerLevel. For Case 2, two VehicleType RVs are constructed and also CSD₁ (Line 1) in CLD 5.1 is used for the parameter of the distribution for the DangerLevel, although there are two tracked vehicles. Thus, the shared parameter (i.e., \( High = \theta_{1,1} \) and \( Low = \theta_{1,2} \)) for CSD₁ in CLD 5.1 is used regardless of the number of the parent instances (i.e., one vehicle in Case 2, two vehicles in Case 2, and so on). In the following sections, we use such a simple aggregation rule for MEBN parameter learning.

5.2.3.3 Dataset for Class-Sub-Local Distribution (CSD)

A CLD can contain class parent conditions (CPC). Each CPC requires its own dataset to be learned to a class-sub-local distribution CSD associated with the CPC. For example, CLD 5.1 contains three CPCs (CPC₁, CPC₂, and CPC₄). Each CPC requires its own dataset. Such a dataset can be classified by two categories: (1) A dataset for a common CPC (e.g., CPC₁ and CPC₂) and (2) a dataset for a default CPC (e.g., CPC₄). In this section, we introduce how to get the dataset for a common CPC first. Then we present how to get the dataset for a default CPC.
Table 5.4 CSD dataset

<table>
<thead>
<tr>
<th>CPC College (GC)</th>
<th>Case</th>
<th>VehicleType</th>
<th>Location.x</th>
<th>Location.y</th>
<th>Location.z</th>
<th>Location.t</th>
<th>Situation.t</th>
<th>Situation.rgn</th>
<th>DangerLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPC1 (GC1)</td>
<td>1</td>
<td>Tracked</td>
<td>Vehicle13</td>
<td>Time18</td>
<td>Region6</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Tracked</td>
<td>Vehicle15</td>
<td>Time21</td>
<td>Region7</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Tracked</td>
<td>Vehicle17</td>
<td>Time24</td>
<td>Region8</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Tracked</td>
<td>Vehicle19</td>
<td>Time27</td>
<td>Region9</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Tracked</td>
<td>Vehicle1</td>
<td>Time2</td>
<td>Region0</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Tracked</td>
<td>Vehicle2</td>
<td>Time5</td>
<td>Region1</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Tracked</td>
<td>Vehicle4</td>
<td>Time8</td>
<td>Region2</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>Tracked</td>
<td>Vehicle5</td>
<td>Time8</td>
<td>Region2</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>Tracked</td>
<td>Vehicle6</td>
<td>Time11</td>
<td>Region3</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>Tracked</td>
<td>Vehicle7</td>
<td>Time11</td>
<td>Region3</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>Tracked</td>
<td>Vehicle8</td>
<td>Time14</td>
<td>Region4</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>Tracked</td>
<td>Vehicle9</td>
<td>Time14</td>
<td>Region4</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPC2 (GC2)</td>
<td>5</td>
<td>Wheeled</td>
<td>Vehicle21</td>
<td>Time30</td>
<td>Region10</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Wheeled</td>
<td>Vehicle23</td>
<td>Time33</td>
<td>Region11</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Wheeled</td>
<td>Vehicle0</td>
<td>Time2</td>
<td>Region0</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Wheeled</td>
<td>Vehicle3</td>
<td>Time5</td>
<td>Region1</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>Wheeled</td>
<td>Vehicle10</td>
<td>Time17</td>
<td>Region5</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>Wheeled</td>
<td>Vehicle11</td>
<td>Time17</td>
<td>Region5</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3 is a joined dataset for the common CPC (i.e., CPC1 and CPC2). It can be sorted according to each CPC as shown in Table 5.4. For example, the CPC1 in CLD 5.1 defines that it is only valid if a case contains a tracked vehicle. Therefore, by CPC1, we can sort the joined dataset in Table 5.3. Thus, the cases 1, 2, 3, 4, 8, 9, 11, 12, 13, 14, 15, and 16 are selected for CSD1, while other cases are used for CSD2 (Table 5.4). We call this dataset a CSD dataset.

**Definition 5.4 (CSD Dataset)** Let there be a dataset $D = \{C_1, C_2, \ldots, C_n\}$, where $C_i$ is each case (or row), and a CLD $L = \{(CPC_1, CSD_1), (CPC_2, CSD_2), \ldots, (CPC_m, CSD_m)\}$. A **CSD Dataset (CD)** is a dataset which is grouped by matching each class parent condition CPC of $L$ and each case $C_i$ in $D$. The set of grouped cases $GC_j = \{C_1, C_2, \ldots, C_i\}$ is assigned to a corresponding class parent condition CPC.
For an RV, if there are cases for which the conditions associated with the parent RVs are not satisfied, the dataset for a default CPC is required. The dataset for the default CPC (i.e., CPC_d) can be obtained by excluding the joined dataset from the original dataset. This is necessary because we need a dataset which doesn’t include cases for which the conditions associated with the parent RVs are satisfied. For example, in Table 5.2, there is an original dataset for the DangerLevel RV (i.e., the dataset in the relation Situation). Table 5.3 shows a joined dataset associated with CPC_1 and CPC_2. The dataset for the default CPC_d can be derived by subtracting the joined dataset (Table 5.3) from the original dataset (Table 5.2). For example, the following is a SQL script to extract the default dataset for the DangerLevel RV from the joined dataset.

```
SQL script 5.2: SQL script for the default dataset of the DangerLevel RV
1   SELECT
2       Situation.rgn, Situation.t, Situation.DangerLevel
3   FROM Situation
4   WHERE NOT EXISTS (  
5       SELECT *  
6       FROM Location, Vehicle  
7       WHERE  
8           Situation.rgn = Location.Location &&  
9           Situation.t = Location.t &&  
10          Vehicle.VID = Location.v  
11 )
```

The dataset for the DangerLevel RV comes from the relation Situation (Line 3). When the dataset is selected, there is a condition (Line 4) in which the dataset should not include a joined dataset derived by Line 5~11. Using this script, the default dataset for the DangerLevel RV is obtained and means the danger level at a certain region, where there is no vehicle.
In the following subsections, a training dataset $D$ means the CSD dataset for a certain CLD.

5.2.3.4 Parameter Learning

In this section, we introduce a parameter learning method to estimate parameters of a class local distribution $L$ given a training dataset $D$ (i.e., a CSD dataset). We can think of a basic type of CLD for a discrete case and a continuous case. For the discrete case, Dirichlet distribution can be used (Section 5.2.3.4.1), while for the continuous case, Conditional Gaussian distribution can be used (Section 5.2.3.4.2). We introduce parameter learning for these types. In Section 3.3.1, a predicate RV for MEBN was discussed. Learning the parameter of the distribution for such a predicate RV, corresponding to a Boolean RV with possible values true and false, from a relational database is discussed in Section 5.2.3.4.3.

5.2.3.4.1 Dirichlet distribution parameter learning

Details on Dirichlet distribution parameter learning can be found in Appendix A. Dirichlet distribution is commonly used because it is conjugate to the multinomial distribution. With a Dirichlet prior distribution, the posterior predictive distribution has a simple form [Heckerman et al., 1995][Koller & Friedman, 2009].

As an illustrative example of the Dirichlet distribution parameter learning for a CLD, we use CLD 5.1. Parameter learning for this CLD is to estimate CSD$_1$'s parameters ($\theta_{1,1}$ and $\theta_{1,2}$), and CSD$_2$'s parameters ($\theta_{2,1}$ and $\theta_{2,2}$), and CSD$_d$'s parameters ($\theta_{d,1}$ and $\theta_{d,2}$). To estimate these parameters, we can use the following predictive distribution using a Dirichlet conjugate prior, discussed in Appendix A. Equation 5.1 shows the
posterior predictive distribution for the value $x_k$ of the RV $X$ given a parent value $a$, the dataset $D$, and a hyperparameter $\alpha$ for the Dirichlet conjugate prior.

$$P(X = x_k \mid A = a, D, \alpha) = \frac{\alpha_{x_k|a} + C[x_k, a]}{\sum_{q=1}^{N} (\alpha_{x_q|a} + C[x_q, a])}, \quad (5.1)$$

where a value $x_k \in \text{Val}(X)$, $a \in \text{Val}(\text{Pa}(X) = A)$, $C[x_q, a]$ is the number of times outcome $x_q$ in $X$ and its parent outcome $a$ in $A$ appears in $D$, a hyperparameter $\alpha = \{\alpha_{x_1|a}, \ldots, \alpha_{x_N|a}\}$, and $N = |\text{Val}(X)|$.

For the case of the CPC1 and CSD1, we can use the set of grouped cases GC1 in Table 5.4 as a training dataset. And CSD1 has two parameters $\theta_{1.1}$ (for High) and $\theta_{1.2}$ (for Low). For the parameters $\theta_{1.1}$, we can use Equation 5.1 such as $\theta_{1.1} = P(\text{DangerLevel} = \text{High} \mid \text{VehicleType} = \text{Tracked}, D = \text{GC1}, a)$, where $a = \{\alpha_{\text{High}|\text{Tracked}}, \alpha_{\text{Low}|\text{Tracked}}\}$. If there were previously one case for $\text{High}|\text{Tracked}$ and two cases $\text{Low}|\text{Tracked}$, $\alpha_{\text{High}|\text{Tracked}} = 1$ and $\alpha_{\text{Low}|\text{Tracked}} = 2$ are used. This approach uses again for the case of the CPC2 and CSD2. To learn the parameter for the CSDd, the default dataset discussed in Section 5.2.3.3 is required. The parameter $\theta_{d.1}$ and $\theta_{d.2}$ can be learned from the default dataset using Equation 5.1 as the case of the CPC1 and CSD1.

5.2.3.4.2 Conditional Gaussian distribution parameter learning

Parameters for conditional Gaussian distribution can be estimated using multiple-regression. In this section, we introduce parameter learning of a conditional linear Gaussian CLD using linear regression. The following CLD shows an illustrative example
of a conditional linear Gaussian CLD for the RV \( \text{Speed}_\text{RPT}(r, t) \). The CLD of the RV is a continuous CLD with hybrid parents (\( \text{MTI\_Condition} \) and \( \text{Speed} \)). In this case, we assume that the discrete parent RV \( \text{MTI\_Condition}(v, mti, t) \) has two states (\( \text{Good} \) and \( \text{Bad} \)) and the RV \( \text{Speed}(v, t) \) is continuous.

\[
\text{CLD 5.4 [Conditional Linear Gaussian]: Speed}_\text{RPT}(r, t)
\]

1 \text{CPC}_1: \text{if some } v.mti.t \text{ have } (\text{MTI\_Condition} = \text{Good}) [ \\
2 \text{CSD}_1: \theta_{1,0} + \theta_{1,1} \cdot \text{Speed} + \text{NormalDist}(0, \theta_{1,2}) \\
3 \text{CPC}_2: \text{if some } v.mti.t \text{ have } (\text{MTI\_Condition} = \text{Bad}) [ \\
4 \text{CSD}_2: \theta_{2,0} + \theta_{2,1} \cdot \text{Speed} + \text{NormalDist}(0, \theta_{2,2}) \\
5 \text{CPC}_d: \text{else } [ \\
6 \text{CSD}_d: \theta_{d,0} + \text{NormalDist}(0, \theta_{d,2}) \\
7 \]

Parameter learning for this CLD is to estimate \( \text{CSD}_1 \)'s parameters (\( \theta_{1,0}, \theta_{1,1} \) and \( \theta_{1,2} \)), \( \text{CSD}_2 \)'s parameters (\( \theta_{2,0}, \theta_{2,1} \) and \( \theta_{2,2} \)), and \( \text{CSD}_d \)'s parameters (\( \theta_{d,0}, \theta_{d,1} \) and \( \theta_{d,2} \)).

We can write this situation more formally. If \( X \) is a continuous node with \( n \) continuous parents \( U_1, \ldots, U_n \) and \( m \) discrete parents \( A_1, \ldots, A_m \), then the conditional distribution \( p(X \mid u, a) \) given parent states \( U = u \) and \( A = a \) has the following form:

\[
p(X \mid u, a) = \mathcal{N}(L^a(u), \sigma^a), \tag{5.2}
\]

where \( L^a(u) = m^a + b_1^a u_1 + \cdots + b_n^a u_n \) is a linear function of the continuous parents, with intercept \( m^a \), coefficients \( b_i^a \), and standard deviation \( \sigma^a \) that depends on the state \( a \) of the discrete parents. Given \( \text{CPC}_j \) (i.e., given the state \( a_j \)), estimating the parameters the intercept \( m^{(a_j)} \), coefficients \( b^{(a_j)}_i \), and standard deviation \( \sigma^{(a_j)} \) corresponds to estimating the \( \text{CSD}'s \) parameters \( \theta_{j,0}, \theta_{j,1} \) and \( \theta_{j,2} \), respectively.
The following shows multiple linear regression which is modified from [Rencher, 2003]. $L^{(a)}(u)$ can be rewritten, if we suppose that there are $k$ observations (Note that for one CSD case, we can omit the state $a$, because we know it).

\[
L_i(u) = m + b_1 u_{i1} + \cdots + b_n u_{in} + \sigma_i, i = 1, \ldots, k
\]  

(5.3)

where $i$ indexes the observations. For convenience, we can write the above equation more compactly using matrix notation:

\[
l = Ub + \sigma,
\]  

(5.4)

where $l$ denotes a vector of instances for the observations, $U$ denotes a matrix containing all continuous parents in the observations, $b$ denotes a vector containing an intercept $m$ and a set of coefficients $b_i$, and $\sigma$ denotes a vector of regression residuals. The following equations show these variables in forms of vectors and a matrix.

\[
l = \begin{bmatrix}
L_1(u) \\
L_2(u) \\
\vdots \\
L_k(u)
\end{bmatrix}
U = \begin{bmatrix}
1 & u_{11} & \cdots & u_{1n} \\
1 & u_{21} & \cdots & u_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
1 & u_{k1} & \cdots & u_{kn}
\end{bmatrix}
\begin{bmatrix}
m \\
b_1 \\
b_2 \\
\vdots \\
b_k
\end{bmatrix}
\sigma = \begin{bmatrix}
\sigma_1 \\
\sigma_2 \\
\vdots \\
\sigma_k
\end{bmatrix}
\]  

(5.5)

From the above settings, we can derive an optimal vector for the intercept and the set of coefficients $\hat{b}$. 

139
\[
\hat{\mathbf{b}} = (\mathbf{U}^T \mathbf{U})^{-1} \mathbf{U}^T \mathbf{l}
\]

(5.6)

Also, we can derive the optimal standard deviation \(\hat{\sigma}\) from the above linear algebra term [Rencher, 2003].

\[
\hat{\sigma} = \sqrt{\frac{(\mathbf{l} - \mathbf{U} \hat{\mathbf{b}})^T (\mathbf{l} - \mathbf{U} \hat{\mathbf{b}})}{k - n - 1}}
\]

(5.7)

Using the above equations, the optimal parameters can be estimated. For CPC\(_1\) in CLD 5.4, CSD\(_1\) can be the following.

\[
p(X \mid \text{Speed, MTI Condition} = \text{Good}) = N(\theta_{1,0} + \text{Speed} \times \theta_{1,1}^{(\text{Good})}, \theta_{1,2}^{(\text{Good})}).
\]

In this section, we discussed how to learn parameters for the conditional linear Gaussian CLD using linear regression. For a conditional nonlinear Gaussian CLD, we can use nonlinear regression. In this section, we didn't consider incremental parameter learning for the conditional linear Gaussian CLD. For this, we can Bayesian regression [Press, 2003], which is more robust to overfitting than the traditional multiple-regression.

5.2.3.4.3 Parameter Learning for the Distribution of the Predicate/Boolean RV

The parameter of the distribution for a predicate or Boolean RV (Section 3.3.1) can be learned from a relational database. To introduce predicate RV parameter learning, the following relations in Table 5.5 as an illustrative example are used to learn the
parameter of the distribution for a predicate RV *Communicate*. The following table contains three relations *Vehicle*, *Communicate*, and *Meet*. The relation *Communicate* means that two vehicles communicate with each other by exchanging radio waves. The relation *Meet* means that two vehicles meet each other by locating in close proximity to each other.

Table. 5.5 Communicate Relation and Meet Relation

<table>
<thead>
<tr>
<th>Vehicle</th>
<th>Communicate</th>
<th>Meet</th>
</tr>
</thead>
<tbody>
<tr>
<td>VID</td>
<td>VID1/Vehicle</td>
<td>VID2/Vehicle</td>
</tr>
<tr>
<td>v1</td>
<td>v1</td>
<td>v1</td>
</tr>
<tr>
<td>v2</td>
<td>v2</td>
<td>v2</td>
</tr>
<tr>
<td>v3</td>
<td>v3</td>
<td>v3</td>
</tr>
<tr>
<td>v4</td>
<td>v4</td>
<td>v4</td>
</tr>
</tbody>
</table>

The above relationship relations (i.e., *Communicate* and *Meet*) show true cases for predicates. For example the relation *Communicate* contains the true cases \{\{v1, v2\}, \{v2, v3\}, \{v3, v4\}\}. However, relationship relations do not explicitly represent false cases for the predicates. By converting the above relations to the following relations, we can see the false cases explicitly. This conversion is justified by CWA. Thus, if a case in a relationship relation is not true, it is assumed to be false.

The relation *Vehicle* in Table 5.6 contains four vehicle entities (v1 ~v4). These entities can be used to develop possible combinations of two vehicles interacting with each other as shown data in the first and second column in the relations *Communicate* and *Meet* (i.e., \{\{v1, v2\}, \{v1, v3\}, \{v2, v3\}, \{v1, v4\}, \{v2, v4\}, \{v3, v4\}\}). The relation
Communicate means the possible combinations between the two vehicles communicating with each other and contains an attribute Communicate indicating whether the two vehicles are communicated (True) or not (False). From data in the relation Communicate in Table 5.5, the true cases for the attribute Communicate in the relation Communicate in Table 5.6 can be derived. The true cases for the attribute Meet in the relation Meet in Table 5.6 are also derived using the same approach. Now, as we can see Table 5.6, the relations Communicate and Meet explicitly contain the true and false cases for the attributes Communicate and Meet, respectively.

<table>
<thead>
<tr>
<th>Vehicle</th>
<th>Communicate</th>
<th>Meet</th>
</tr>
</thead>
<tbody>
<tr>
<td>VID</td>
<td>VDI/Vehicle</td>
<td>VD2/Vehicle</td>
</tr>
<tr>
<td>v1</td>
<td>v1</td>
<td>v2</td>
</tr>
<tr>
<td>v2</td>
<td>v1</td>
<td>v3</td>
</tr>
<tr>
<td>v3</td>
<td>v1</td>
<td>v4</td>
</tr>
<tr>
<td>v4</td>
<td>v2</td>
<td>v3</td>
</tr>
<tr>
<td></td>
<td>v2</td>
<td>v4</td>
</tr>
<tr>
<td></td>
<td>v3</td>
<td>v4</td>
</tr>
</tbody>
</table>

To construct the set of combination between the four vehicles in the relation Vehicle, we can use the following script.

**SQL script 5.3**: Combination between the four vehicles

```
1  CREATE TABLE
2     All_Vehicles AS
3     ( SELECT
4         t1.VID AS VID1,
5         t2.VID AS VID2
```
The above script generates a new relation called \textit{All\_Vehicles}. The dataset for the relation \textit{All\_Vehicles} contains \{\{v1, v2\}, \{v1, v3\}, \{v2, v3\}, \{v1, v4\}, \{v2, v4\}, \{v3, v4\}\}. The above script selects the set of combination between the four vehicles occurring only once. To generate the dataset for the relation \textit{Communicate} in Table 5.6, we can use the following script.

\textbf{SQL script 5.4: For the Relation \textit{Communicate} in Table 5.6}

\begin{verbatim}
1 SELECT DISTINCT t2.VID1, t2.VID2,
2    (SELECT
3       IF(t1.VID1 = t2.VID1 && t1.VID2 = t2.VID2, "True", "False")
4    ) AS Communicate
5 FROM All\_Vehicles t1, Communicate t2
\end{verbatim}

The above script compares data between the relations \textit{All\_Vehicles} and \textit{Communicate}. If there is a same primary key between them, a value \textit{True} is assigned to an attribute \textit{Communicate}. If not, a value \textit{False} is assigned to the attribute \textit{Communicate}.

To generate the dataset in the relation \textit{Meet} in Table 5.6, we can use the same approach.

For the relations in Table 5.6, we assume the following CLD 5.5 in which the meeting between two vehicles may influence the event for communication between the vehicles (i.e., $P(\text{Communicate} \mid \text{Meet})$). In CLD 5.5, CPC\(_1\) (Line 1) indicates a condition where two vehicles meet. CPC\(_2\) (Line 3) indicates a condition where two vehicles don’t meet. For example, CSD\(_2\) (Line 4) represents the probability that two vehicles $VID1$ and $VID2$ communicate with each other in the situation where the two vehicles are not nearby.

\textbf{CLD 5.5 [Predicate RV]: Communicate ($VID1$, $VID2$)}

\begin{verbatim}
1 CPC1: if some $VID1.VID2$ have (Meet = True) [ \\
2 CSD1: True = $\Theta_{1,1}$, False = $\Theta_{1,2}$
\end{verbatim}
To learn parameters in CLD 5.5, CSD datasets for CPC₁ and CPC₂ are required. To generate such datasets, the processes in Section 5.2.3.1 *Map to Reasoning Model* can be used. For example, a joined dataset between the relations *Communicate* and *Meet* is generated by matching same vehicle entities in both relations. The joined dataset contains four attributes *VID₁*, *VID₂*, *Communicate*, and *Meet* (e.g., \{v₁, v₂, True, True\}, ..., \{v₃, v₄, True, False\}). Then, parameter learning as described in Section 5.2.3.4 *Parameter Learning* is used to construct the parameters in CLD 5.5 (i.e., \(P(Communicate \mid Meet)\)).

### 5.2.4 Test Reasoning Model

In the *Test Reasoning Model* step, a learned reasoning model is evaluated to determine whether to accept it. The accepted reasoning model is output as a final result in this step. This step is decomposed into two sub-steps (Fig. 5.7): (1) an *Experiment Reasoning Model* step and (2) an *Evaluate Experimental Results* step.

![Figure 5.7 Test Reasoning Model](image-url)
5.2.4.1 Conduct Experiments for Reasoning Model

The Experiment Reasoning Model step tests the learned reasoning model using a test dataset. The test dataset can be generated from simulations, existing data and/or actual experiments. This experiment can consist of the following five steps. (1) The learned reasoning model is exercised on a test case from the test dataset. (2) The test dataset provides ground truth data to evaluate with a certain metric (e.g., the continuous ranked probability score) in the requirements defined in the Analyze Requirement step. (3) The metric is used to measure performance between results from the learned reasoning model and the ground truth data. (4) Steps 1-3 are repeated for all testing cases. (5) This step results in a result value integrating all measured values (e.g., an average of the continuous ranked probability scores).

5.2.4.2 Evaluate Experimental Results

In the Evaluate Experimental Results step, the performance of estimation and prediction for the learned reasoning model is assessed by the performance criteria in the requirements defined in the Analyze Requirement step (e.g., an average of the continuous ranked probability scores < 0.001). If the measured value satisfies the criteria, the learned reasoning model is accepted and this step results in the learned reasoning model. If the requirement is not satisfied, we can return to the previous steps to improve the performance of the learned reasoning model.

5.3 Conclusion

We introduced a MEBN learning framework, called HMLP, which includes three components (MEBN-RM, the PSAW-MEBN reference model, and MEBN learning). HMLP contained four steps ((1) Analyze Requirements, (2) Define World Model, (3)
Construct Reasoning Model, and (4) Test Reasoning Model. The following list shows their specific sub-steps.

1. **Analyze Requirements**
   - (1.1) Identify Goals
   - (1.2) Identify Queries/Evidence
   - (1.3) Define Performance Criteria

2. **Define World Model**
   - (2.1) Define Structure Model
   - (2.2) Define Rules
     - (2.2.1) Define Causal Relationships between RVs
     - (2.2.2) Define Distributions of RVs

3. **Construct Reasoning Model**
   - (3.1) Map to Reasoning Model
     - (3.1.1) Perform Entity-Relationship Normalization
     - (3.1.2) Perform MEBN-RM Mapping
     - (3.1.3) Update Reasoning Model using the Rules
       - (3.1.3.1) Join Relations
       - (3.1.3.2) Construct Input/Parent Nodes
       - (3.1.3.3) Construct Context Nodes
       - (3.1.3.4) Refine Context Nodes
   - (3.2) Learn Reasoning Model

4. **Test Reasoning Model**
   - (4.1) Conduct Experiments for Reasoning Model
     - (4.1.1) Test Reasoning Model from Test Dataset
     - (4.1.2) Measure Performance for Reasoning Model
   - (4.2) Evaluate Experimental Results

In (1) the **Analyze Requirements** step, there are three sub-steps: (1.1) the **Identify Goals** step, (1.2) the **Identify Queries/Evidence** step, and (1.3) the **Define Performance Criteria** step. The goals representing missions of the reasoning model is defined in (1.1). The queries, specific questions for which the reasoning model is used to estimate and/or predict answers, and the evidence, inputs used for reasoning, are defined in (1.2). Each query should include performance criteria (1.3) for evaluation of reasoning.

In (2) the **Define World Model** step, there are two sub-steps: (2.1) the **Define Structure Model** step and (2.2) the **Define Rules** step. The **Define Rules** step (2.2)
contains two sub-steps: (2.2.1) the Define Causal Relationships between RVs step and (2.2.2) the Define Distributions of RVs step. In (2.2.1), candidate causal relationships (e.g., influencing(A, B) and causal(A, B)) between RVs are specified using expert knowledge. In (2.2.2), a (conditional) local distribution of an RV is defined by expert knowledge.

In (3) the Construct Reasoning Model step, there are two sub-steps: (3.1) the Map to Reasoning Model step and (3.2) the Learn Reasoning Model step. The Map to Reasoning Model step (3.1) is composed of three sub-steps: (3.1.1) the Perform Entity-Relationship Normalization step, (3.1.2) the Perform MEBN-RM Mapping step, and (3.1.3) the Update Reasoning Model using the Rules step. Before applying MEBN-RM to a relational model, the relational model is normalized using Entity-Relationship Normalization (3.1.1). In (3.1.2), MEBN-RM is performed to construct an initial MTheory from the relational model. In (3.1.3), the initial MTheory is updated according to the rules defined in (2.2). The Update Reasoning Model using the Rules step (3.1.3) contains four sub-steps: (3.1.3.1) the Join Relations step, (3.1.3.2) the Construct Input/Parent Nodes step, (3.1.3.3) the Construct Context Nodes step, and (3.1.3.4) the Refine Context Nodes step. In (3.1.3.1), some relations are joined and an updated M_frag is created, if RVs in a rule are defined in different relations. The causal relationships for the RVs in the rule are defined in the updated M_frag through (3.1.3.2). In (3.1.3.2), if there is an input node, ordinary variables associated with the input node are defined in the updated M_frag. In (3.1.3.3), the context nodes associated with the RVs in the rule are defined in the updated M_frag. For this, the conditions (specified by a “Where”
conditioning statement in SQL) in a joining script, used for joining relations in (3.1.3.1), can be reused to construct such context nodes. In (3.1.3.4), ordinary variables sharing the same entity (e.g., Isa (t, TIME) and Isa (t1, TIME)) are converted into a single ordinary variable (e.g., Isa (t, TIME)). Then, equal-context nodes (e.g., t = t1) for such ordinary variables are removed. In (3.2) the Learn Reasoning Model step, a parameter learning algorithm performs to each RV in the updated MFragment using a training dataset to generate the parameter of the distribution for the RV.

In (4) the Test Reasoning Model step, there are two sub-steps: (4.1) the Conduct Experiments for Reasoning Model step and (4.2) the Evaluate Experimental Results step. In (4.1) there are two sub-steps: (4.1.1) the Test Reasoning Model from Test Dataset step and (4.1.2) the Measure Performance for Reasoning Model step. In (4.1.1), the learned MTheory from (3) the Construct Reasoning Model step is tested using a test dataset and (4.1.2) measured for performance between results from the learned MTheory and the ground truth data in the test dataset. In (4.2) the Evaluate Experimental Results step, whether the learned MTheory is accepted or not is decided using the performance criteria defined in (1.3).

In this research, some steps in HMLP are automated (e.g., (3.1.2) the Perform MEBN-RM Mapping step), while some other steps are not yet automated (e.g., (3.1.1) the Perform Entity-Relationship Normalization step) but could be automated. Also, some other steps (e.g., (1.1) the Identify Goals step) require aid from human (i.e., human centric). The following table shows the level of automation (i.e., Automated, Automatable, and Human centric) for each step in HMLP.
Table 5.7 Processing Method for Steps in HMLP

<table>
<thead>
<tr>
<th>Main Steps</th>
<th>Sub-steps</th>
<th>Processing Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Analyze Requirements</td>
<td>(1.1) Identify Goals</td>
<td>Human centric</td>
</tr>
<tr>
<td></td>
<td>(1.2) Identify Queries/Evidence</td>
<td>Human centric</td>
</tr>
<tr>
<td></td>
<td>(1.3) Define Performance Criteria</td>
<td>Human centric</td>
</tr>
<tr>
<td>(2) Define World Model</td>
<td>(2.1) Define Structure Model</td>
<td>Human centric</td>
</tr>
<tr>
<td></td>
<td>(2.2) Define Rules</td>
<td>Human centric</td>
</tr>
<tr>
<td>(3) Construct Reasoning Model</td>
<td>(3.1.1) Perform Entity-Relationship Normalization</td>
<td>Automatable</td>
</tr>
<tr>
<td></td>
<td>(3.1.2) Perform MEBN-RM Mapping</td>
<td>Automated</td>
</tr>
<tr>
<td></td>
<td>(3.1.3) Update Reasoning Model using the Rules</td>
<td>Automatable</td>
</tr>
<tr>
<td></td>
<td>(3.2) Learn Reasoning Model</td>
<td>Automated</td>
</tr>
<tr>
<td>(4) Test Reasoning Model</td>
<td>(4.1) Conduct Experiments for Reasoning Model</td>
<td>Automatable</td>
</tr>
<tr>
<td></td>
<td>(4.2) Evaluate Experimental Results</td>
<td>Automatable</td>
</tr>
</tbody>
</table>

For example, the (3.2) Learn Reasoning Model step is automated by the MEBN-RM mapping algorithm (Section 3.6). The (3.1.1) the Perform Entity-Relationship Normalization step is automatable by developing an algorithm converting from ordinary relations to the relations satisfying Entity-Relationship Normalization. The (1.1) Identify Goals step is human centric and require human support to perform it. Automatable steps can become automated steps by developing specific processes, algorithms, and software programs. We leave these as future studies.

We developed an HMLP Tool that performs MEBN-RM and the MEBN parameter learning. The HMLP Tool is a JAVA based open-source program that can be used to create an MTheory script from a relational data. This enables rapid development of an MTheory script by just clicking a button in the tool. This is available on Github\(^\text{10}\)

---

\(^\text{10}\) Github is a distributed version control system (https://github.com).
(see Appendix F). The current version of the HMLP Tool only supports context nodes containing the “IsA” property. We leave the use of context nodes representing sophisticated relational patterns as future work. Some specifications and limitations can be found in Appendix F.
CHAPTER SIX: APPLYING HMLP - USE CASES AND EXPERIMENT

In this section, we introduce three use cases (a critical infrastructure defense system, a maritime domain awareness system, and a smart manufacturing system) using HMLP and one experiment to evaluate the efficiency of HMLP as measured by the development time for a PSAW-MTheory.

6.1 Use Case 1 - HERALD
HERALD is a proof-of-concept PSAW system, designed to ward off attacks against critical infrastructure by means of early detection of threatening targets, identification of the targets, estimation of the target’s activities, and short-term prediction of future situations. The HERALD system is composed of a reasoning module, a control module, and a scenario simulator. The reasoning module infers current and future situations using a HERALD MTheory. The HERALD MTheory is designed by domain experts and learned using the proposed MEBN learning algorithm. The control module is used to query the situation of interest and display the results of reasoning. The HERALD scenario simulator generates ground truth data to learn the HERALD MTheory and to evaluate the designed and learned HERALD MTheories. Section 6.1.1 introduces the mission of the HERALD system with operational scenarios. Section 6.1.2 describes the HERALD system associated with sensor systems. Section 6.1.3 discusses the development performance for HMLP. Note that Appendix G describes the process of
HMLP for the development of the HERALD MTheory and inference results using the HERALD MTheory. Appendix I shows the HERALD scenario simulator and its database.

6.1.1 Critical Infrastructure Defense Situation

A civilian and military critical infrastructure (CI) can be exposed to various attacking threats. Before the attacks are conducted, harbingers for them should be monitored and detected to prevent the attacks. If a commander protecting the CI is able to understand the characteristics of the threats, the commander will have the ability to prosecute a response to the threats. However, it is often difficult for the commander to maintain sufficient situational awareness as well as to perform dynamic decision making under uncertainty. HERALD supports early detection of threats of concern. The primary aim of HERALD is to improve PSAW by fusing reports from several sensor systems (e.g., MTI (Moving Target Indicator), IMINT (Imagery Intelligence) Sensor, and GEOINT (Geospatial intelligence) Sensor). HERALD performs data fusion from these sensor systems and provides a PSAW capability for the commander.

Considering the design of a critical infrastructure defense situation, our proof-of-concept focuses on threats from terrorists and UAVs approaching critical infrastructure elements. The following subsections present an attacker (Red Team) and a defender (Blue Team) scenario developed to exercise the HERALD system. These scenarios were developed by a domain expert who reviewed real cases for the threats and used his knowledge to develop these scenarios.
6.1.1.1 Red (Intruder) Team Scenarios

HERALD presents two Red Team (intruders to CI) scenarios attacking a nuclear power plant in South Korea: (1) terrorist attack and (2) UAV attack. Both scenarios are performed in the same environmental conditions in which the weather is clear, it is evening, and the season is the end of the spring. (1) In the terrorist attack scenario, three armed terrorists head toward the nuclear power plant. The terrorists break through barriers (i.e., a fence and a wall) and infiltrate the plant. The terrorists occupy a critical infrastructure element at the plant, set up explosives at a vulnerable point of the critical infrastructure element, and evacuate from the area. (2) In the UAV attack scenario, three UAVs flying in formation at speed 74 MPH head toward a nuclear power plant. Each UAV contains a bomb. The UAVs move to 12 miles from the plant and scatter to distract the Blue team from surveillance. At 6 miles from the plant, each UAV flies into the plant and detonates its bomb by crashing into the plant.

6.1.1.2 Blue Team Scenarios

For the two Red Team scenarios, the Blue Team responds to the Red Team. A Blue Team command center uses sensors around the plant for PSAW. If targets are detected, the command center reacts to the targets (e.g., double-check using a mobile sensor). If the targets are identified more precisely, the command center defends the operations from the targets.

The Blue Team scenarios vary according to the result of target discrimination. Target discrimination is a process in which the target is assigned to an object type from a set of object types. As shown Fig. 6.1, target discrimination along the locations of the target can be classified into four categories: (1) Detection, (2) Classification, (3)
Recognition, and (4) Identification, or DCRI [Self et al., 2005]. In the detection layer, an unconfirmed object has been acquired (e.g., object of interest). Normally detection occurs at a further range than classification. In the classification layer, the unconfirmed object is determined by class (e.g., tracked vehicle, wheeled vehicle, rotary-wing aircraft, fixed-wing aircraft, human, and animal). However, in this layer, threat status of the target is not determined. In the recognition layer, threatening targets are determined (e.g., tank, armored car, armored unmanned aerial vehicles, helicopter gunship, and human carrying an object). In the identification layer, the specific model of the threatening target is determined (e.g., M1A2, AMZ Żubr, AH-64D, RQ-7, and human carrying a rifle). Each sensor system has its own these DCRI layers. Also, these DCRI layers can vary according to internal and external states of the sensor system observing targets and the targets’ signatures (e.g., size, shape, material, and temperature) distinguishing the targets from their background [Holst, 2000].

Figure 6.1 Ranges of a target from a critical infrastructure
The scenario of the *Blue Team* is as follows. (1) If a target enters the detection layer, the commander of the *Blue Team* observes the target’s activity closely. The target can either enter the classification layer or leave the detection layer. (2) If the target enters the classification layer, the commander of the *Blue Team* tries to identify whether the target is armed or not. The target can either enter the recognition layer or leave the classification layer. (3) If the target enters in the recognition layer and determined as a threatening target, the commander of the *Blue Team* prepares for an attack from the target. The target can move to some places, stay at a place, prepare for an attack, or attack the critical infrastructure. (4) If the target enters the identification layer, the commander of the *Blue Team* orders attack against the target. The target engages with the *Blue team*. Note that in this scenario, we don't consider a situation in which the blue team commander sends out some type of sensor to do further discrimination.

### 6.1.2 HERALD System

In 2014, Samsung Thales coined the design of a critical infrastructure defense system which is the predecessor of the HERALD system. Samsung Thales wanted to design and develop a next-generation system for critical infrastructure defense which would overcome problems and difficulties of existing systems. A domain expert from Samsung Thales advised on such a system and provided knowledge for design of HERALD (e.g., environmental factors, knowledge for sensors, and systems/system operations in the critical infrastructure defense situation). The system was a future system which we (our team and Samsung Thales) had never designed and developed before, so
we began with several assumptions. The following lists our assumptions for the development of the HERALD system.

**Assumption 1.** The HERALD system follows the quick adaptation paradigm.\(^{11}\) The initial system and reasoning model can be imperfect, however, they will be upgraded after the system is deployed.

**Assumption 2.** The initial HERALD system is designed to perform under simple conditions (e.g., weather condition, natural environment, and time).

**Assumption 3.** The operational scenarios of threatening entities and non-threatening entities derived from stakeholders are reasonable and realistic.

**Assumption 4.** The sensor systems and sensor data format derived from stakeholders are reasonable and realistic.

Although the initial HERALD system is developed under artificial situations, the system can be adapted to actual situations under the quick adaptation paradigm.

The HERALD system consists of (1) a reasoning module, (2) a control module, and (3) a scenario simulator (see Appendix H for the simulator). (1) HERALD reasons about current and future situations involving the *Blue* and *Red Team*. The reasoning module includes detection, object identification, activity identification, mission identification, and situation identification. Report data from sensor systems are transmitted to a HERALD reasoning module. The reasoning module uses the HERALD MTheory to reason about the situation. (2) The control module of HERALD is used to

---

\(^{11}\)The quick adaptation paradigm means that a system is designed to be adaptable quickly to a changing situation. The quick adaptation paradigm contains some assumptions: a system initially developed is imperfect and the system can evolve to address situational changes.
manage input parameters for the reasoning module (e.g., sensor reports) and display results from the reasoning module. (3) The HERALD scenario simulator generates ground truth data and sensor data for learning and evaluating.

![HERALD System in a C4I system](image)

**Figure 6.2 HERALD System in a C4I system**

The HERALD system aims to support a C4I (Command, Control, Communications, Computers, and Intelligence) system at a command center to estimate and predict situations using MEBN. Fig. 6.2 shows an illustrative example of HERALD system operation in the C4I system. The HERALD system fuses data from sensor systems that support the C4I system. The sensor systems are responsible for
preprocessing the sensor data using various techniques (e.g., clustering, filtering, and feature extraction) and then producing information. These information are stored in the database. These information are used to infer situations given targets’ evidence (e.g., reported target image and reported target temperature). To support the reasoning module, an MTheory, called the HERALD MTheory, is developed. The HERALD MTheory aims to estimate and predict situations occurring in the region, missions of entities in the region, activities of the entities, and values of attributes of the entities from the sensors’ data. The current version of the HERALD MTheory is base on a first-order Markov assumption\textsuperscript{12}. The reasoning module uses the HERALD MTheory to construct an SSBN (e.g., an SSBN in Appendix K) that is used to answer the PSAW questions discussed in Chapter 4 (see Appendix G.1 for the specific PSAW questions). The HERALD MTheory was learned using HMLP (see Appendix G) and was evaluated using the HERALD scenario simulator (see Appendix H).

6.1.3 Discussion of HMLP Performance
In this section, we discuss the efficiency of HMLP by comparing to a common approach for the development of an MTheory such as UMP-ST (discussed in Chapter 2) [Carvalho et al., 2016].

\textsuperscript{12} If an RV at time $n$ only depends on another RV at time $n-1$. 
Commonly, we use UMP-ST as the development framework for an MTheory. However, UMP-ST doesn't contain specific processes for MEBN learning and useful information for the development of a PSAW-MTheory. HMLP uses (PSAW-MEBN-REF (PSAW-MEBN reference model), MEBN-RM, and MEBN learning) to help reduce the development period for a PSAW-MTheory. Fig. 6.3 shows a comparison between UMP-ST and HMLP as an illustrative example. In the first step of HMLP, the PSAW-MEBN reference model (Chapter 4) provides a set of PSAW questions (see Appendix B), which enables us to grasp some ideas for what questions the PSAW-MTheory, which we are developing, should answer. This gets us comprehensive questions or requirements for PSAW quickly. In the second step of HMLP, the PSAW-MEBN reference model provides some guidance on groups of entities to be defined (i.e., Context, Reported Object, Observing Condition, Target Object, and Situation). The reference model also provides candidate entities (i.e., $T$, $OR$, $SR$, $TR$, and $RT$), random variables, and the causal
relationships among the random variables. This guidance may enhance the quality of the world model in HMLP as well as reduce the development time for them. In the third step, HMLP provides automation approaches (MEBN-RM and MEBN parameter learning). MEBN-RM supports the development of entities, random variables, and MFrags from a relational database for the world model. MEBN parameter learning can learn parameters for an MTheory given an initial MTheory and a training dataset, so the MTheory can be efficiently constructed as compared with a manual development approach. For example, before we had developed HMLP, we had an experience for the development of a HERALD MTheory from scratch (i.e., without MEBN-RM, the PSAW-MEBN reference model, and MEBN parameter learning). At the time, to develop one RV, a domain expert provided domain knowledge to construct values and parameters for the RV. When we could not have specific parameters for the RV, we researched to find theories related to the RV. The parameters for the RV, then, could be derived the theories. At least one day was spent on each RV. For some RVs, more than a week was required to develop the distribution. To analyze the efficiency of HMLP, we consider an illustrative example using these assumptions for the time required to develop distributions. For example, the HERALD MTheory (MTheory I.1 in Appendix I) contains 31 RVs. To develop distributions for these 31 RVs, we would need at least 31 days according to our assumptions. In other words, UMP-ST requires at least 31 days to develop 31 RVs. If we assume that there are datasets for these 31 RVs, these 31 days can be reduced significantly. In our research, we spent less than five days to learn the HERALD MTheory from a training dataset. In the task, data pre-processing like preparation of
dataset was included. Table 6.1 shows the development period comparison between UMP-ST and HMLP. The numbers in the table, which means working days, are estimated numbers with the assumption that developing the distribution for one RV takes at least one day.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>UMP-ST (Manual)</td>
<td>40</td>
<td>20</td>
<td>31</td>
<td>1</td>
<td>92</td>
</tr>
<tr>
<td>HMLP (Semi-automatic)</td>
<td>5</td>
<td>1</td>
<td>66</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Overall, when a training dataset is available, HMLP can accelerate the modeling of MTheories for PSAW as compared to the traditional manual MEBN modeling approach.

6.2 Use Case 2 - PROGNOS

In Chapter 2, we introduced a modeling framework for Probabilistic Ontologies, called UMP-ST [Carvalho et al., 2016]. An example of using UMP-ST was the development of a probabilistic ontology to support PROGNOS (PRobabilistic OntoloGies for Net-Centric Operational Systems) [Carvalho, 2011][Costa et al., 2012], a system that supports Maritime Domain Awareness (MDA). The PROGNOS probabilistic ontology provides semantically aware uncertainty management to support fusion of heterogeneous input and probabilistic assessment of situations to improve MDA. However, manually developing and maintaining a probabilistic ontology is a labor-
intensive and insufficiently agile process. Greater automation through a combination of reference models and machine learning methods may enhance agility for the development of MDA systems. For this reason, we suggested HMLP in Chapter 5. In previous work, we used UMP-ST to develop the PROGNOS probabilistic ontology (PO). This section presents an extended PROGNOS PO developed using HMLP. The contribution of this research is to introduce the extended PROGNOS PO and present a comparison between two processes (UMP-ST and HMLP) to evaluate the performance of HMLP.

6.2.1 Introduction

PROGNOS is a proof-of-concept system to support MDA. The existing system for MDA (e.g., US Navy's Net-Centric infrastructure, FORCENet) is used to fuse lower-level multi-sensor data, analyze the fused data by human analysts, and support decision-making for naval operations. However, the era of big data requires greater automation. The PROGNOS PO [Carvalho et al., 2010] supports ingestion of lower-level data, fusion of heterogeneous input, and probabilistic assessment of situations to improve MDA. PROGNOS aims especially to identify threatening targets (e.g., terrorist-ships).

In previous work, UMP-ST was used to develop the PROGNOS PO. This research presents an extended PROGNOS PO developed using HMLP. In this research, we (1) introduce the original PROGNOS PO derived from UMP-ST (see Appendix L), (2) present the extended PROGNOS PO derived from HMLP in Section 6.2.2, and (3) compare two processes (UMP-ST and HMLP) in Section 6.2.3.
6.2.2 PROGNOS PO via HMLP

In this section, we introduce an extended PROGNOS PO derived from the HMLP process. The following shows how the development operates.

6.2.2.1 Analyze Requirements

This step is not much different from the requirement step in UMP-ST. Therefore, we can reuse requirements developed from the PROGNOS project (Appendix L). The full requirements can be found in [Carvalho, 2011]. However, the PSAW-MEBN reference model can provide more items by which a PSAW modeler can consider predefined entities, RVs, and MFrags for PSAW. Recall the four M_frag groups from the reference model: Reported Object, Observing Conditions, Target Object, and Situation (Chapter 4). For the awareness of the PROGNOS situation, we add the following new requirement.

**New Goal 1**: Recognize emergency situation at sea

**Query 1.1**: How high is the potential terrorist threat?

**Evidence 1.1.1**: Ship(s) of interest

**Evidence 1.1.2**: Crew member(s) of interest

The new goal aims to alert a response team when the threat reaches a certain level. This will be accomplished by estimating potential terrorist attacks in the field given estimation of terrorist ships and terrorist crew members.

In HMLP, a requirement can contain a performance criterion specifying a measure of accuracy (e.g., the mean squared error or the Brier score [Brier, 1950]). For example, we might require that the mean squared error between ground truth and
estimated results from the probabilistic ontology shall be less than a given threshold (e.g., a mean squared error < 0.1).

6.2.2.2 Define World Model

This step performs two sub-steps (Define Structure Model and Define Rules). The Define Structure Model step is to define a structure model for PROGNOS from the requirements defined in the previous step.

In this step, the PSAW-MEBN reference model can be used to identify possible entities, random variables, and causal relationships between the random variables. Fig. 6.4 shows a PROGNOS structure model represented in an EER (enhanced entity–relationship) model. We develop the PROGNOS structure model using the requirements and the reference model.

The reference model suggests four groups: (1) Reported Object, (2) Observing Condition, (3) Target Object, and (4) Situation. A structure model for the original PROGNOS PO included the seven relations (e.g., Target, Ship, Person, Organization, Person_Org, Ship_Person, and Ship_Ship). The original PROGNOS PO treated only the target object group. In other words, it did not emphasize sensing. We would expect evidence (e.g., reported objects) to be reported to estimate actual targets (e.g., target objects), so relations (i.e., Ship_Report, Person_Report, Organization_Report, Ship_Ship_Report, Person_Org_Report, Ship_Person_Report, and ReportedTarget) for the reported object group are added in the structure model for the extended PROGNOS PO. Observations may contain observation errors influenced by observing conditions (e.g., weather). The observing condition group contains two relations Sensor and
SensorProperty. In the previous step, a requirement for the awareness for a situation was added. Therefore, we added a relation Field for the situation group in Fig. 6.4. Relations (i.e., Location, SensorOf, and ActualTarget) which are not classified in these groups are supporting relations used to join the relations in the four groups.

The reference model provides some rules or causal relationships between these groups as shown in the arrows (Fig. 6.4). The observing conditions group and the target object group can influence the reported object group. For example, the attribute sensorPerformance in the relation SensorProperty influenced the report attributes in the report relations Ship_Report, Person_Report, Organization_Report, Ship_Ship_Report, Person_Org_Report, and Ship_Person_Report. The arrows in Fig. 6.4 indicate these causal relationships. The following shows a few of these rules.

**Rule 1:** causal (\{hasErraticBehavior, sensorPerformance\}, hasErraticBehaviorRPT)

**Rule 2:** causal (\{isShipOfInterest, isTerroristPerson\}, PotentialTerroristAttacks)

...  

Rule 1 means that two attributes hasErraticBehavior and sensorPerformance cause the attribute hasErraticBehaviorRPT. Rule 2 means that two attributes isShipOfInterest and isTerroristPerson cause the attribute PotentialTerroristAttacks.
6.2.2.3 Construct Reasoning Model

This step performs two sub-steps (Map to Reasoning Model and Learn Reasoning Model) to construct the PROGNOS PO. MEBN-RM provides a converting rule from RM to a probabilistic ontology. Entity relations which contain only one attribute for the primary key of the relation (e.g., ship and person) can be defined as entity types in the probabilistic ontology. Each of the attributes in the relations could be mapped to a resident node in the probabilistic ontology using MEBN-RM. For example, the attribute
hasErraticBehavior of the relation Ship became the resident node hasErraticBehavior(ship).

Rules which are defined in the previous step are used to develop causal relationships between resident nodes in the probabilistic ontology. For example, from Rule 1, we had a conditional dependence P(hasErraticBehaviorRPT(ship_report) | hasErraticBehavior(ship), sensorPerformance(shipSensor, ship)). From Rule 2, we had a conditional dependence P(PotentialTerroristAttacks(field) | isShipOfInterest(ship), isTerroristPerson(person)).

We could model the extended PROGNOS PO as shown in Fig. 6.5 using the resident nodes, the causal relationships between the resident nodes, and the MFragment groups.

Fig. 6.5 shows a set of MFrags in the extended PROGNOS PO. The list on the left indicates the four MFragment groups. Each group is decomposed into sub-groups. For example, the target object group contains five sets of MFrags (Person MFrags, Ship MFrags, MFrags for the relationship between two ships, MFrags for the relationship between a person and a ship, and MFrags for the relationship between a person and an organization). The following list (PO 6.1) shows part of new MFrags added into the extended PROGNOS PO.
PO 6.1: Part of New MFrags added into the original PROGNOS probabilistic ontology

1  [F: Organization_Report_MFrag
2     [C: isA(sr,SENSOR), isA(tr,ORGANIZATION), isA(rt,REPORTEDTARGET)]
3     [C: SensorOf(sr, tr), tr = ReportedTarget(rt)]
4     [R: isTerroristOrganizationRPT(rt)
5          [IP: isTerroristOrganization(tr)]
6          [IP: performance(sr, tr)]]
7  ]
8  ]
9  [F: Situation_MFrag
10     [C: isA(ship,SHIP), isA(person,PERSOON), isA(field,FIELD)]
11     [C: field = Location(ship)]
12     [C: hasCrewMember(ship, person)]
13     [R: PotentialTerroristAttacks(field)
14          [IP: isShipOfInterest(ship), isTerroristPerson(person)]]
15  ]
16  …

In PO 6.1, we added the ship report MFragment which can be used to reason about Rule 1. Also, we added the situation MFragment which can be used to reason about Rule 2.
In the *Learn Reasoning Model* step, the extended PROGNOS PO can be refined using a MEBN learning algorithm. The goal of MEBN learning is to learn an MTheory from a training dataset. A basic MEBN learning method for relational datasets was suggested [Park et al., 2013a][Park et al., 2013b]. This approach assumes that the training dataset is stored in a relational database based on RM. MEBN learning searches parameters, variables, and structures to find an MTheory that provides a good fit to the training dataset. In our case, the structures are given by the above steps as suggested in the PSAW reference model. Therefore, only parameter learning is required. Further information about MEBN parameter learning can be found in Chapter 5.

**6.2.2.4 Test Reasoning Model**

This step performs two sub-steps (*Experiment Reasoning Model* and *Evaluate Experimental Results*) to evaluate the extended PROGNOS PO from the test dataset. In the *Experiment Reasoning Model* step, the performance of estimation and prediction for the extended PROGNOS PO can be assessed using a performance measure (e.g., the mean squared error or the Brier score). Each experiment consists of the following five steps. (1) The test dataset provides entity information (e.g., ship1, person1, and field1) and ground truth information (e.g., isShipOfInterest_ship1 = true, isTerroristPerson_person1 = true) to the extended PROGNOS PO. (2) Given these, the extended PROGNOS PO is used to compute a marginal probability distribution (e.g., \( P(\text{PotentialTerroristAttacks_field1} \mid \text{isShipOfInterest_ship1} = \text{true}, \text{isTerroristPerson_person1} = \text{true}) \)) in response to a query. (3) The test dataset provides
ground truth data (e.g., PotentialTerroristAttacks_field1 = High). (4) Steps 1-3 are repeated for all test cases. (5) Finally, for results for all cases, the measures are calculated.

In the *Evaluate Experimental Results* step, we evaluate the measures using the performance criteria in the requirements defined in the *Analyze Requirement* step (e.g., a mean squared error < 0.1). If the evaluation is not satisfied (e.g., a mean squared error >= 0.1), we can return to the previous steps to improve the performance of the extended PROGNOS PO.

### 6.2.3 Comparing UMP-ST and HMLP

HMLP is a modification of UMP-ST that specifies some detailed sub-steps and uses two reference models (the PSAW-MEBN reference model and MEBN-RM) and MEBN learning. These reference models can support efficient modeling for a probabilistic ontology for PSAW. The first steps (*Requirement*) for both processes are almost same. In the case of HMLP, the PSAW-MEBN reference model provides a set of PSAW questions to develop requirements in this step. In the second step of HMLP, the reference model also supports developing a world model in terms of PSAW by providing candidate entities (i.e., $T$, $OR$, $SR$, $TR$, and $RT$), attributes, and causal relationships. In the third step of HMLP, MEBN-RM supports the development of entities, random variables, and MFrags from a relational model. HMLP also makes use of MEBN learning algorithms, so given a training dataset, a probabilistic ontology can be efficiently constructed. The second and third steps are mainly different with UMP-ST. These steps in HMLP can accelerate the modeling for probabilistic ontologies for PSAW and produce more comprehensive models.
Table 6.2 shows feature comparison between the original PROGNOS PO and the extended PROGNOS PO. Each number in the table means the number of the features (entities, random variables, causal relationships between random variables, and MFrags). For example, the number of entities in the original model is three (Ship, Person, and Organization), while the number of entities in the extended model is ten (Field, Ship, Person, Organization, ShipSensor, PersonSensor, OrganizationSensor, ReportedShip, ReportedPerson, and ReportedOrganization). Table 6.2 shows that the feature of the extended PROGNOS PO is more comprehensive than the feature of the original PROGNOS PO. The original PROGNOS PO contains 51 RVs, while the extended PROGNOS PO contains 115 RVs. This means that the extended PROGNOS PO can answer more various questions. For example, a reasoning about potential terrorist attacks in a field can be performed using the extended PROGNOS PO, but the original PROGNOS PO can’t. Also, the extended PROGNOS PO contains observing conditions for sensors, so this may enable us to perform more accurate reasoning.

<table>
<thead>
<tr>
<th></th>
<th>Entities</th>
<th>Random Variables</th>
<th>Causal Relationships</th>
<th>MFrags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original PROGNOS PO</td>
<td>3</td>
<td>51</td>
<td>53</td>
<td>18</td>
</tr>
<tr>
<td>Extended PROGNOS PO</td>
<td>10</td>
<td>116</td>
<td>147</td>
<td>36</td>
</tr>
</tbody>
</table>

If we assume that there is a training dataset for MEBN learning, the development period for the PROGNOS PO can be reduced. Usually, to develop an RV and its
parameter, we study literature related to the RV and find possible parameters for the RV. Another way for the development of such an RV is to use domain expert knowledge. A subject matter expert (SME) may provide values and parameters for the RV, and causal relationships between RVs. In the PROGNOS project, to develop one RV, we used the following steps: (1) an SME in the maritime domain explained domain knowledge to an RV developer, (2) the RV developer developed the RV using the MEBN/PR-OWL software [Costa et al., 2008], and (3) the RV in the MEBN/PR-OWL software was evaluated by the SME. These steps were implemented with at least one day per RV. If we assume that for each RV, one day may be required to develop it by one RV developer and one SME, then the original PROGNOS PO requires around 51 days. Note that in our experience for the original PROGNOS PO development, we spent more than 51 days. On the contrary, if we assume that all datasets are available, the development with MEBN learning may require around one day for setting the datasets and learning a PO using a MEBN learning algorithm.

6.3 Use Case 3 - Smart Manufacturing

Smart manufacturing relies on a combination of different sources providing key information to support diverse activities throughout the manufacturing process. Most smart manufacturing systems focus on activities directly related to the management of robots, conveyor belts, maintenance logs, and others that ensure the process runs smoothly. An initial step to support such smart manufacturing systems is an awareness process for estimating current situations and predicting future situations in manufacturing. This awareness process is called Predictive Manufacturing Situation Awareness (MSAW),
a special type of Predictive Situation Awareness (Chapter 4) for manufacturing. Our research addresses developing an MSAW system with the goal of enhancing industrial competitiveness (e.g., lower cost in shorter time with higher quality) for the manufacturing industry. This requires constant monitoring of market conditions, prices of manufacturing assets, and other inputs that would help to define how the production line behaves. This input is highly stochastic, which makes fusing the data from the diverse sources a challenge. In such situations, the MSAW system requires efficient knowledge representation for various situations and expeditious reasoning methods for estimating current situations as well as predicting future situations. In this research, we provide an overview of the data fusion process supporting MSAW using HMLP, including the representation of situations with associated uncertainty, and reasoning methods to support improved manufacturing processes.

6.3.1 Introduction

Innovation using technological convergence is accelerating throughout the industry with the advent of the latest technologies such as Artificial Intelligence (AI), Cloud Computing, Internet of Things (IoT), Cyber-Physical Systems (CPS), and Big Data. Across industry, such new technologies have been used to realize innovations for more efficient manufacturing by combining approaches from traditional manufacturing and information technology through effective use of real-time big data obtained from sensors applied to the manufacturing process.

Manufacturing is the process of producing products that address requirements from customers. The requirements of traditional manufacturing can be to reduce
production cost, improve product quality, and reduce development time. Technological convergence through AI, Cloud Computing, IoT, CPS, and/or Big Data enables smart manufacturing: “From an engineering standpoint, smart manufacturing is the intensified application of advanced intelligence systems to enable rapid manufacturing of new products, dynamic response to product demand, and real-time optimization of manufacturing production and supply chain networks. [Coalition, 2011]”. Smart manufacturing uses AI technology to respond proactively, responsively, and adaptively to market requirements. To do so requires capabilities such as self-awareness, self-prediction, self-reconfiguration, and self-maintenance. The initial step to support smart manufacturing can be to perform Predictive Situation Awareness (PSAW) (Chapter 4). Predictive Situation Awareness for Manufacturing (or MSAW) is PSAW applied to the manufacturing domain.

A system supporting MSAW, called an MSAW system, requires comprehensive knowledge representation for various situations and expeditious reasoning methods to estimate current situations as well as to predict future situations. MSAW models expressed as MEBN, called MSAW-MEBN models, can be constructed using the PSAW-MEBN reference model (Chapter 4) which is a reference model to develop a MEBN model for PSAW, enabling rapid and efficient construction of MSAW-MEBN models. Situation analysis using such MSAW-MEBN models will support effective and efficient decision making.

In this research, we introduce an MSAW-MEBN model, developed via the PSAW-MEBN reference model, to reason about complex situations in manufacturing.
Also, we present a use case of the MSAW-MEBN models supporting MSAW for assessing product quality in steel plate manufacturing. Further, we describe how to evaluate uncertainty from reasoning results performed in the use case.

This research (1) introduces background information for the MSAW-MEBN model (see Appendix M.1), (2) discusses smart manufacturing (see Appendix M.1), (3) provides representation for uncertainty in smart manufacturing (see Appendix M.2), and (4) presents a use case for evaluation of the MSAW-MEBN model we present in this research (see Appendix M.3).

**6.3.2 MSAW-MEBN Model for Predictive Situation Awareness**

For smart manufacturing, we consider a basic uncertainty model for a manufacturing process (see Fig. M.4 in Appendix M). This model is highly simplified. In a more complex process, most data for manufacturing comes through sensors. Performance of such sensors will be affected by environmental factors as well as sensor conditions (e.g., sensor degradation and failure). Thus, data from sensors can contain errors. Also, manufacturing information is not the only relevant information which we want to know. A decision maker in charge of management for manufacturing may need some overall situation information (e.g., manufacturing states, total cost, and total time). The PSAW-MEBN reference model (Chapter 4) provides some information on how to represent such situations. In this research, we present an extended MSAW-MEBN model derived using the PSAW-MEBN reference model. The following figure shows an uncertainty model that extends the basic uncertainty model for manufacturing process (Fig. M.4 in Appendix M) according to the PSAW-MEBN reference model.
Fig. 6.6 shows four groups corresponding to four MFrags (Observing condition M_frag, Report M_frag, Target M_frag, and Situation M_frag) in the PSAW-MEBN reference model. An observing condition group represents probabilistic knowledge about conditions of a sensor (e.g., sensor capability and sensing environment). A report group represents probabilistic knowledge about a relation or an attribute of observed targets (e.g., reported system quality and reported item quality). A target object group represents
probabilistic knowledge about a relation or an attribute for actual targets (e.g., actual system working time and actual item quality). A situation group represents probabilistic knowledge about situations of manufacturing factors (e.g., evaluation measure for manufacturing factors).

Fig. 6.6 also shows possible causal relationships (conditional relationships in the figure). RVs in the observing condition group and the target object group can cause RVs in the report group. RVs in the target object group can cause RVs in the situation group. These four groups can be used to construct MFrags for an MSAW-MEBN model for PSAW. The causal relationships can be used to model the causal relationship between RVs in such an MSAW-MEBN model.

In Chapter 4, we introduced five core entities for a MEBN model for PSAW: The observer entity \( OR \), the target entity \( TR \), the reported target \( RT \), the sensor entity \( SR \), and the time entity \( T \) in the PSAW-MEBN reference model. In the MSAW-MEBN model, the target entity \( TR \) is replaced by two entities (a target system entity \( TS \) and a target item entity \( TI \)). Also, the reported target \( RT \) is replaced by two entities (a reported target system entity \( RS \) and a reported target item entity \( RI \)). Therefore, the MSAW-MEBN model for PSAW contains 7 entities: \( OR, SR, T, TS, TI, RS, \) and \( RI \).

The following summarizes the elements for the MSAW-MEBN model.

**MFrags**

*Observing condition M_frag*

*Report M_frag*

*Target (System, Input, Output) M_frag*
Random Variables

Situation Property $SIT_RV$
System Property $SYS_RV$
Item Property $IT_RV$
Reported System Property $RSYS_RV$
Reported Item Property $RIT_RV$
Observing Condition $OC_RV$

Causal Relationships between Random Variables

$\{SYS_RV, IT_RV, SIT_RV \} \rightarrow SIT_RV$
$\{IT_RV, SIT_RV \} \rightarrow SYS_RV$
$\{SYS_RV, IT_RV, SIT_RV \} \rightarrow IT_RV$
$\{SYS_RV, OC_RV\} \rightarrow RSYS_RV$
$\{IT_RV, OC_RV\} \rightarrow RIT_RV$
$\{OC_RV\} \rightarrow OC_RV$

Entities

Observer Entity $OR$
Sensor Entity $SR$
Time Entity $T$
Target System Entity $TS$
Target Item Entity $TI$
Reported Target System Entity RS
Reported Target Item Entity RI

Note that the MSAW-MEBN model can contain five MFrags discussed in the PSAW-MEBN reference model in Chapter 4. In the MSAW-MEBN model, the Target M Frag is divided into three MFrags: System, Input, and Output (see Fig. M.5 in Appendix M) to represent the manufacturing process.

6.3.3 Discussion
In this research, we introduced an MSAW-MEBN model, developed via the PSAW-MEBN reference model (Chapter 4), to reason about complex and complicated situations in manufacturing. A use case using the MSAW-MEBN model can be found in Appendix M.3. The MSAW-MEBN model in the use case supports MSAW for a steel plate manufacturing process to predict the total cost, total time, and total quality rate. In this research, we were able to develop the MSAW-MEBN model quickly using the PSAW-MEBN reference model, by following the process of HMLP, in order to develop a smart manufacturing system. Results of evaluation are given in Appendix M.3.

6.4 Experimental Comparison between UMP-ST and HMLP
We conducted an experiment to compare two MEBN development processes (UMP-ST and HMLP) in terms of development time and accuracy. The experiment protocol was approved by the Institutional Review Board (IRBNet ID: 1054232-1). A MEBN model can be constructed by either UMP-ST or HMLP. UMP-ST is the traditional manual approach to develop a MEBN model, while HMLP is the new approach which is studied in this dissertation.
In this experiment, there were two groups (A and B) selected from six adult people. Both groups were required to develop a MEBN model from stakeholder requirements. The main requirement was to develop a MEBN model for a very simplified domain of a steel plate factory. Thus, we conducted a simplified development experiment, MEBN modelling for simple heating machinery. For purposes of validity, we tried to constitute similar conditions for both groups (e.g., same level of knowledge of the domain, BN, MEBN, and MEBN modeling). Finding and inviting participants who are working for a same domain and have same level of knowledge is difficult. For that reason, the participants who didn’t have any experience in the target domain for the experiment were selected and were given domain knowledge to develop a MEBN model. Thus, knowledge for simple heating machinery was given to both groups. This knowledge given to the participants is introduced in Section 6.4.1.

For the experiment, we performed three processes (preparation, execution, and evaluation). In the preparation process, we prepared the experimental settings to make both groups to have same conditions in terms of knowledge and skill for MEBN modelling for the simple heating machinery. In the execution process, the main experiment for MEBN modeling was conducted. In the process, participants in both groups had developed MEBN models using the methods assigned to each of them. In the evaluation process, development times by the participants were analysed and MEBN models developed by them were tested in terms of accuracy using a simulated test dataset.
6.4.1 Preparation Process

For the experiment, six adult people were invited as subjects. Initially, they didn’t have much knowledge about BN and were completely unfamiliar with MEBN, UMP-ST, and HMLP. (1) Before the execution process, the six people were given such knowledge to develop a MEBN model for the simple heating machinery in the execution process. The lecture contained the minimum amount of knowledge for (continuous) BN, BN modelling, MEBN, a script form of MEBN, and MEBN modelling (i.e., UMP-ST) to develop the MEBN model for the simple heating machinery (See Appendix N.1 for the slides presented to participants). Note that a lecture for full knowledge of such domains may require several semesters, so the scope of the experiment was reduced to a smaller size (i.e., the simple heating machinery) rather than the development of a MEBN model for full heating machinery. The six people were divided into two groups (Groups A and B). Group A used UMP-ST, while Group B used HMLP to develop a MEBN model. (2) To constitute same conditions for each group in terms of skills and domain knowledge for MEBN and UMP-ST, a short test for such knowledge was taken to all of the participants. (3) The short test was graded by a MEBN expert who was not investigator for this experiment. An identity of each participant was not given to the MEBN expert to prevent intervention of prejudice. To penalize our new approach HMLP, the first (third and fifth) ranked participant belonged to Group A. Second (fourth and sixth) ranked participant belonged to Group B.


### Table 6.3 Preparation Process

<table>
<thead>
<tr>
<th>Steps</th>
<th>Group A (UMP-ST)</th>
<th>Group B (HMLP)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Obtain relevant knowledge</td>
<td>Provided a lecture for BN, MEBN, the script form of MEBN, and UMP-ST</td>
<td></td>
<td>4 hours</td>
</tr>
<tr>
<td>2. Take a short test</td>
<td>Provided a short test for UMP-ST &amp; MEBN</td>
<td></td>
<td>30 min</td>
</tr>
<tr>
<td>3. Divide into two groups</td>
<td>Graded the test results and selected participants for two groups</td>
<td></td>
<td>1 hour</td>
</tr>
<tr>
<td>4. Obtain HMLP knowledge</td>
<td>None</td>
<td>Provided a lecture for HMLP</td>
<td>Time was checked</td>
</tr>
</tbody>
</table>

Before the execution process, (4) an HMLP lecture was provided to Group B (see Appendix N.1). The time for the lecture was checked as *Time A*. The lecture contained the process of HMLP, the PSAW-MEBN reference model, and how to use HMLP Tool (see Appendix F).

### 6.4.2 Execution Process

In the execution process, both groups were requested to develop a MEBN model for a simple heater system heating a slab to support a next manufacturing step (e.g., a pressing step for the slab to make a steel plate). The MEBN model aimed to predict a total cost for the heater given input slabs.

(5) In the first step of the execution process, both groups were given a stakeholder requirement, “Develop a MEBN model which is used to predict a total cost given input slabs”. Also, domain knowledge was given to the participants (see Appendix N.1).

The domain knowledge was about the following information: The simple heater system is associated with two infrared thermal imaging sensors to sense the temperature
of a slab, each sensor has a sensing error with a normal distribution with a mean zero and a variance three, $N(0, 3)$ (e.g., if it sensed 10 °C this means that the actual temperature is in a range between 7.15 °C and 12.85 °C with the 95th percentile), the heater system contains an actuator which is used to control an energy value to heat a slab (i.e., the actuator calculates the energy value given the input slab temperature), there is no energy loss when the energy value is used in the heater, all manufacturing factors (e.g., the temperature, energy value, and cost) are normally distributed continuous values, the energy unit is kWh (kilowatt-hour), there is a fixed slab weight 100kg, there is an ordered fixed temperature 1200 °C for an output slab coming from the heater, and the energy cost is 20cent/kWh.

![Figure 6.7 Situation for the simple heating machinery](image)

Also, an idea of how to model the sensor error using BN was given. For example, to include the sensor error, two random variables are used. The first random variable is
for an actual temperature, while the second random variable is for a sensed temperature. The actual temperature, then, influences the sensed temperature with the error normal distribution (i.e., N(0, 3)). This can be modelled in a BN as \( P(\text{sensed temperature} \mid \text{actual temperature}) = \text{actual temperature} + N(0, 3) \).

For the situation of the simple heating machinery, datasets were generated by a simulator containing a ground truth model designed by a domain expert. The ground truth model contained two parts. The first part is for an actual model which represents a physical world which can’t be observed exactly. The second part is for a sensed model which represents an observed world where we can see using sensors. Therefore, the datasets were divided into two parts: Actual data and sensed data (Fig. 6.8). The sensed data (data sets in the rounded boxes in Fig. 6.8) were provided to both groups in two formats: The data in an excel format and the data in a relational database (RDB) (See Appendix N.2). The actual data (e.g., actual temperatures) were not given to either group.
Also, the simulator generated two datasets (as shown in Fig. 6.8): One was a training dataset which was used by the participants to understand the context of the situation and learn a MEBN model using HMLP, and another was a test dataset which was used to evaluate the models developed by the participants in terms of prediction accuracy for the total cost (6.4.3 Evaluation Process). For this model evaluation, the actual and sensed data in the test dataset were used. For example, sensed data for the temperature of an input slab were used as evidence for the developed model and the developed model was used to reason about a predicted total cost. The predicted total cost was compared with a total cost derived from the actual data in the test dataset.

Participants were requested to (6) develop MEBN model requirements, (7) Define World Model, and (8) construct Reasoning Model. And the development times Time B, Time C, and Time D respectively were checked.
Table 6.4 Execution Process

<table>
<thead>
<tr>
<th>Steps</th>
<th>UMP-ST (Group A)</th>
<th>HMLP (Group B)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5. Obtain stakeholder requirements and domain knowledge</td>
<td>Provided stakeholder requirements and domain knowledge to both groups</td>
<td></td>
<td>1 hour</td>
</tr>
<tr>
<td>6. Analyze Requirements</td>
<td>Developed MEBN model Requirements</td>
<td></td>
<td>Time was checked (Time B)</td>
</tr>
<tr>
<td>7. Define World Model</td>
<td>Developed a structure model and rules</td>
<td></td>
<td>Time was checked (Time C)</td>
</tr>
<tr>
<td>8. Construct Reasoning Model</td>
<td>Developed MEBN model using the script form of MEBN</td>
<td>Develop MEBN model using the HMLP tool</td>
<td>Time was checked (Time D)</td>
</tr>
</tbody>
</table>

6.4.3 Evaluation Process

In this process, the MEBN models developed by the participants were evaluated and their development times were analyzed. For the model evaluations, simulated test datasets were used. The development times for both were measured according to the use of methods UMP-ST and HMLP.

Our goal for this experiment is to compare two methods in terms of the development time. However, in some cases, a MEBN model is developed quickly with low accuracy. The comparison between a low quality model and a high quality model in terms of the development time is unfair. Thus, obviously, in order to demonstrate the superiority of HMLP against UMP-ST, the results of this experiment should ensure two things: A quality of the model developed using HMLP is equal or better than a quality of the model developed using UMP-ST, and the development time using HMLP is faster than using UMP-ST. In this experiment, we used the prediction accuracy as the quality of
the developed model, because the mission of the developed model is to predict the total cost for the simple heating machinery.

(9) The first step in this process is to evaluate the accuracies of the MEBN models developed by the participants. For this, an accuracy test for the models was performed to determine how well the models predict the total cost using the test dataset generated from the simulator. The simulator generated the test dataset regarding a situation in which three slabs were inputs and a total cost for heating the three slabs was an output. The total cost was calculated using the energy values in the actuator. The output (the total cost) was used to compare a predictive cost reasoned from the MEBN models. To the comparison between the total cost and the predictive cost, we used a continuous ranked probability score (CRPS) in which a perfect prediction yields a score of zero (Appendix C). For prediction accuracy metrics, we can use a mean absolute error (MAE). MAE uses a mean value only to compare an actual (or observed) value with a predictive value, while CRPS uses a predicted probability distribution for comparison (i.e., a mean and a variance). Therefore, CRPS analysis is more precise than MAE analysis. In this step, for a case (i.e., three input slabs and one output total cost) in the test data, CRPS was calculated using a predictive cost. Then, 100 cases were used to compute 100 CRPSs and they were averaged (i.e., Average CRPS).

(10) The development times for both groups were measured according to each step in UMP-ST and HMLP. The development times Time A–D were checked in the preparation process and the execution process. In this step, a total development time was
calculated. For Group A, a total development time included $Time B-D$, while for Group B, a total development time included $Time A-D$.

Table 6.5 Evaluation Process

<table>
<thead>
<tr>
<th>Steps</th>
<th>UMP-ST (Group A)</th>
<th>HMLP (Group B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9. Evaluate accuracy of model</td>
<td>Tested both models in terms of accuracy using a simulated test dataset</td>
<td></td>
</tr>
<tr>
<td>10. Evaluate development time</td>
<td>Measured the development times for both according to the use of methods UMP-ST and HMLP</td>
<td></td>
</tr>
</tbody>
</table>

6.4.4 Comparison Results

Table 6.6 shows an average CRPS to show model accuracy and a total development times to show efficiency of modelling methods for each participant. In the table, the grand CRPS average for Group A is higher than the grand CRPS average for Group B. This means that the MEBN models from Group B are better than the models from Group A in terms of accuracy. Then, the comparison for the total development times makes sense. The average of the total development times from Group A is higher than the average of the total development times from Group B. This implies that HMLP is a faster process than UMP-ST.
Table 6.6 Comparison results for total development times and average accuracies

<table>
<thead>
<tr>
<th>Group</th>
<th>Participants</th>
<th>Average CRPS</th>
<th>Total Development Times (Hours: Minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group A</td>
<td>#1</td>
<td>1735.3</td>
<td>1:06</td>
</tr>
<tr>
<td>(UMP-ST)</td>
<td>#2</td>
<td>74.6</td>
<td>2:48</td>
</tr>
<tr>
<td></td>
<td>#3</td>
<td>114.78</td>
<td>2:21</td>
</tr>
<tr>
<td></td>
<td>Grand Average (Standard Deviation)</td>
<td>641.53 (947.45)</td>
<td>2:05 (0:52)</td>
</tr>
<tr>
<td>Group B</td>
<td>#4</td>
<td>45.05</td>
<td>1:02</td>
</tr>
<tr>
<td>(HMLP)</td>
<td>#5</td>
<td>45.05</td>
<td>0:58</td>
</tr>
<tr>
<td></td>
<td>#6</td>
<td>40.48</td>
<td>1:28</td>
</tr>
<tr>
<td></td>
<td>Grand Average (Standard Deviation)</td>
<td>43.53 (2.64)</td>
<td>1:09 (0:16)</td>
</tr>
</tbody>
</table>

In the experiment, we expected the participants would develop an ideal MEBN model. The following figure shows ideal causal relationships between random variables for the simple heating machinery. In the ideal model, there are three parts: A situation group, an actual target group, and a report group. The situation group contains a random variable representing an overall total cost for this system. The actual target group contains three random variables (a temperature for an input slab, an actual energy for heating, and a temperature for an output slab). The report group contains two random variables (an observed/sensed temperature for the input slab and an observed/sensed temperature for the output slab).
In the experiment, we observed where the participants spent a lot of time. Table 6.7 shows time-consuming tasks in the experiment. The mark “X” in the table means that it is a time-consuming task for the method.

Table 6.7 Comparison results for time-consuming tasks

<table>
<thead>
<tr>
<th>Time-consuming tasks in the experiment</th>
<th>Group A (UMP-ST)</th>
<th>Group B (HMLP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Following process (UMP-ST or HMLP)</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Supported by HMLP tool)</td>
</tr>
<tr>
<td>2. Finding structure model/rules</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Supported by the PSAW-MEBN reference model)</td>
</tr>
<tr>
<td>3. Finding entity/RV/MFrag from relational data</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Supported by MEBN-RM)</td>
</tr>
<tr>
<td>4. Finding parameter</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Supported by MEBN parameter learning)</td>
</tr>
</tbody>
</table>

(1) Following UMP-ST process: Although the participants had studied UMP-ST, it was not easy to follow the process. They didn’t have many experiences to develop a
MEBN model using UMP-ST, so they were not familiar with the process. They remembered the process by reading a UMP-ST paper and developed their model according to each step of UMP-ST. For Group B, the HMLP tool supported the development of a MEBN model. By clicking some buttons in the HMLP tool, each step in HMLP was shown and the participants could make the models quickly. (2) Finding Structure Model/Rules: The participants in both Groups were required to find the structure model for the simple heating machinery. Although knowledge of the simple heating machinery situation was given, the participants in both groups struggled to find the structure model and rules. Group B was taught about the PSAW-MEBN reference model. The PSAW-MEBN reference model provided knowledge about a set of random variable groups (Situation, Actual Target, and Report) as shown Fig. 6.9 and causal relationships (i.e., rules). However, such knowledge did not have much influence on the development time for the structure model and rules, because the context for the simple heating machinery was too simple to use the PSAW-MEBN reference model. So, the participants in the two groups thought about their models in similar ways. However, the participants could not be sure whether or not their models were correct, so they spent relatively more time to think about their structure models and rules. (3) Finding entity/RV/MFrag from the RDB: The participants in Group A could not be sure of which elements in the RDB can be entity/RV/MFrag in MEBN, so they used times to figure out this. On the other hand, the participants in Group B used the HMLP tool containing MEBN-RM, so they didn’t consider this step much. (4) Finding CLD: The participants in
Group A looked at data to find normal distributions and regression models for RVs, while the participants in Group B used the MEBN parameter learning built in the HMLP tool.

In conclusion, HMLP could be used to develop more quickly a MEBN model than the traditional approach. Also, in this experiment, we found that the PSAW-MEBN reference model should be better explained to MEBN developers. For this, several examples for the PSAW-MEBN model may help the developers.
CHAPTER SEVEN: CONCLUSION AND FUTURE WORK

In this research, we introduced a new development framework for MEBN, providing a semantically rich representation that also captures uncertainty. MEBN was the core logic for the probabilistic ontologies used in the PSAW systems (PROGNOS and HERALD). The original probabilistic ontologies for PROGNOS were constructed manually with the help of domain experts. This manual MEBN modeling was labor-intensive and insufficiently agile. To address this problem, we introduced a development framework combining machine learning with subject matter expertise to construct MEBN-based probabilistic ontologies. (1) To enable learning from relational databases, we presented a bridge between MEBN and the Relational Model, which we called MEBN-RM. (2) We introduced a PSAW-MEBN reference model which provided a set of basic templates to support the design of a MEBN model for PSAW. (3) We proposed HMLP which was a framework to develop a MEBN model from the domain expert's knowledge combined with relational data. (4) We also presented a MEBN parameter learning for MEBN. The future research topics for the above contributions will be discussed in the following subsections.

7.1 MEBN-RM Mapping Model

MEBN-RM can support a MEBN learning algorithm, which develops an MTheory from a dataset, or an MTheory developer, who aims to develop an MTheory
using domain knowledge and MEBN knowledge. HMLP exploits MEBN-RM for efficient development of an MTheory. The idea behind MEBN-RM can be used to develop other mapping models for different types of database (e.g., ontology, graph, and event database) as an example mapping model. Recently non-relational databases, called NoSQL, are receiving increasing attention. In the era of Big Data, we may need a scalable and flexible database to manage the many and varied types of data. In this research, we only focus on the Relational Model as a source data model to develop an MTheory. Future work will consider extensions to NoSQL data and other types of data.

7.2 The PSAW-MEBN Reference Model

The PSAW-MEBN reference model could support the design of a PSAW-MTheory and improve the quality of the PSAW-MTheory. In future work, the reference should be evaluated in other use cases of PSAW. Thereby, the PSAW-MEBN reference model may evolve into a more thorough reference model to address more complex situations. Also, the presented model should be extended to address other questions related to PSAW other than the questions addressed in this research.

7.3 Human-aided MEBN Learning for PSAW

HMLP improved MEBN learning by providing expert knowledge which was used to limit the search space of parameters and structures for a PSAW-MTheory. The systems in the three case studies (PROGNOS, HERALD, and MSAW system) were proofs of concept, which were early models to test a concept or process for actual systems. Future steps for HMLP are to apply it to a realistic reasoning system for PSAW. Also, HMLP should be more thoroughly investigated in terms of efficiency (agility for the
development of a reasoning model) and effectiveness (producing a correct reasoning model).

7.4 Future Research Questions
There remain many open research issues in MEBN Learning. In this section, we introduce some issues which are left for future research.

7.4.1 Aggregating Influence Problem
This is the problem for learning an aggregating function in an aggregating situation where an instance child random variable depends on multiple instance parents which are generated from an identical class random variable. For example, in Chapter 2, we can assume that there is a danger MFragment for \( P(DangerLevel \mid VehicleType) \) (i.e., Fig. 2.1). If there are one region and three vehicle entities locating in the region and then we will have an SSBN containing one \( DangerLevel \) RV and its three \( VehicleType \) parent RVs. Because there are three parents (e.g., \( VehicleType_{V1} \sim 3 \)), the instance local distribution of the \( DangerLevel \) RV, which is derived from a class local distribution of a \( DangerLevel \) resident node, should reflect the multiple parent situation (i.e., Fig. 2.2). In our example, the instance local distribution of the \( DangerLevel \) RV should have all combination of the conditional probability such as \( P(DangerLevel = High \mid VehicleType_{V1} = Tracked, VehicleType_{V2} = Tracked, VehicleType_{V3} = Tracked) \) ... \( P(DangerLevel = Low \mid VehicleType_{V1} = Wheeled, VehicleType_{V2} = Wheeled, VehicleType_{V3} = Wheeled) \). Now, we can think of a case of different number parents in which we may have 10 vehicle entities locating in the region. Obviously, the instance local distribution of the \( DangerLevel \) RV should be generated by a different way. In such
case, the class local distribution should have a shared parameter generating various instance local distributions given the different number of parents. We call it an aggregating influence situation and a solution for this issue is to find a method which learns a class local distribution from data.

7.4.2 Dynamic Model Learning
This is the problem for learning a dynamic model in which relations of objects change over time. An MTheory can represent a static situation as well as a dynamic situation (containing a temporal change). For the dynamic situation, we can think of a dynamic MTheory which can reason about the change of a situation over time and is required for PSAW. In Section 6.1, the HERALD MTheory has dynamic components (see Fig. G.2 in Appendix G). For example, in the HERALD MTheory (MTheory I.1 in Appendix I), the RV $Temperature(tr, pret)$ is one step prior to the RV $Temperature(tr, t)$. Thus, the HERALD MTheory is based on a first-order Markov assumption (i.e., an RV at a time $n$ only depends on an RV at a time $n - 1$), while a dynamic MTheory can be based on a second-order Markov assumption (i.e., an RV at a time $n$ depends on two RVs at times $n - 1$ and $n - 2$, respectively). Dynamic model learning for an MTheory under a multi-order Markov assumption will be researched in the future.

7.4.3 Continuous Random Variable Learning
This is the problem for learning an MTheory which includes continuous random variables. An MTheory for PSAW may contain continuous random variables which are assigned to a real value with a continuous probability distribution. In this issue, we should find best parameters for the continuous probability distribution which fits well
data. In Chapter 5, we introduced MEBN parameter learning for Conditional Linear Gaussian. We will extend this to other continuous distributions (e.g., Pareto distribution).

### 7.4.4 Learning from Incomplete Data

This is the problem for approximating parameters and structures of an MTheory for missing data and variables. Commonly, having complete data is not realistic. Although many BN Learning methods address this problem, we should deal with a different case in MEBN Learning. In the BN Learning case, there are missing values, “null”, in a dataset for RVs of a BN, as depicted the dataset A in Table 7.1 below. In the MEBN learning case, an RV can be related with a different number of instance RVs. For example, as the dataset B depicted in Table 7.1, in the case 1, the $Y$ RV is related with three $X$ RVs and the third value is missing. In the case 2, the value of the $Y$ RV is missing.

In MEBN learning we are presenting, we assume that there is no missing data in the dataset. In an advanced MEBN learning, we can use an expectation–maximization (EM) algorithm [Dempster et al., 1977] to handle missing data.

In the expectation step in the EM algorithm, sufficient statistics for missing data are replaced by expected values using estimation from the current estimates of parameters and observed data. In the maximization step, estimated sufficient statistics in the expectation step are used to find new maximized parameters for data. The maximized parameters used in the next expectation step again to estimate the missing data. These steps expectation and maximization iterate until the parameters converge.
Table 7.1 Dataset A represents the BN data case. Dataset B represents the MEBN data case

<table>
<thead>
<tr>
<th>Case</th>
<th>X</th>
<th>Y</th>
<th>Case</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>null</td>
<td>T</td>
<td>1</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>F</td>
<td>2</td>
<td>T</td>
<td>null</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>F</td>
<td>3</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>null</td>
<td>T</td>
<td>4</td>
<td>null</td>
<td>T</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>null</td>
<td>5</td>
<td>null</td>
<td>T</td>
</tr>
<tr>
<td>6</td>
<td>T</td>
<td>T</td>
<td>6</td>
<td>null</td>
<td>T</td>
</tr>
</tbody>
</table>

It is possible that there are insufficient evidence for variables, structure, and data to develop a model. For example, suppose that we have a dataset from sensors; however, the dataset doesn’t contain the necessary information to identify a target intention (or mission). To address this problem, we can rely on expert's knowledge and insight. For example, a domain expert provides probabilistic information for the target intention. And this information and the dataset from the sensors are used to design the model. MEBN learning in this research is HMLP which relies partially on expert's knowledge and insight to reduce the search space as well as to supplement insufficient evidence. We did not address missing data in this research, however this is an important issue for future research.

7.4.5 Learning for Hidden Variables

When a random variable is missing, this is called a “latent” or “hidden” variable. Learning such a hidden variable is treated extensively in the domain of Bayesian network learning [Cooper, 1995]. For such learning, candidate variables are given by a domain knowledge expert and these candidate variables are investigated by measuring how they fit well to data. This is an important problem that we did not address in this research. In the future research, we should address it.
7.4.6 Incremental MEBN Learning
This is the problem for learning an MTheory from updated observations. In the real world, commonly, data are updated over time, so a learned model is no more valid. Learning a new model from all dataset is an inefficient process, because of a big dataset issue. A model which was learned before has information for the previous dataset, so an approach in which the learned model and the updated data are combined is a more efficient approach. Bayesian approach enables MEBN learning to be incremental MEBN learning. For example, information for a learned model is used for a prior probability and incremental MEBN learning combines the prior probability and updated information to calculate a posterior probability.

7.6 Conclusion
PSAW is an emerging field in the military, commercial, and governmental domain. These domains require high-quality decision making. To perform this, the ability for PSAW should be improved in terms of agility and accuracy. A PSAW system supported by cooperation between human experts and machines may enable us to enhance the PSAW capability. Risks, threats, and opportunities around us can be detected, identified, classified, estimated, and predicted by a PSAW system. Such a PSAW system can be used for several domains such as cyber defense, critical infrastructure defense, military defense, crime prevention, terrorism prevention, natural disaster prevention, health risk detection, customer demand detection, and public sentiment detection. In the era of big data, the explosion of dark data, which is not being used, can hinder efficient PSAW. Human operators for PSAW cannot efficiently and effectively treat such a
situation. Relying on a PSAW system supported by human and supporting for human will be ubiquitous in our life.
APPENDIX

Appendix A introduces Bayesian Network learning which is used for MEBN parameter learning (Chapter 5). Appendix B shows a set of PSAW questions, which were introduced in Chapter 4. Appendix C provides formal definitions of the Brier score and the continuous ranked probability score (CRPS). Appendix D introduces the grammar for Local Probability Description Language (LPDL), which is used to describe a class local distribution of an RV. Appendix E shows the grammar for Conditional Probability Script Language (CPSL) for an instance local distribution of an RV. Appendix F shows an HMLP tool that performs MEBN-RM and the MEBN parameter learning. Appendix G describes the HMLP application for HERALD (Section 6.1). Appendix H describes the HERALD scenario simulator, which was introduced in Section 6.1. Appendix I provides the HERALD MTheory in a graphic form. Appendix J describes the confusion matrix for categorical variables from the reasoning results using the HERALD MTheory in Chapter 6.1. Appendix K contains one of SSBNs for the HERALD MTheory. Appendix L explains the development of PROGNOS PO via UMP-ST for Section 6.2. Appendix M presents case study 3 (Predictive Situation Awareness Model for Smart Manufacturing in Section 6.3) in which we provide an overview of the data fusion process supporting MSAW using HMLP, including the representation of situations with associated uncertainty, and reasoning methods to support improved manufacturing processes.
Appendix N presents slides and data examples used for the experiment in Section 6.4, and experimental results. Appendix O explains an inference algorithm, called Hybrid Message Passing with Gaussian Mixture Reduction, which is used for SSBNs of the MTheories in Chapter 6. Appendix P shows test Hybrid Bayesian Networks for the inference algorithm in Appendix O.
Appendix A. Bayesian Network Learning

Bayesian Networks (BN) learning from data is a process to find a Bayesian network that fits data well. Given a graph of BN and a dataset, Parameter Learning is the problem of finding a parameter $\theta$ that provides a good fit to the data. Structure Learning is the problem of finding a graph $G$ of BN that provides a good fit to the data. Structure Learning can have following topics: (1) dependency or independency between nodes can be learned, (2) a hidden or an unobserved node in a BN can be found, and (3) a functional form of a local distribution for a node can be identified.

In Bayesian theory, any uncertain aspect of the world can be represented as a random variable (RV). In BN learning, data $D$, graph $G$, and parameter $\theta$ can be RVs. The set of data $D$, graph $G$, and parameter $\theta$ are represented as $D$, $G$, and $\Theta$, respectively. For BN, data $D$ means flat data which has no relationships among its records, while for MEBN learning in this research, data $D$ means relational data. In the following subsections, we introduce common approaches for BN parameter learning. We refer to [Pearl 1988][Heckerman, 1998][Koller & Friedman, 2009] for following subsections.

A.1 BN Parameter Learning

BN parameter learning is to find parameters that fit a dataset well. We can think of this as inference in probability theory. Two of the most popular approaches to estimating parameters from data are Maximum Likelihood Estimation (MLE) and Bayesian inference. In the Bayesian approach, we begin with a prior distribution for the parameter and use the data to obtain a posterior distribution. MLE finds the parameter that maximizes the likelihood function, and does not use a prior distribution. The use of
the prior distribution in the Bayesian approach can help to overcome the over-fitting problem of learned parameters. We introduce MLE first and then the Bayesian approach.

### A.1.1 Maximum Likelihood Estimation

For MLE, let’s assume that we are doing a statistical experiment (e.g., tossing coins, where H is a head and T is a tail). In the experiment, there is a set of independent and identically distributed (IID) observations \( D = \{D_1, D_2, ..., D_n\} \) (e.g., \{H, H, T\}), which are drawn at random from a distribution with an unknown probability density or mass function \( f(D_i \mid \theta_A) \), where \( n \) is the number of the observations and \( \theta_A \) is an actual parameter for the distribution. Since we don’t know the actual parameter \( \theta_A \), we find an estimator \( \theta^* \) which would be close to the actual parameter \( \theta_A \). For this, we introduce a function, called *likelihood*, which is a function of a parameter \( \theta \) for given observations \( D \) and is used to find the estimator \( \theta^* \).

\[
L(\theta : D) = P(D \mid \theta), \tag{A.1}
\]

where \( D \) is the observations and \( \theta \) is a parameter.

The parameter \( \theta \) can contain sub-parameters \( \{\theta_1, \theta_2, ..., \theta_m\} \), where \( m \) is the number of the sub-parameters in \( \theta \). For example, the parameter \( \theta \) for a normal distribution can contain two sub-parameters \( \theta_1 \) (for mean) and \( \theta_2 \) (for variance). Also, a parameter \( \theta \) can have only one sub-parameter \( \theta_1 \) (e.g., \( \theta_1 = P(H) \)).

The likelihood function \( L(\theta : D) \) in Equation A.1 is equal to the probability of the observations \( D \) given a parameter \( \theta \). We then define an estimator \( \theta^* \), called a *maximum likelihood estimator* in Equation A.2.
\[ \theta^* = \arg \max_{\theta \in \Theta} L(\theta : D). \quad (A.2) \]

where \( \Theta \) is a set of parameters.

Equation A.2 means that in the set of parameters, a maximum likelihood estimate or parameter which maximizes the likelihood function is found. We can think of various types of distribution for the observations \( D \) in the experiment. If we consider an RV \( X \) for the observations \( D \) with the multinomial distribution, we can have the following equation which is the maximum likelihood estimator for the parameter \( \theta \).

\[ \theta_k^* = \frac{C[x_k]}{\sum_{q=1}^{N} C[x_q]}, \quad (A.3) \]

where \( C[.] \) is a function returning the number of times a value \( x_k \in \text{Val}(X) \) in an RV \( X \) appears in \( D \) and \( N = |\text{Val}(X)| \). Note that for a variable \( X \), a function \( \text{Val}(X) \) returns a set of values for \( X \).

For example, suppose that there is an RV \( X \) for an observation \( D_i \). The RV \( X \) contain two values \( x_1 = H \) and \( x_2 = T \) and there is a set of observations \( D = \{H, H, H, T\} \). On a set of observations, the count of the number for \( x_1 \) and \( x_2 \) is observed using the function \( C[.] \). For example, \( C[x_1] = 3 \) and \( C[x_2] = 1 \). Using Equation A.3, we can calculate the maximum likelihood estimates for \( x_1 \) and \( x_2 \) as \( \theta_1^* = 3/4 \) and \( \theta_2^* = 1/4 \), respectively.

We can use MLE for a Bayesian network (BN) to estimate a parameter of an RV in the BN. Suppose that there is an RV \( X_i \) in the BN, \( x_k \) is a value for the RV \( X_i \) (i.e., \( x_k \in \text{Val}(X_i) \)).
Val(X_i)), there is a set of parent RVs for the RV (i.e., Pa(X_i) = U), and u is some instantiation for the set of parent RVs (i.e., u \in Val(U)). If we assume that each RV X_i is the multinomial distribution and the observations associated with the RV X_i are independent and identically distributed, then the maximum likelihood estimator for a value x_k|u in the RV X_i in the BN can be formed as Equation A.4.

$$\theta_{i,x_k|u}^* = \frac{C[x_k, u]}{\sum_{q=1}^{n} C[x_q, u]}$$  \hspace{1cm} (A.4)

where C[x_q, u] is the number of times observation x_q in X and its parent observation u in Val(U) appears in D.

For example, we assume that there are a node X_1 in a BN, Val(X_1) = \{x_1 = T, x_2 = F\}, the set of parent nodes for X_1 (i.e., Pa(X_1) = U = \{U_1\}), and Val(U_1) = \{u_1 = A, u_2 = B\}. Also, there is a data set D = \{D_1 = \{T, A\}, D_2 = \{T, A\}, D_3 = \{F, A\}\}, where the first value in D_k is for X_1 and the second value in D_k is for U_1. If u = u_1 = A, then \(\theta_{1,x_1|u_1}^* = 2/3\).

**A.1.2 Bayesian Parameter Estimation**

For Bayesian approach, we assume that the parameter \(\theta\) in the statistical experiment in Section A.1 is a value of an RV \(\Theta\) (i.e., \(P(\Theta = \theta) = g(\theta)\)). In this setting, we try to draw inference about the RV \(\Theta\) given a set of IID observations \(D = \{D_1, D_2, \ldots, D_n\}\), where n is the number of the observations. We then find a *posterior distribution* of the parameter \(\theta\) given the observations \(D\) (i.e., \(P(\theta \mid D)\)). For this, we can use Bayes' theorem, shown in Equation A.5. Note that the posterior distribution can be used to
compute a *posterior predictive distribution* (or simply a predictive distribution) which is the distribution of a future observation given past observations. The posterior predictive distribution will be discussed later. The following equation shows the posterior distribution.

\[
P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)}, \tag{A.5}
\]

where \(D\) is the current observations and \(P(D) > 0\).

Bayesian inference computes the posterior distribution \(P(\theta \mid D)\) using a prior probability \(P(\theta)\) and a likelihood function \(P(D \mid \theta)\) in Bayes’ theorem. The prior probability \(P(\theta)\) is the probability of a parameter \(\theta\) before the current observations \(D\) are observed. The likelihood function \(P(D \mid \theta)\) (Equation A.1) is the probability of the observations \(D\) given the parameter \(\theta\). In Equation A.5, \(P(D)\) is a marginal likelihood (or a normalizing constant) which is the probability distribution for the observations \(D\) integrated over all parameters (i.e., \(P(D) = \int_\theta P(D \mid \theta)P(\theta)\,d\theta\)). The posterior distribution \(P(\theta \mid D)\) is updated using the prior probability \(P(\theta)\) and the likelihood function \(P(D \mid \theta)\), and this update can be repeated. For example, after applying some observations to Equation A.5, we can have a posterior distribution. This posterior distribution can be regarded as a prior probability. And then given some new observations, we can compute a new posterior distribution.

For the parameter \(\theta\), we can have a hyperparameter \(\alpha\) which influences the parameter \(\theta\) and can be formed as \(\theta \sim P(\theta \mid \alpha)\), where the hyperparameter \(\alpha\) can be a
vector or have sub-hyperparameters. In this setting, the probability for the parameter \( P(\theta) \) can be changed to the probability given the hyperparameter \( P(\theta | \alpha) \). By adding the hyperparameter to Equation A.5, Equation A.6 is derived under some assumptions that (1) the observations are independent of a hyperparameter given the parameter associated with the hyperparameter and (2) the sample space for parameters is a partition.

\[
P(\theta | D, \alpha) = \frac{P(D | \theta) P(\theta | \alpha)}{\int \theta P(D | \theta) P(\theta | \alpha) d\theta}.
\]  
(A.6)

We use this posterior distribution containing the hyperparameter (Equation A.6) to compute the predictive distribution. The predictive distribution is the distribution of a new observation given past observations.

\[
P(D_{\text{new}} | D, \alpha) = \int \theta P(D_{\text{new}} | \theta) P(\theta | D, \alpha) d\theta,
\]  
(A.7)

where \( D_{\text{new}} \) is a new observation and is independent of the past IID observations \( D \) given a parameter \( \theta \).

In Equation A.7, the predictive distribution integrates over all parameters for the new observation and the posterior distributions (Equation A.6). To compute the predictive distribution (Equation A.7), we should deal with the posterior distribution (Equation A.6) first. If there is no closed form expression for the integral in the denominator in Equation A.6, we may need to approximate the posterior distribution. If there is a closed form expression for the integral in the denominator and, the prior
distribution and the likelihood are a conjugate pair, then an exact posterior distribution can be found.

A probability distribution in the exponential family (e.g., normal, exponential, and gamma) has a conjugate prior [Gelman et al, 2014]. We can consider an RV X with a categorical probability distribution. For such a categorical probability distribution, Dirichlet conjugate distribution is commonly used. Using Dirichlet distribution, the predictive distribution will be a compact form [Koller & Friedman, 2009].

\[
P(D_{\text{new}} | D, \alpha) = \frac{\alpha_k + C[x_k]}{\sum_j \alpha_j + \sum_{q=1}^N C[x_q]},
\]

where \( C[.] \) is a function returning the number of times a value \( x_k \in \text{Val}(X) \) in a variable X appears in \( D \) and \( N = |\text{Val}(X)| \), \( \alpha \) is a hyperparameter, and \( \alpha_j \) is a sub-hyperparameter in Dirichlet distribution as shown the following.

\[
\theta \sim \text{Dirichlet} (\alpha_1, \alpha_2, \ldots, \alpha_N) \text{ if } P(\theta) \propto \prod_j^{N} \theta_j^{\alpha_j-1},
\]

where the sub-hyperparameter \( \alpha_j \) is the number of samples which have already happened [Koller & Friedman, 2009].

The Bayesian approach above can be used for BN parameter learning. If a prior distribution for an RV \( X_i \), \( P(\theta^i | \alpha) \), is the Dirichlet prior with a hyperparameter \( \alpha = \{ \alpha_{x_1|u}, \ldots, \alpha_{x_N|u} \} \), then the Dirichlet posterior for \( P(\theta^i | \alpha) \) is \( P(\theta^i | D, \alpha) \) with a
hyperparameter $\alpha = \{ \alpha_{x_1|u} + C[x_1, u], \ldots, \alpha_{x_N|u} + C[x_N, u] \}$, where a value $x_k \in \text{Val}(X_i)$, $u \in \text{Val}(\text{Pa}(X_i) = U)$, and $C[x_q, u]$ is the number of times observation $x_q$ in $X_i$ and its parent observation $u$ in $\text{Val}(U)$ appears in $D$. Using the Dirichlet posterior, we can derive the predictive distribution for a value of $X_i$ in a BN under some assumptions: (1) local parameter independences and (2) global parameter independences [Heckerman et al., 1995].

$$P(X_i = x_k | U = u, D, \alpha) = \frac{\alpha_{x_k|u} + C[x_k, u]}{\sum_{q=1}^{N}(\alpha_{x_q|u} + C[x_q, u])},$$  \hspace{1cm} (A.10)

where $N = |\text{Val}(X_i)|$.

Equation A.10 shows the posterior predictive distribution for the value $x_k$ of the $i$-th RV $X_i$ in the BN given a parent value $u$, the observations $D$, and a hyperparameter $\alpha$ for Dirichlet conjugate distribution.
Appendix B. PSAW Questions

Roy [2001] proposed a broad spectrum of questions to be answered in situation analysis (e.g., situation element acquisition, data alignment and association, situation element perception refinement, situation element contextual analysis, situation element interpretation, situation classification/recognition, situation assessment, situation element projection, impact assessment, situation watch, and process refinement [Roy, 2001]). The following PSAW questions are adapted from the ones proposed by [Roy, 2001].

**Situation element acquisition**

(1) Does a (grouped) target exist?

(2) Is the (grouped) target identical to the previous detected (grouped) target?

(3) What are the environmental conditions?

**Data alignment and association**

(1) Is the target's information aligned with the common spatiotemporal space?

(2) Do the information items pertain to the same target or different targets?

**Situation element perception refinement**

(1) How many targets are moving?

(2) What size is the target?

(3) What type is the target?

(4) What group does the target belong to?

(5) What property of the target is not detected?

**Situation element contextual analysis**
(1) What is a relationship between the targets?
(2) What the grouped target is composed of?
(3) What causal relationships exist between the targets?
(4) How are activities related to each other?

**Situation element interpretation**

(1) What is the target doing?
(2) How often does an event take place?
(3) What are the aim, goal, objective, plan, and purpose of the target?

**Situation classification/recognition**

(1) What is the situation in which the targets are involved?
(2) What is the previous or next situation from the current situation?
(3) Has the situation existed previously?

**Situation assessment**

(1) How important is the situation?
(2) How dangerous is the situation?
(3) What is a remarkable event in the situation?

**Situation element projection**

(1) Where are the (grouped) targets located in the future?
(2) What are the next activities performed by the (grouped) targets?
(3) When will be the activities from the (grouped) targets done?
Appendix C. Metrics for Prediction Accuracy

In MEBN, there are two types of random variable (discrete and continuous). For prediction using a MEBN model, both random variables are measured. For a discrete random variable, we can use the Brier score for a multi-category forecast, where perfect prediction yields a score of zero, and a sure prediction on the wrong hypothesis yields a score of two [Brier, 1950].

\[
BS = \frac{1}{n} \sum_{j=1}^{r} \sum_{i=1}^{n} (f_{ij} - E_{ij})^2
\]  
\hspace{1cm} (C.1)

where \( n \) denotes the number of instances of all classes, \( r \) denotes the number of possible classes, \( f_{ij} \) denotes the probability for prediction, and \( E_{ij} \) denotes a function that returns 1, the event in the class \( j \) occurred, or 0, otherwise.

For a continuous random variable, we can use the continuous ranked probability score (CRPS), where perfect prediction yields a score of zero [Gneiting & Raftery, 2007]. Traditionally, for prediction accuracy metrics, we can use a mean absolute error (MAE). MAE uses a mean value only to compare an actual (or observed) value with a predictive value, while CRPS uses a predicted probability distribution for comparison (i.e., a mean and a variance). Therefore, CRPS analysis is more precise than MAE analysis. The following definitions are taken from [Gneiting & Raftery, 2007].
\[ CRPS(F, x) = \int_{-\infty}^{\infty} (F(x) - \mathbb{1}\{y \geq x\})^2 dy \]

where \( F \) denotes the cumulative distribution function of a predictive distribution, \( x \) denotes the actual value, and \( \mathbb{1} \) denotes a heaviside step function that returns 1, if \( y \geq x \), or 0, otherwise.
Appendix D. Local Probability Description Language (LPDL)

The Local Probability Distribution (LPD) specifies numerical probability information for resident random variables of an MFragment. It is a kind of function which should represent discrete and continuous probability distributions. An LPD can be described by a Local Probability Description Language (LPDL).

Grammar of LPDL

The following list shows the grammar of LPDL.

```
table ::= statement | if_statement
if_statement ::= "if" constraintSet "have" "(" b_expression ")" statement "else" else_statement
allop ::= "some" | "all"
varsetname ::= ident ["," ident]*
constraint ::= allop varsetname
constraintSet ::= constraint (constraint)*
b_expression ::= b_term [ "|" b_term ]*
b_term ::= not_factor [ "&" not_factor ]*
not_factor ::= [ "~" ] b_factor
b_factor ::= ident "=" ident | "(" b_expression ")"
else_statement ::= statement | if_statement
statement ::= "[" assignment_or_if "]"
assignment_or_if ::= assignment | if_statement
assignment ::= ident "=" expression [ "," assignment ]* | expression ( ',' assignment )*expression ::= term [ addop term ]*
term ::= signed_factor [ mulop signed_factor ]*
signed_factor ::= [ addop ] factor
factor ::= number | function | "(" "expression ")"
function ::= possibleVal
| "CARDINALITY" "(" varsetname ")"
| "MIN" "(" expression "," expression ")"
| "MAX" "(" expression "," expression ")"
| "Mean" "(" expression "," expression ")"
| "Sum" "(" expression "," expression ")"
| "Multiply" "(" expression "," expression ")"
possibleVal ::= ident
addop ::= "+" | "-"
mulop ::= "+" | "-"
ident ::= letter [ letter | digit ]*
```
An example of LPDL.

The following list shows an example of LPDL.

```plaintext
if any obj have ( ObjectType = Tracked ) [  
  if any rgn have ( Weather = Clear ) [  
    Tracked = .8, Wheeled = .15, NonVehicle = .05  
  ] else [ Tracked = .6, Wheeled = .3, NonVehicle = .1 ]  
] else if any obj have ( ObjectType = Wheeled ) [  
  if any rgn have ( Weather = Clear ) [  
    Tracked = .1, Wheeled = .8, NonVehicle = .1  
  ] else [ Tracked = .2, Wheeled = .6, NonVehicle = .2 ]  
] else [  
  if any rgn have ( Weather = Clear ) [  
    Tracked = .05, Wheeled = .05, NonVehicle = .9  
  ] else [ Tracked = .15, Wheeled = .15, NonVehicle = .7 ]  
]```
Appendix E. Conditional Probability Script Language (CPSL)

The Conditional Probability Script Language (CPSL) specifies a discrete and continuous probability distribution for a random variable of a BN.

Grammar of CPSL

The following list shows the grammar of CPSL.

```plaintext
grammar Distribution;

options {
    output = AST;
}

@header{
    package UMath;
}

@lexer::header{
    package UMath;
}

program : statement* ;

block : '{' blockStatement* '}' -> ^(SLIST blockStatement*) ;

blockStatement :
    statement ;

statement :
    block
    | 'if' ifExpression statement (options {k=1;}:'else'! statement)?
    | 'print' arguments? ';' -> ^( 'print' arguments )
    | 'return' expression? ';
    | 'throw' expression ';' 
    | 'break' Identifier? ';
    | 'continue' Identifier? ';
    | '!' statementExpression ';
    | Identifier '!' statement ;
```

217
statementExpression
   : expression

constantExpression
   : expression

ifExpression
   : '(' expression ')' -> ^(EXP expression)

parExpression
   : '!' expression '!'!

expressionList
   : expression (','! expression)*

arguments
   : '!' expressionList? '!'!

expression
   : conditionalExpression (assignmentOperator^ expression)?

expression2
   : conditionalExpression (assignmentOperator! expression)?

assignmentOperator
   : '='
   | '+='
   | '-=
   | '*='
   | '/='
   | '&='
   | '|='
   | '^='
   | '%='

type : primitiveType ('[' ']')*

primitiveType
   : 'boolean'
   | 'char'
   | 'byte'
| 'short' |
| 'int' |
| 'long' |
| 'float' |
| 'double' |
| 'number' |
| 'string' |

conditionalExpression
   : conditionalOrExpression ( '?' conditionalExpression ':' conditionalExpression )?
   ;

conditionalOrExpression
   : conditionalAndExpression ( "||"^ conditionalAndExpression )* 
   ;

conditionalAndExpression
   : inclusiveOrExpression ( "&&"^ inclusiveOrExpression )* 
   ;

inclusiveOrExpression
   : exclusiveOrExpression ( "|"^ exclusiveOrExpression )* 
   ;

exclusiveOrExpression
   : andExpression ( "^"^ andExpression )* 
   ;

andExpression
   : equalityExpression ( "&"^ equalityExpression )* 
   ;
equalityExpression
   : relationalExpression ( "==" | "!=" )^ relationalExpression )* 
   ;

relationalExpression
   : additiveExpression ( relationalOp^ additiveExpression )* 
   ;

relationalOp
   : '!='
   | '=='
   | '<=
   | '>'
   | '>
   ;
elementValue
   : conditionalExpression 
   | elementValueArrayInitializer
additiveExpression
   : multiplicativeExpression ( ('+' | '-')^ multiplicativeExpression )* 
;

multiplicativeExpression
   : unaryExpression ( ('*' | '/' | '%')^ unaryExpression )* 
;

unaryExpression
   : '+' unaryExpression 
   | '-' unaryExpression 
   | '++' unaryExpression 
   | '--' unaryExpression 
   | unaryExpressionNotPlusMinus 
;

unaryExpressionNotPlusMinus
   : '~' unaryExpression 
   | '!' unaryExpression 
   | primary { System.out.println($primary.text ); } 
;

elementValueArrayInitializer
   : '{' (elementValue (',' elementValue)*)? (',')? '}' 
;

literal
   : integerLiteral 
   | FloatingPointLiteral 
   | booleanLiteral 
   | 'null' 
;

primary
   : parExpression | 
   'this' ('.' Identifier)* identifierSuffix? 
   | literal 
   | createdName 
   | STRING_LITERAL 
   | Identifier^ arguments 
   | Identifier (',' Identifier)* identifierSuffix? 
   | ('[' ']')* '.' 'class' 
   | 'void' '.' 'class' 
;

createdName
   : primitiveType expression2 -> ^(VAR primitiveType expression2 ) 
;

identifierSuffix
integerLiteral
  : DecimalLiteral -> DecimalLiteral ;

booleanLiteral
  : 'true' -> 'true'
  | 'false' -> 'false' ;

EXP : 'EXP' ;
VAR : 'VAR' ;
SLIST : 'SLIST' ;

DecimalLiteral : ('0' | '1'..'9' '0'..'9'*) ;

FloatingPointLiteral
  : ('0'..'9')+ '.' ('0'..'9')* ;

STRING_LITERAL
  : "" ( EscapeSequence | ~('\'|'^') ) * "" ;

Identifier
  : Letter (Letter|JavaIDDigit)* ;

fragment
Letter
  : 'u0024'
  | 'u0041'..'u005a' | 'u005f' |
  | 'u0061'..'u007a' | 'u00c0'..'u00d6' |
  | 'u00d8'..'u00f6' | 'u00f8'..'u00ff' |
  | 'u0100'..'u1ff' |
  | 'u3040'..'u318f' |
  | 'u3300'..'u337f' |
  | 'u3400'..'u3d2d' |
  | 'u4e00'..'u9fff' |
  | 'uf900'..'ufaff' ;

fragment
JavaIDDigit
  : 'u0030'..'u0039'
  | 'u0060'..'u0069' |
  | 'u06f0'..'u06f9' ;
Example of CPSL

Followings show an example of a CPSL.
defineNode(A, DescriptionOfA);
 {  
  defineState(Discrete, a1, a2);
  P(A)={a1:0.8; a2:0.2}
 }
defineNode(X, DescriptionOfX);
 {  
  defineState(Continuous);
  P(X)=NormalDist(-3, 1);
 }
defineNode(Y, DescriptionOfY);
 {  
  defineState(Continuous);
  P(Y)=NormalDist(3, 0.5);
 }
defineNode(W, DescriptionOfW);
 {  
  defineState(Continuous);
  P(W | A,X,Y)=if(A==a1) { 
    X - 0.5*Y + NormalDist(2, 1);
  } else if(A==a2) { 
    X + Y + NormalDist(-1, 0.5) ;
  }
}
Appendix F. An HMLP Tool

We developed an HMLP Tool (Fig. F.1) that performs MEBN-RM and the MEBN parameter learning. The HMLP Tool is a JAVA based open-source program\textsuperscript{13} that can be used to create an MTheory script from a relational schema. This enables rapid development of an MTheory script from a relational database by just clicking a button in the tool. The current version of the HMLP Tool uses MySQL for the relational database. The most recent version and source codes of the HMLP Tool are available online at https://github.com/pcyoung75/GMU_HMLP.git. Also, source codes for the HMLP Tool are in the GMU_HMLP Github repository.\textsuperscript{14}

\textsuperscript{13}Researchers around the world can debug and extend the tool.

\textsuperscript{14}Github is a distributed version control system (https://github.com).
The HMLP Tool which contains three panels: (1) a left tree panel shows a list of relational database, (2) a right top panel shows a result MTheory script, and (3) a right bottom panel shows an input window in which we can insert some information. The following figure shows the interface of the HMLP Tool and a result MTheory script using the tool.

The current version of the HMLP Tool only supports context nodes containing the “IsA” property. We remain the use of context nodes representing sophisticated relational patterns as future work. The steps not yet implemented on the HMLP tool, namely “Create World Model”, “Set Joining Condition” and “Class Local Distribution”, are also interesting features to be added in the future. These steps enrich the process as the user can manually edit parts of the MTheory before the final learning phase. In this context, being able to go back from one step to another on the wizard is another useful feature to be implemented. Additionally, the option to more generally set connections to different relational databases (e.g., MS SQL, Oracle DB, and PostgreSQL) is also a promising future work.
Appendix G. Use Case 1: Human-Aided MEBN Learning for HERALD and Test Results

In this appendix, we introduce a use case of HMLP for the HERALD MTheory. In this process, an MTheory developer in charge of the development of the HERALD MTheory cooperates with a stakeholder who requests the development of the HERALD system. They design, develop, and evaluate the HERALD MTheory based on HMLP. The following subsections show how HMLP for the HERALD MTheory is performed.

6.1 Analyze Requirements
HERALD focuses on threats from terrorists and UAVs approaching the critical infrastructure (CI). In Section 6.1, we discussed the operational scenarios for the red and blue team. From the operational scenarios, we developed goals of reasoning models with the stakeholder. Two goals were developed: Goal 1 was to identify threatening targets and Goal 2 was to recognize the emergency situation for the critical infrastructure. From these goals, we identified queries and evidence required for the queries to achieve the goals. The following list shows the goals, queries, and evidence.

**Goal 1:** Identify threatening targets
- **Query 1.1:** What is the type of the target?
- **Query 1.2:** Where will be the target located?
- **Query 1.3:** What type activity will be the target performing?
- **Query 1.4:** What mission does the target have?

**Evidence 1.1.1 ~ 1.4.1:** Some reports from the sensor systems in HERALD

To identify threatening targets, the above four questions were chosen and they should be answered using some reports from the sensor systems (e.g., the MTI system and the IMINT sensor system) in HERALD.
Goal 2: Recognize emergency situation for the critical infrastructure

Query 2.1: How high is the level of danger to the critical infrastructure?

Evidence 2.1.1: Some reports from the sensor systems in HERALD

To recognize an emergency situation for the critical infrastructure, the level of danger to the critical infrastructure should be answered using reports from the sensor systems in HERALD.

The criteria in the requirements were specified in terms of some measure of accuracy. There are two types of random variable (discrete and continuous). For a discrete random variable, we used the Brier score for a multi-category forecast, where perfect prediction yields a score of zero, and a sure prediction on the wrong hypothesis yields a score of two [Brier, 1950], as a metric for a performance criterion. For a continuous random variable, we used the continuous ranked probability score (CRPS), where perfect prediction yields a score of zero [Gneiting & Raftery, 2007] (these metrics are defined in Appendix C). Generally, a threshold value for each performance criterion is determined by an acquirer or a stakeholder, who orders the development of the system, during the stage of the requirement engineering [Roedler & Jones, 2005]. In the quick adaptation paradigm, the threshold value can be changed after testing the operation of the system and actual applying for the system in the real world. Initially, we chose some threshold values for HERALD. The performance criterion for the discrete random variable was that each average Brier score for reasoning results from the HERALD MTheory should be less than 0.5, while the performance criterion for the continuous random variable was that each average CRPS for reasoning results from the HERALD MTheory should be less than 0.1. The perfect estimation and prediction for these scores
yield a score of zero, so our stakeholder may want the score of zero the both cases (Brier score and CRPS). However, this is too ideal to achieve such a performance, since noisy data in the training dataset is unavoidable. Therefore, these threshold values for this proof of concept system were determined by a discussion with our stakeholder.

G.2 Define World Model

This step contains two sub-steps (Define Structure Model and Define Rules). The first sub-step was to define a structure model for HERALD from the requirements, domain knowledge and/or existing data schemas. The PSAW-MEBN reference model in Chapter 4 could be used in this step to identify possible entities, variables, and relationships. The reference model provided information about possible entities (i.e., $T$, $OR$, $SR$, $TR$, and $RT$). Entities derived from these categories could have various attributes (e.g., an attribute speed for a target $tr1$) and be related to other entities (e.g., a relation for a communication between a target $tr1$ and a target $tr2$). Fig. G.1 extended from Fig. H.2 (Appendix H) shows the HERALD database schema with some rules. In the figure, the dashed line rectangles denote relations which contain at least two attributes and the non-dashed line rectangles denote relations which contain only one attribute. A relation in the dashed line rectangle can be converted to an MFragment and the attributes in the relation can be random variables (RVs) in MEBN. For example, the attribute $\text{DistanceToSensor}$ in the relation $\text{SensorTemporalProperty}$ can be a random variable $\text{DistanceToSensor}(tr, sr, t)$ with three ordinary variables $tr$ (for a target entity), $sr$ (for a sensor entity), and $t$ (for a time entity).
The relations in Fig. G.1 can be classified by five types of MFrags in the PSAW-MEBN reference model (i.e., Observing Condition, Reported Object, Target Object, Situation, and Context).

**Observing Condition**
1. SensorTemporalProperty Relation

**Target Object**
2. Target Relation, 3. TargetTemporalProperty Relation

**Reported Object**

**Situation**
7. CI_Situation Relation

**Context**

For example, the relation SensorTemporalProperty can belong to the Observing Condition MFragment. The relation MTI_Report, IMINTS_Report, and GEOINTS_Report can belong to the Reported Object MFragment. The relations Target and TargetTemporalProperty (e.g., attributes TargetType and Latitude) can belong to the Target Object MFragment. The relation Situation can belong to the Situation MFragment. The remaining relations (e.g., SensorOf and Predecessor), which are used to join the relations, can belong to the Context MFragment. The following list shows possible MFrags classified by the PSAW-MEBN reference model.

**Observing Condition Group**

1. SensorTemporalProperty Relation
   RV: {DistanceToSensor, SensorTemporalProperty}

**Target Object Group**

2. Target Relation
   RV: {TargetType, TargetSize, TargetImage}

3. TargetTemporalProperty Relation
   RV: {Latitude, Longitude, Altitude, Latitude_Velocity, Longitude_Velocity, Altitude_Velocity, DistanceToCI, DirectionToCI, RegionType, Temperature, Activity, Mission, TargetTemporalProperty}

**Reported Object Group**

4. MTI_Report Relation
   RV: {LatitudeReport, LongitudeReport, AltitudeReport, DistanceToCIReport, DirectionToCIReport, MTI_Report}

5. IMINTS_Report Relation

6. GEOINTS_Report Relation
   RV: {RegionTypeReport, GEOINTS_Report}

**Situation Group**

7. CI_Situation Relation
   RV: {DangerLevel}

**Context Group**

8. SensorOf Relation
   RV: {SensorOf}

9. ReportedTarget_MTIRPT Relation
The PSAW-MEBN reference model provides knowledge about causal relationships between variables. For example, a Reported Object RV depends on a Target Object RV. A Situation RV depends on a Target Object RV. Using the reference model, we derived the causal relationships between variables as shown the red arrow lines in the Fig. G.1. These causal relationships were used to identify rules later.

Also, the PSAW-MEBN reference model provides knowledge about context variable types (i.e., ObserverOf, ActualObject, and Predecessor). The relation SensorOf can be the variable SensorOf which can belong to the context type ObserverOf. The attributes ActualTarget in the relations ReportedTarget_IMINTSRPT, ReportedTarget_MTIRPT, and ReportedTarget_GEOINTRPT can be the variable ActualObject which can belong to the context variable type ActualObject. The relation Predecessor can be the variable Predecessor belonging to the context type Predecessor.

From the above discussion, we derived the RVs, the causal relationships for the RVs, and the context variables. These are listed in the following rules and contexts. Note that the notations for causality in the following list can be found in Section 5.2.2.

**Report Object Group**

4. **MTI Report Relation**

**Context:** SensorOf & ReportedTarget_MTIRPT

- **Rule 4.1:** causal({DistanceToSensor, Latitude}, LatitudeReport)
- **Rule 4.2:** causal({DistanceToSensor, Longitude}, LongitudeReport)
- **Rule 4.3:** causal({DistanceToSensor, Altitude}, AltitudeReport)
Rule 4.4: causal({DistanceToSensor, DistanceToCI}, DistanceToCIReport)
Rule 4.5: causal({DistanceToSensor, DirectionToCI}, DirectionToCIReport)

5. IMINTS_Report Relation
   Context: SensorOf & ReportedTarget.IMINTSRPT
   Rule 5.1: causal({DistanceToSensor, Image}, ImageReport)
   Rule 5.2: causal({DistanceToSensor, Temperature}, TemperatureReport)
   Rule 5.3: causal({DistanceToSensor, Size}, SizeReport)

6. GEOINTS_Report Relation
   Context: SensorOf & ReportedTarget.GEOINTRPT
   Rule 6.1: causal({DistanceToSensor, RegionType}, RegionTypeReport)

The above rules and contexts were directly derived from the PSAW-MEBN reference model. However, they are not enough to design the HERALD MTheory. A domain expert provided more information for rules and contexts. For example, in the relation Target, the RV TargetType can influence the RVs TargetImage and TargetSize, so Rules 2.1 and 2.2 were derived. In the relation TargetTemporalProperty, the RV Activity can influence the RVs DistanceToCI and DirectionToCI, so Rules 3.1 and 3.2 were derived. Also, there is a temporal assumption in which each RV in the relation TargetTemporalProperty depends on a previous RV. For example, the RV Temperature is influenced by the RVs TargetType and Activity as well as a previous RV Temperature, called Pre_Temperature. This temporal assumption is called a Markov assumption. If an RV at time n only depends on another RV at time n-1, we call it a first-order Markov assumption. The HERALD MTheory is based on a first-order Markov assumption (e.g., Rule 3.1 ~ Rule 3.12 in the following list). The rules and contexts developed from expert's knowledge are listed in the following.

Target Object Group

2. Target Relation
   Rule 2.1: causal(TargetType, TargetImage)
   Rule 2.2: causal(TargetType, TargetSize)

3. TargetTemporalProperty Relation
Context: Predecessor

Rule 3.1: causal(\{TargetType, Activity, RegionType, Pre_Latitude_Velocity\},
\{Latitude, Velocity\})

Rule 3.2: causal(\{TargetType, Activity, RegionType, Pre_Longitude_Velocity\},
\{Longitude, Velocity\})

Rule 3.3: causal(\{TargetType, Activity, RegionType, Pre_Altitude_Velocity\},
\{Altitude, Velocity\})

Rule 3.4: causal(\{Pre_Latitude_Velocity, Pre_Latitude\}, Latitude)

Rule 3.5: causal(\{Pre_Longitude_Velocity, Pre_Longitude\}, Longitude)

Rule 3.6: causal(\{Pre_Altitude_Velocity, Pre_Altitude\}, Altitude)

Rule 3.7: causal(\{Activity, Pre_DistanceToCI\}, DistanceToCI)

Rule 3.8: causal(\{Activity, Pre_DirectionToCI\}, DirectionToCI)

Rule 3.9: causal(\{TargetType, Activity, Pre_Temperature\}, Temperature)

Rule 3.10: causal(\{Mission, Pre_Activity\}, Activity)

Rule 3.11: causal(\{TargetType, Pre_Mission\}, Mission)

Rule 3.12: causal(\{Pre_RegionType\}, RegionType)

Situation Group

Situation Relation

Rule 7.1: causal(\{TargetType, Mission\}, DangerLevel)

After defining these rules, we can choose the type of a local probability
distribution for each possible RV (e.g., Conditional Gaussian distribution). For a discrete
random variable (e.g., Mission and TargetType), we can use a Categorical
distribution. For a continuous random variable (e.g., Latitude_Velocity and Latitude), we can use a
Conditional Gaussian distribution which is shown in the following.
Fig. G.2 shows a Hybrid BN with Conditional Gaussian distributions in which there is a discrete RV Type with two states ((h)ead and (t)ail)) and there are four continuous RVs $X_{1-2}$ and $R_{1-2}$. The RV $R_k$ is for a report for the RV $X_k$. According to the RV Type, the RV $X_k$ is determined. Once the RV Type is the state $h$, the RV $X_1$ can be a Conditional Gaussian distribution with a mean $m_1$ and a conditional variance $s_1$, and the RV $X_2$ can be a Conditional Gaussian distribution with a mean $c_1 X_1 + m_2$ and a conditional variance $s_2$. The $m_2$ is a regression intercept and $c_1$ is a regression coefficient. Also, the RV $R_k$ can be a Conditional Gaussian distribution with a mean $c_2 X_1 + m_3$ and a conditional variance $s_3$. These continuous RVs are (*common*) Conditional Gaussian distributions. In some cases, we need a special type of Conditional Gaussian distribution as shown Table G.1 (e.g., a Deterministic-Conditional Gaussian distribution, an Only-Variance-Conditional Gaussian distribution, and a Uniform Gaussian distribution).
### Table G.1 Gaussian Distribution Types

<table>
<thead>
<tr>
<th>Distribution Type</th>
<th>Regression intercept</th>
<th>Conditional variance</th>
<th>Regression coefficient</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional Gaussian</td>
<td>$m$</td>
<td>$\sigma^2$</td>
<td>$b_i$</td>
<td>$\mathcal{N}(m + b_i P_1, \sigma^2)$</td>
</tr>
<tr>
<td>Deterministic-Conditional Gaussian</td>
<td>0</td>
<td>very small value</td>
<td>1</td>
<td>$\mathcal{N}(P_1, 0.0001)$</td>
</tr>
<tr>
<td>Only-Variance-Conditional Gaussian</td>
<td>0</td>
<td>$\sigma^2$</td>
<td>1</td>
<td>$\mathcal{N}(P_1, \sigma^2)$</td>
</tr>
<tr>
<td>Uniform Gaussian</td>
<td>0</td>
<td>very large value</td>
<td>0</td>
<td>$\mathcal{N}(0, 10000)$</td>
</tr>
</tbody>
</table>

The Deterministic-Conditional Gaussian distribution is a Conditional Gaussian distribution whose regression intercept is zero, regression coefficient is one, and variance is a very small value. The Only-Variance-Conditional Gaussian distribution is a Conditional Gaussian distribution whose regression intercept is zero and regression coefficient is one. The Uniform Gaussian distribution is a Conditional Gaussian distribution whose mean is zero and variance is a very large value. Such special distributions can be used for a simple Kalman filter model. For example, Fig. G.3 shows an illustrative example of the simple Kalman filter model in BN. The reason why we distinguish these conditional Gaussian distributions is that the MEBN parameter learning requires specific information for how to use dataset for the MEBN parameter learning. For example, for the Only-Variance-Conditional Gaussian distribution, a dataset is used for calculating only a variance, but not a mean.
Fig. G.3 shows a Hybrid BN for a simple Kalman filter model in which there is a discrete RV *Type* with two states (*head* and *tail*) and there are eight continuous RVs $V_1$, $X_2$, $X_3$, and $R$. The RV $V_k$ is for a velocity of the RV $X_{k+1}$ and the RV $R_k$ is for a report for the RV $X_k$. According to the RV *Type*, the velocity $V_k$ is determined. Once the RV *Type* is the state *head*, the RV $V_1$ can be a Conditional Gaussian distribution with a mean $m_1$ and a variance $s_1$, and the RV $V_2$ can be an Only-Variance-Conditional Gaussian with a mean $V_1$ and a variance $s_1$. The RV $X_1$ can be a Uniform Gaussian distribution with a mean 0 and a large variance $s_2$, while the RV $X_3$ can be a Deterministic-Conditional Gaussian distribution with a mean $X_2 + V_2$ and a small variance $s_3$. The RV $R_1$ can be an Only-Variance-Conditional Gaussian distribution with a mean $X_1$ and a variance $s_4$. The reason why we distinguish these Conditional Gaussian distributions is that according to their type, a different parameter learning should be used. Table G.2 shows a CLD type for each RV.
Table G.2 Possible CLD Types for Possible RVs

<table>
<thead>
<tr>
<th>Node Type</th>
<th>CLD Type</th>
<th>Random Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete</td>
<td>Categorical</td>
<td>Mission, Activity, TargetType, TargetSize, TargetImage, RegionType, ImageReport, RegionTypeReport</td>
</tr>
<tr>
<td>Continuous</td>
<td>Conditional Gaussian</td>
<td>DistanceToCI, DirectionToCI, Temperature</td>
</tr>
<tr>
<td></td>
<td>Uniform Gaussian</td>
<td>Latitude at t1, Longitude at t1, Altitude at t1</td>
</tr>
<tr>
<td></td>
<td>Deterministic-Conditional Gaussian</td>
<td>Latitude at not t1, Longitude at not t1, Altitude at not t1</td>
</tr>
<tr>
<td></td>
<td>DangerLevel-CLD</td>
<td>DangerLevel</td>
</tr>
</tbody>
</table>

The CLD for the RV DangerLevel uses a special CLD called a DangerLevel-CLD. The RV DangerLevel depends on the pair RVs (TargetType and Mission). The following shows the DangerLevel-CLD used for the RV DangerLevel.

CLD G.1 : DangerLevel(ci, t)
1    if some g have ( TargetType = ThreateningAirTarget or
2        TargetType = ThreateningGroundTarget ) [
3        if some g.t have ( Mission = Attack ) [
4            1 - 1 / EXP(CARDINALITY(g) /10) + NormalDist(0, 0.00000001)
5        ] else [
6            NormalDist(0, 0.00000001)
7        ]
8    ] else [
9        NormalDist(0, 0.00000001)
10  ]

To understand this more intuitively, Fig. G.4 is used to show an increasing aspect of the exponential function in the CLD for the RV DangerLevel. If there is no threatening target which has an attacking mission, the danger level is zero. As increasing the number of the threatening target, the danger level increases up to 1.
The DangerLevel-CLD is given by expert's knowledge and will be not learned from a dataset, because in practice it is very difficult to obtain such a dataset and in most cases, such a measure (i.e., the danger level) is given by a subjective judgment.

![Figure G.4 Danger Level over the Number of Threatening Targets](image)

In this subsection, we derived possible MFrags, RVs, CLDs, and rules for the RVs using the PSAW-MEBN reference model and expert's knowledge. In the next subsection, these derivatives will be used to construct an initial MTheory using MEBN-RM and the initial MTheory will be learned by the MEBN learning.

**G.3 Construct Reasoning Model**

This step consists of two sub-steps (*Map to Reasoning Model* and *Learn Reasoning Model*) to construct a reasoning model from a training dataset. By MEBN-RM, an entity relation which contained only one attribute for the primary key of the relation (e.g., *sensor* and *target*) could be defined as an entity type in MEBN. Thus, there are 7 entity types (*sensor*, *target*, *time*, *criticalinfrastructure*, *reportedtarget_mtirpt*, *reportedtarget_imintsrpt*, and *reportedtarget_geointsrpt*). Also, each of the attributes in the relations can be mapped to a resident node in MEBN using MEBN-RM. For example,
the attribute \textit{Activity} of the relation \textit{targettemporalproperty} became the resident node \textit{Activity}(tr, t) with a target ordinary variable \textit{tr} and a time ordinary variable \textit{t}. The attribute \textit{DangerLevel} of the relation \textit{ci\_situation} became the resident node \textit{DangerLevel}(ci, t) with a critical infrastructure ordinary variable \textit{ci} and a time ordinary variable \textit{t}.

Rules which were defined in the previous step were used to develop conditional dependence between resident nodes. For example, from Rule 7.1, a conditional dependence \( P(\textit{DangerLevel} \mid \textit{TargetType}, \textit{Mission}) \) could be considered and this causal relationship could be represented in an initial MTBey for HERALD.

Fig. G.5 shows an MFrag, called a \textit{CI Situation} MFrag in the possible HERALD MTBey. Note that this MFrag is a part of the HERALD MTBey, which contains several MFrags (e.g., \textit{MTI\_Report}, \textit{IMINTS\_Report}, and \textit{GEOINTS\_Report}).

The MFrag (Fig. G.5) represents probabilistic knowledge of critical infrastructure situations involving a group of targets. It contains three \textit{isa} context nodes for entities \textit{Target}, \textit{CriticalInfrastructure}, and \textit{Time}, two input nodes \textit{TargetType} (with states \textit{Others}, \textit{Mission})
GroundThreateningTarget, and AirThreateningTarget) and Mission (with states Others and Attacking), and one continuous resident node DangerLevel. The continuous resident node DangerLevel(ci, t) has a critical infrastructure ordinary variable ci and a time ordinary variable t as arguments. It indicates whether the situation for the group of targets in the times t is dangerous if the value is 1 or safe if the value is 0. The resident node DangerLevel(ci, t) is influenced by the input node TargetType(tr) with a target ordinary variable tr and Mission(tr, t) with the target ordinary variable tr and the time ordinary variable t. Like the CI Situation MFragment, remaining MFrags could be constructed using MEBN-RM and updated using the rules as shown the following list in MTheory G.1.

Note that full MFrags and a graphical representation for the HERALD MTheory can be found in Appendix I.

**MTheory G.1: Partial MFrags for Updated HERALD MTheory**

```plaintext
[F: SensorTemporalProperty
  [C: Isa(tr, TARGET), Isa(sr, SENSOR), Isa(t, TIME)]
  [R: DistanceToSensor(tr, sr, t)]
]

[F: Target
  [C: Isa(tr, TARGET)]
  [R: TargetType(tr)]
  [R: TargetSize(tr)]
  [RP: TargetType(tr)]
]

[R: TargetImage(tr)
  [RP: TargetType(tr)]
]

...]

[F: Situation
  [C: Isa(tr, TARGET), Isa(t, TIME), Isa(ci, CRITICALINFRASTRUCTURE)]
  [R: DangerLevel (ci, t)]
  [IP: Mission(tr, t), TargetType(tr)]
]
```

In the Learn Reasoning Model step, the updated HERALD MTheory was refined using a MEBN learning algorithm from a training dataset. For the HERALD MTheory,
we used a parameter learning given the updated HERALD MTheory. The training dataset was generated by the HERALD scenario simulator. The HERALD scenario simulator simulated ground truth information of a situation in which blue and red teams operated against each other. The Red Team could be a group of terrorists or UAVs (unmanned aerial vehicles). The Red Team aimed to destroy a critical infrastructure element defended by the Blue Team. The Blue Team had several types of sensors that could detect threatening targets of interest, and a command center, charged with using data from the sensors to identify the situation and make decisions about future actions. The scenario simulator simulated the operational scenarios in Section 6.1.1. Like the training dataset was generated from the HERALD scenario simulator, a test dataset was also generated from the HERALD scenario simulator to be used for evaluation. These datasets were separately stored in a relational database.

6.4 Test Reasoning Model
This step consists of two sub-steps (Experiment Reasoning Model and Evaluate Experimental Results) to evaluate a reasoning model from the test dataset. In the Experiment Reasoning Model step, the performance of estimation and prediction for the learned HERALD MTheory was assessed using some measures (e.g., accuracy, sensitivity, specificity, AUC, and Brier score). These measures indicate how much the learned HERALD MTheory fits well a ground truth. For example, if the performance of prediction assessed by the Brier score for a multi-category forecast yields a score of zero, this indicates that the learned HERALD MTheory fits well the ground truth. In this proof
of concept system, we used the simulated test dataset as ground truth. The development using an actual dataset for ground truth remains for the future research.

We conducted five experiments for five queries (i.e., Query 1.1, Query 1.2, Query 1.3, Query 1.4, and Query 2.1): (1) The type of the target, (2) the location of the target, (3) the activity of the target, (4) the mission of the target, and (5) the level of danger of the critical infrastructure. Each experiment consisted of the following five steps. (1) The test dataset provided entity information (e.g., target1, sensor1, and time1) and simulated ground truth information (e.g., ImageReport = Unknown, TemperatureReport = 90, SizeReport = 6) to the learned HERALD MTheory. (2) Given these, a marginal probability of a query (e.g., Query 2.1: P(TargetType | ImageReport = Unknown, TemperatureReport = 90, SizeReport = 6) was calculated using the learned HERALD MTheory. (3) The test dataset provided a ground truth data (e.g., TargetType = ThreateningTarget). (4) The steps 1-3 were repeated for all testing cases. (5) Finally, for results for all cases, the measures were calculated.

This step can support evaluation of the usefulness of our methodologies. Thus, effectiveness for HMLP is assessed by how well the learned HERALD MTheory fit a test dataset. In this test, the evaluation of the performance of the learned HERALD MTheory is required. We will judge whether the learned HERALD MTheory is adequate for its purpose by measuring the estimation and prediction performance. Explained variation (e.g., Statistics $R^2$ and the continuous ranked probability score) and the Brier score were used as customary statistical approaches for measuring prediction models [Steyerberg et al., 2010]. Explained variation is used for a continuous variable, while the Brier score is
used for a categorical variable. In this research, we used the Brier score [Brier, 1950] for the discrete case and the continuous ranked probability score [Gneiting & Raftery, 2007] for the continuous case (see Appendix C). A good Brier score and a good continuous ranked probability score (i.e., the best possible score is zero) mean a fitness score for the PSAW-MTheory to the test dataset. Note that each confusion matrix for categorical variables can be found in Appendix J.

In the experiment, we used the following experiment factors: (1) SSBNs were generated from the learned HERALD MTheory (i.e., SSBN\_g1\_t2, SSBN\_g1\_t3, SSBN\_g1\_t4 (Appendix K), SSBN\_1g, SSBN\_2g, SSBN\_3g, SSBN\_4g, and SSBN\_5g). The SSBN\_g1\_t4 was an SSBN generated with a target entity \(g1\) and four time entities \(t1, t2, t3,\) and \(t4\). The SSBN\_5g was an SSBN generated with five target entities \(g1, g2, g3, g4,\) and \(g5\). (2) The type of inference algorithm for the experiment was the Hybrid MP with Gaussian Mixture Reduction (HMP-GMR)\(^{15}\) (Appendix O). (3) For all experiments, the convergence criterion for HMP-GMR was \(max\_prs = 10^{-3}\). The maximum allowable number of components was \(nc = 1\). The maximum number of iterations was \(it = 100\). The maximum execution time was \(max\_time = 200000\) millisecond (ms). (4) The number of

\(^{15}\) Reasoning for SSBN is a challenging issue. In PSAW, an SSBN representing a situation of PSAW can be very computationally challenging. For example, a situation may contain thousands of sensors, targets, and time steps as well as these entities can be associated with not only categorical variables but also continuous variables. So, we need a reasoning algorithm which can be used for a complex Hybrid Bayesian Network (HBN), which contains both discrete and continuous variables. In some cases for PSAW, an execution time for reasoning is critical (e.g., early detection of a target). For such a case, we need an anytime reasoning algorithm that can provide a solution even if it is interrupted before completion. HMP-GMR can address such issues.
runs for each experiment was 100. (5) The experiments were run on a 3.40GHz Intel Core i7-3770 processor.

For the five queries, we conducted two types of experiments. The first type was to identify threatening targets over time to see a dynamic aspect of the PSAW situation. In this experiment, estimation and prediction for Query 1.1 ~ 1.4 were tested. The second type was to investigate a situation in which there are multiple targets and to recognize the emergency situation for the critical infrastructure. Thus, estimation for Query 2.1 was answered. In this case, because we didn’t have actual or simulated datasets for emergency situation reflecting reality, accuracy for the inference was not evaluated. We only conducted an experiment in which operational performances (e.g., memory usage, execution time, the limitation for a complex SSBN) of the HMP-GMR algorithm for SSBNs generated from the HERALD MTheory were tested.

G.4.1 Situation for Dynamic Aspects
In this experiment, the accuracy of estimation and prediction for Query 1.1 ~ 1.4 was tested using some accuracy scores (the Brier score, Sensitivity, Specificity, and AUC). Three SSBNs (SSBN_g1_t2, SSBN_g1_t3, and SSBN_g1_t4) were used. For each SSBN, a data set at t1 was extracted from the test dataset and the dataset was used as the report evidence. For SSBN_g1_t2, the prediction at t1 (i.e., the inference result at t2) was evaluated using the accuracy scores.

Before looking at the accuracy results for Query 1.1 ~ 1.4, the performance averages of the inference algorithm are shown Table G.3. The three performance factors (Time, Memory usage, and Iteration) for three times were tested. Time is the execution
time of the inference algorithm. Iteration is the iteration number of the HMP-GMR algorithm (numbers in parentheses are standard deviations). For example, the inference algorithm required Avg. 18206.32 ms and Avg. 41.65 megabytes for SSBN_g1_t4.

Table G.3 Performance Averages for Dynamic Situation

<table>
<thead>
<tr>
<th>SSBN</th>
<th>Time</th>
<th>Memory Usage</th>
<th>Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSBN_g1_t2</td>
<td>7594.47 (4171360.35)</td>
<td>25.33 (1.66)</td>
<td>9.63 (8.96)</td>
</tr>
<tr>
<td>SSBN_g1_t3</td>
<td>12275.82 (12511332.49)</td>
<td>33.80 (2.65)</td>
<td>9.85 (10.86)</td>
</tr>
<tr>
<td>SSBN_g1_t4</td>
<td>18206.32 (27638930.89)</td>
<td>41.65 (3.10)</td>
<td>9.89 (11.19)</td>
</tr>
</tbody>
</table>

Results for the time and the memory usage show a similar pattern. Thus, the increases in both were linear over t2, t3, and t4. The iteration numbers over time increased slightly and all inferences converged in around 9 iterations.

**Query 1.1: What is the type of the target?**

For this, an RV for the type of a target g1 (i.e., TargetType_g1) was queried. The RV TargetType_g1 was discrete and contained three states Others, ThreateningGroundTarget, and ThreateningAirTarget. Table G.4 shows the accuracy results of the RV TargetType_g1 for the SSBNs SSBN_g1_t2, SSBN_g1_t3, and SSBN_g1_t4. For example, for the SSBN SSBN_g1_t2, the average Brier score had the average 0.3692 and the standard deviation 0.193. Also, the sensitivity, specificity, and AUC were 0.85, 0.735, and 0.6853, respectively.

As we can see Table G.4, the accuracy scores for the SSBNs were almost same, although the Brier score increased slightly over time. The RV TargetType_g1 is a static RV which doesn’t contain a time entity and doesn’t change over time. The velocities for
the target’s location (i.e., Latitude, Longitude, and Altitude) were used to identify the type of the target; however, their changes over time were very small and didn’t much influence the probability changes for the RV TargetType_g1.

Table G.4 Accuracies for the Target Type Query

<table>
<thead>
<tr>
<th>SSBN</th>
<th>Target Type</th>
<th>Brier score</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSBN_g1_t2</td>
<td>TargetType</td>
<td>0.36923</td>
<td>0.85</td>
<td>0.735</td>
<td>0.6853</td>
</tr>
<tr>
<td>SSBN_g1_t3</td>
<td>TargetType</td>
<td>0.36923</td>
<td>0.85</td>
<td>0.735</td>
<td>0.6853</td>
</tr>
<tr>
<td>SSBN_g1_t4</td>
<td>TargetType</td>
<td>0.36923</td>
<td>0.85</td>
<td>0.735</td>
<td>0.6853</td>
</tr>
</tbody>
</table>

Figure G.6 Average Brier Scores for the Target Type Query

Query 1.2: Where will be the target located?

For this, nine RVs for a target g1 (i.e., Latitude_g1_t2~t4, Longitude_g1_t2~t4, and Altitude_g1_t2~t4) were queried. Each RV was conditional Gaussian given a velocity RV and a previous location RV. Table G.5 shows the accuracy results (i.e., average of continuous ranked probability score (CRPS)) of these RVs for the SSBNs SSBN_g1_t2,
SSBN_g1_t3, and SSBN_g1_t4. For example, for the SSBN SSBN_g1_t2, the averages of CRPS for Latitude_g1_t2, Longitude_g1_t2, and Altitude_g1_t2 were 0.01004, 0.01524, and 0.00198, respectively.

<table>
<thead>
<tr>
<th>SSBN</th>
<th>Latitude</th>
<th>Longitude</th>
<th>Altitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSBN_g1_t2</td>
<td>0.01004 (0.00003)</td>
<td>0.01524 (0.00005)</td>
<td>0.00198 (0.000001)</td>
</tr>
<tr>
<td>SSBN_g1_t3</td>
<td>0.01829 (0.00012)</td>
<td>0.02656 (0.00015)</td>
<td>0.0025 (0.000002)</td>
</tr>
<tr>
<td>SSBN_g1_t4</td>
<td>0.02666 (0.00029)</td>
<td>0.03771 (0.0003)</td>
<td>0.0031 (0.0003)</td>
</tr>
</tbody>
</table>

We observed that the accuracies for the RVs Latitude, Longitude, and Altitude decreased, because of process errors in the velocity RVs influencing the RVs Latitude, Longitude, and Altitude. Fig. G.7 shows the increasing aspect of the average CRPSs.
Query 1.3: What type activity will the target perform?

For this, three RVs for the activity of a target g1 over time t2, t3, and t4 (i.e., Activity_g1_t2, Activity_g1_t3, and Activity_g1_t4) were queried. These RVs were discrete and contained three states HeadingToOthers, HeadingToCI, and Staying. The RV activity depends on the three RVs (TargetType, Mission, and PreActivity) and influences the six RVs (LatitudeVelocity, LongitudeVelocity, AltitudeVelocity, DistanceToCI, DirectionToCI, and Temperature). Table G.6 shows the accuracy results of these RVs for the SSBNs SSBN_g1_t2, SSBN_g1_t3, and SSBN_g1_t4. For example, for the SSBN SSBN_g1_t2, the average Brier score had the average 0.5625 and the standard deviation 0.5857. Also, the sensitivity, specificity, and AUC were 0.92, 0.96, and 0.8921, respectively.

<table>
<thead>
<tr>
<th>SSBN</th>
<th>Activity</th>
<th>Brier score</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSBN_g1_t2</td>
<td></td>
<td>0.56254 (0.58574)</td>
<td>0.92</td>
<td>0.96</td>
<td>0.892</td>
</tr>
<tr>
<td>SSBN_g1_t3</td>
<td></td>
<td>0.60141 (0.57543)</td>
<td>0.91</td>
<td>0.945</td>
<td>0.86894</td>
</tr>
<tr>
<td>SSBN_g1_t4</td>
<td></td>
<td>0.64179 (0.56476)</td>
<td>0.9</td>
<td>0.9</td>
<td>0.81419</td>
</tr>
</tbody>
</table>

As we can see Table G.6, the accuracy scores for the queries decreased over time.
Fig. G.8 shows the increasing aspect of the average Brier scores.

**Query 1.4: What mission does the target have?**

For this, three RVs for the mission of a target $g1$ over time $t2$, $t3$, and $t4$ (i.e., $Mission_{g1\_t2}$, $Mission_{g1\_t3}$, and $Mission_{g1\_t4}$) were queried. These RVs were discrete and contained two states *Attack* and *Others*. The RV *Mission* depends on the two RVs (*TargetType* and *Pre-Mission*) and influences the RV *Activity*. Table G.7 shows the accuracy results of these RVs for the SSBNs $SSBN_{g1\_t2}$, $SSBN_{g1\_t3}$, and $SSBN_{g1\_t4}$. For example, for the SSBN $SSBN_{g1\_t2}$, the average Brier score had the average 0.39 and the standard deviation 0.22. Also, the sensitivity, specificity, and AUC were 1, 0.97, and 0.91, respectively.
### Table G.7 Accuracies for the Target Mission Query

<table>
<thead>
<tr>
<th>Mission</th>
<th>Brier score</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSBN g1 t2</td>
<td>0.39003 (0.22122)</td>
<td>1</td>
<td>0.97</td>
<td>0.9132</td>
</tr>
<tr>
<td>SSBN g1 t3</td>
<td>0.39006 (0.22124)</td>
<td>1</td>
<td>0.97</td>
<td>0.9132</td>
</tr>
<tr>
<td>SSBN g1 t4</td>
<td>0.39008 (0.22126)</td>
<td>1</td>
<td>0.97</td>
<td>0.9132</td>
</tr>
</tbody>
</table>

As we can see from Table G.7, the accuracy scores for the SSBNs were almost the same, although the Brier score increased slightly over time.

![Figure G.9 Average Brier Scores for the Target Mission Query](image)

Fig. G.9 shows the very slightly increasing aspect of the average Brier scores.

#### G.4.2 Situation for Multiple Targets

In Appendix G.2, we saw the CLD for the RV *DangerLevel*. This case deals with the aggregating influence situation in which many parent nodes influence a child node. The RV *DangerLevel* belongs to the case. The RV *DangerLevel* depends on the pair RVs (*TargetType* and *Mission*). If there are many target entities, many pair RVs for them (e.g.,
can be constructed. However, the aggregating influence situation can be problematic, when there are many discrete parent nodes. For example, The RVs TargetType and Mission are discrete. The number of states for the RV TargetType and the number of states for the RV Mission are 3 and 2, respectively. If there is only one target entity, two RVs (i.e., TargetType and Mission) are constructed and the number of IPCs for the child RV DangerLevel will be $3 \times 2 = 6$. The number of IPCs increases exponentially. For example, if there are five target entities, the number of IPCs for the child RV DangerLevel will be $6^5 = 7776$. Therefore, we can't use this approach and we introduce a decomposable approach.

This approach decomposes many parent nodes for a child node into a set of pairs for a parent node and a dummy child node [Heckerman & Breese, 1996][Jurgenelait & Lucas, 2005]. For example, in our case, we use a dummy child node for a parent node. The dummy child node deals with the IPC from its parent node, so this can prevent the exponential growth of the number of IPCs. The dummy child node, then, influences a next dummy child node depending on a next parent node. This connection keeps being made until a last dummy child node for the last parent node. The last dummy child node may or may not influence a resulting child node which is used to calculate the last result from the result from the last dummy child node. The following shows a set of instance local distributions (ILDs) for a case of two target entities. For the two target entities, two dummy child nodes ($X1$ and $X2$) are constructed and the last result from these nodes is calculated in the resulting child node (DangerLevel_ci1).
As we can see the above script, the resulting child node (*DangerLevel_ci1*) depends on the last dummy child node *X2* which produces a summation of the dummy
child nodes and it is used in the exponential function in the resulting child node (i.e., $\text{Exp}(X_2/10, e)$).

Query 2.1 (How high is the level of danger to the critical infrastructure?) is to recognize the emergency situation for the critical infrastructure. However, we didn’t have actual or simulated datasets for emergency situation reflecting reality. Their accuracy for the inference was not evaluated. We only conducted an experiment in which operational performances (e.g., memory usage, execution time, the limitation for a complex SSBN) of the HMP-GMR algorithm for SSBNs generated from the HARALD MTheory were tested. For this query, five SSBNs ($SSBN_{1g}$, $SSBN_{2g}$, $SSBN_{3g}$, $SSBN_{4g}$, and $SSBN_{5g}$) were used. Table G.8 shows the inference result for Query 2.1 over the number of the targets. The table contains the three performance factors (Time, Memory usage, and Iteration). Time is the execution time of the inference algorithm. Iteration is the iteration number of the HMP-GMR algorithm (numbers in parentheses are standard deviations).

We observed that the execution times and the memory usages increased linearly, while the DMP iterations for all cases had the mean of 9.57 with the standard deviation of 7.85. This means that the models we used were stable and feasible.

<table>
<thead>
<tr>
<th>SSBN</th>
<th>Time</th>
<th>Memory Usage</th>
<th>DMP Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SSBN_{1g}$</td>
<td>7607 (3776183.23)</td>
<td>24 (0.00)</td>
<td>9.57 (7.95)</td>
</tr>
<tr>
<td>$SSBN_{2g}$</td>
<td>21736.88 (35667683.66)</td>
<td>42.53 (0.25)</td>
<td>9.57 (7.95)</td>
</tr>
<tr>
<td>$SSBN_{3g}$</td>
<td>41100.62 (109539859.00)</td>
<td>60.0 (0.06)</td>
<td>9.57 (7.95)</td>
</tr>
<tr>
<td>$SSBN_{4g}$</td>
<td>68580.52 (313273212.31)</td>
<td>78.03 (0.19)</td>
<td>9.57 (7.95)</td>
</tr>
<tr>
<td>$SSBN_{5g}$</td>
<td>101198.45 (685775084.21)</td>
<td>96.72 (0.37)</td>
<td>9.57 (7.95)</td>
</tr>
</tbody>
</table>
In conclusion for all queries, the results of evaluation for the target type, the target location, the target activity, and the target mission are summarized in Table G.9.

<table>
<thead>
<tr>
<th>Target Type</th>
<th>Estimation and Prediction</th>
<th>Threshold</th>
<th>Performance requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSBN_g1_t2</td>
<td>0.36923</td>
<td>0.5</td>
<td>Satisfied</td>
</tr>
<tr>
<td>SSBN_g1_t3</td>
<td>0.36923</td>
<td>0.5</td>
<td>Satisfied</td>
</tr>
<tr>
<td>SSBN_g1_t4</td>
<td>0.36923</td>
<td>0.5</td>
<td>Satisfied</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Target Location</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSBN_g1_t2</td>
<td>Latitude</td>
<td>0.01004</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>Longitude</td>
<td>0.01524</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>Altitude</td>
<td>0.00198</td>
<td>0.1</td>
</tr>
<tr>
<td>SSBN_g1_t3</td>
<td>Latitude</td>
<td>0.01829</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>Longitude</td>
<td>0.02656</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>Altitude</td>
<td>0.00250</td>
<td>0.1</td>
</tr>
<tr>
<td>SSBN_g1_t4</td>
<td>Latitude</td>
<td>0.02666</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>Longitude</td>
<td>0.03771</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>Altitude</td>
<td>0.00310</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Target Activity</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSBN_g1_t2</td>
<td></td>
<td>0.56254</td>
<td>0.5</td>
</tr>
<tr>
<td>SSBN_g1_t3</td>
<td></td>
<td>0.60141</td>
<td>0.5</td>
</tr>
<tr>
<td>SSBN_g1_t4</td>
<td></td>
<td>0.64179</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Target Mission</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSBN_g1_t2</td>
<td></td>
<td>0.39003</td>
<td>0.5</td>
</tr>
<tr>
<td>SSBN_g1_t3</td>
<td></td>
<td>0.39006</td>
<td>0.5</td>
</tr>
<tr>
<td>SSBN_g1_t4</td>
<td></td>
<td>0.39008</td>
<td>0.5</td>
</tr>
</tbody>
</table>

The results from the target type, the target location, and the target mission addressed the performance requirements. For example, the Brier scores of the target type queries for all SSBNs were 0.37 which addressed the performance requirement for the query (0.5). However, the results from the target activity did not address the performance requirements. To resolve this unsatisfactory result, we can return to the previous steps to improve the performance. We can investigate the learned HERALD MTheory in the Construct Reasoning Model step. Unsatisfactory performance can be caused by a training database of insufficient size. In this case, we may find more datasets and apply them to the learning process. Also, it is possible that the MEBN learning algorithm which we use
is ineffective. In this case, the application of a more effective MEBN learning algorithm is required. The world model in the *Construct Reasoning Model* step can be incorrect. For this, we may need to conduct a further field investigation and research to develop a more accurate world model. The requirements in the *Analyze Requirements* step can be impracticable or requires a too high standard to address it. In this case, readjustments for the requirements can be performed by the stakeholders.

For this proof of concept system, we used the test dataset generated from the simulation to evaluate the learned HERALD MTheory. Obviously, the HERALD MTheory which was learned from the training dataset can be over-fitted. When we use the HERALD MTheory in reality, the learned HERALD MTheory should be learned again using a real dataset from sensors and expert's knowledge on the ground. This is crucial to develop a more sound HERALD MTheory. Because of this, we assumed that we would follow the *quick adaptation paradigm*. 
Appendix H. Use Case 1: HERALD Scenario Simulator

HERALD simulates ground truth information of a situation in which blue and red teams operate against each other. The red team can be a group of terrorists or UAVs (unmanned aerial vehicles). The red team aims to destroy a critical infrastructure element defended by the blue team. The blue team has several types of sensor systems that can detect threatening targets of interest, and a command center, charged with using data from the sensors to identify the situation and make decisions about future actions.

The simulation module simulates a critical infrastructure defense problem with two attack scenarios (terrorist and UAV attacks) in Section 6.1.1. The simulation considers environmental factors (e.g., geography of the target area, weather, temperature, humidity, and wind speed) and spatial objects (e.g., buildings, roads, and critical infrastructures). The simulation contains various simulation entities (e.g., terrorists, UAVs, security guards, omnidirectional radars, directional radars, ships, airplanes, and animals). Operational rules of the simulation entities are derived from historical information, sensor theories, and domain expert knowledge, and they are applied in the simulation.

H.1 Environmental Factors, Spatial objects, Way Points, and Time

The simulation considers environmental factors (e.g., geography, weather, temperature, humidity, and wind speed), spatial objects (e.g., water, forest, land, parking lot, building, fence, farm, road, and key facility), and waypoints (e.g., waypoints of a sea/ground/air route).
The simulation contains a set of layers for geospatial information. Each layer can be used for a special purpose. For example, a layer can represent a map of the situation, regions indicating countries, and regions depicting religious populations. Each layer contains an ordered set of maps indicating x and y coordinates. The order of the maps can be used to represent z coordinate to support a three-dimension. Each map consists of a set of tiles in the x and y coordinates. Each tile can have information about the environmental factors, spatial objects, and/or waypoints. The simulation entities interact each other using their own operational rules. Mobile simulation entities (e.g., bird, car, airplane, and ship) have their own waypoints for movement. The time interval for the simulation is one second and can be represented as $t_1$, $t_2$, and $t_3$ for 0 ~ 1 second, 1 ~ 2 second, and 2 ~ 3 second, respectively.
H.2 Simulation Entities

The simulation contains various simulation entities (e.g., terrorists, UAVs, security guards, omnidirectional radars, directional radars, long-range cameras, tension fences, fiber optic fences, UWB wireless motion sensors, access controls, CCTV cameras, ships, airplanes, and animals). These simulation entities possess immutable properties (e.g., entity type and entity subtype) and mutable properties (e.g., mission, state, sound, temperature, appearance, departure, destination, speed, latitude, longitude, and altitude). The simulation entities can be classified as the blue/red team and neutral entities. The neutral entities (i.e., animals, cars, ships, and airplanes) impede situational awareness in the simulation, while the blue team entities (i.e., various sensor systems) aim to be aware of aspects of the red team entities (i.e., terrorists and UAVs).

H.2.1 Operational Rules for Simulation Entities

The simulation entities interact with environmental factors, spatial objects, and/or waypoints according to operational rules. Each of the simulation entities possesses a set of operational rules describing how the entity works given some conditions. For example, if an entity is a fishing ship, it can contain the following operational rules: (1) at certain time, it sails from a small port to a fishing area, (2) at the fishing area, it starts fishing for a few hours, (3) after fishing, it sails to another fishing area and repeats this operation, (4) after some hours of operation, it returns to the port it departed.

An operational rule consists of a set of condition statements and a set of action statements. A condition statement contains the type of a condition (e.g., Location and Time) and its parameters (e.g., FishingArea1, 6:00 AM). An action statement contains the type of an action (e.g., Move and Fish), its parameters (e.g., FishingArea2 and two
hours), and the probability the action event occurs (e.g., $P(\text{Move}) = 0.8$ and $P(\text{Fish}) = 0.2$). For example, the following statement shows an operational rule for a fishing ship.

If $[\text{Ship1.Location} = \text{FishingArea11}]$ and $[\text{Global.Time} = 6:00]$, 
Then $[\text{Move, FishingArea11, 0.8}]$ or $[\text{Fish, two hours, 0.2}]$

This means if the location of $\text{Ship1}$ is $\text{FishingArea11}$ and the global time of the simulation is 6:00, then the fishing ship moves to $\text{FishingArea11}$ with the probability 0.8 or the fishing ship fishes for two hours with the probability 0.2.

The simulation entities also contain operational rules for sensors (e.g., security guards, omnidirectional radars, directional radars, fiber optic fences, access controls, and electro-optical cameras). A security guard is a kind of mobile sensor which can move around some scheduled waypoints, detect targets, and report unusual events from the targets. Also, there are stationary sensors such as omnidirectional radars which were installed on a building. The stationary sensors were placed in and around the critical infrastructure. The field of view for a sensor varies according to the type of the sensor. Each sensor features an angle of field (e.g., the angle of field for an omnidirectional radar is 360 degrees) and working distance (e.g., the working distance for an omnidirectional radar is 1 km). Not all areas in the HERALD scenario simulator are covered by these sensors and some areas are covered by multiple sensors. Depending on conditions of the sensors (e.g., distance between the target and sensor) and conditions of the environment (e.g., weather, air pollution, humidity, and day/night), the observed results by the sensors can be varied. These conditions are programmed in the HERALD scenario simulator.
using a set of conditional probability distributions. As an illustrative example, the following statement shows one case.

\[ P(\text{Image type} = \text{AMO} \mid \text{Sensor Performance} = \text{high}, \text{Distance} = \text{long}, \text{Reported Image} = \text{AMO}) = 0.85 \]

This means that given three conditions (the sensor performance is high, the distance is long, and the reported image is an air-moving object (AMO)), the probability of the AMO image type for a target is 0.85.

In the simulation, the world and entity information are randomly initialized. Once the simulation starts, the number of entities and some of the environmental factors (e.g., weather, temperature, humidity, and wind speed) are randomly initialized as well as the initial values of the attribute, starting time of operational rules, and the type of operational rules of an entity are randomly initialized. However, the geography of the world, spatial objects (e.g., water, forest, land, parking lot, common building, fence, farm, road, and key facility), and waypoints (e.g., waypoints of a sea route and waypoints of an air route) are initialized from ready-made models.

**H.3 Sensor Systems in the Simulation**

The sensor systems convey reports to the HERALD system that enable awareness of aspects of targets of interest (e.g., terrorists and UAVs). The sensors are of various types such as an MTI (Moving Target Indicator) system, an IMINT (Imagery Intelligence) system, and a GEOINT (Geospatial intelligence) system.
The MTI system contains an omnidirectional radar and directional radars. The radar sensor systems can estimate the range, location, and region [Curry, 2007][Dunn III et al., 2004]. For example, the omnidirectional radar can detect where a target is located. The location of the target can be described by the specific values for latitude, longitude, and altitude. The specific values for latitude, longitude, and altitude can provide a region identification (ID) indicating the region where the target stands. Thus, the location of the target is described by the values of latitude, longitude, and altitude, while the region of the target observed by the sensor is described by the region ID.

The IMINT sensor system contains infrared cameras and long-range electro-optical cameras. The electro-optical camera sensor system can estimate the range, size, location, image type, and region [Dudzik, 1993][Geetha & Narayanan, 2008][Brezeale & Cook, 2008]. The infrared camera sensor system can estimate the range, size, location, temperature, image type, and region [Holst, 2000][Abarbanel et al., 2000][Vollmerhausen & Jacobs, 2004]. For example, an infrared camera can detect a target and provide imagery and/or full motion video data. From these data, an appearance type ID (e.g., air-moving object (AMO), land-moving object (LMO), or sea-moving object (SMO)) of the target can be estimated. The infrared camera also can be used to detect an average temperature of a target using a thermal image processing method.

The GEOINT sensor system provides information about regions, in which natural object information (e.g., plants, weather, and region types), artificial object information (e.g., roads, buildings, and facilities), and social information (e.g., population, religious propensity, and ethnos) can be reported.
In the simulation, each sensor system provides its own intelligence. The MTI system provides a target location report, a target-distance-to-CI report, and a target-direction-to-CI report. The location report can be described by the specific values for latitude, longitude, and altitude (e.g., N 38°51'10.0", W 77°18'44.3", and 64.05 meters). For simplicity, a variable X for latitude and a variable Y for longitude can be used to represent the location of a target. The target-distance-to-CI report means a distance between a target and CI. The target-direction-to-CI report means an angle of the relative bearing of a target to CI. For example, if a target-direction-to-CI report is close to zero degrees, the target is heading to CI. The IMINT sensor system provides a target image report, a target temperature report, and a target size report. The target image report gives the apparent type of the target [Self et al., 2005]. The target temperature is the target's area weighted average temperature [Dudzik, 1993]. The target size report is taken as the square root of the viewed target area and is averaged over all aspect angles.

<table>
<thead>
<tr>
<th>MTI system</th>
<th>Reports</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Target location</td>
<td>Latitude, longitude, and altitude</td>
</tr>
<tr>
<td></td>
<td>Target-distance-to-CI</td>
<td>A real number (meter)</td>
</tr>
<tr>
<td></td>
<td>Target-direction-to-CI</td>
<td>Angle</td>
</tr>
<tr>
<td>IMINT sensor system</td>
<td>Target image</td>
<td>E.g., military tracked vehicle and commercial wheeled vehicle</td>
</tr>
<tr>
<td></td>
<td>Target temperature</td>
<td>A real number (Fahrenheit)</td>
</tr>
<tr>
<td></td>
<td>Target size</td>
<td>A real number (meter)</td>
</tr>
<tr>
<td>GEOINT system</td>
<td>Target region type</td>
<td>E.g., road and off-road</td>
</tr>
</tbody>
</table>

Table H.1 Sensor Systems and Reports
[Vollmerhausen & Jacobs, 2004]. The GEOINT system provides a target region type report. The target region type report means a type of the region where the target stands.

**H.4 Ground Truth Dataset**

The simulation generates a training dataset to learn a HERALD MTheory and a test dataset to evaluate the learned HERALD MTheory. These datasets are stored in a relational database.

For these datasets which the simulation provides, we can think of two situations: one is that we can have an actual entity dataset and another is that we can have only a sensor dataset. The actual entity dataset is a ground truth dataset which is regarded as true information for entities. For example, there is an oracle who knows everything in the situation. The sensor dataset is a dataset which is generated by sensors in the situation. If we think of the real world (not an artificial world) situation, a dataset which we can easily have is the sensor dataset. Using the sensor dataset, a sensor model, and its performance can be learned. However, it is very difficult to have the actual entity dataset (We are not the oracle). However, in some cases, we can obtain an actual entity dataset. For example, after a critical event happens (e.g., a terror attack), a dataset reflecting the critical event can be obtained and it can be modeled. Probably, a counter-terrorism expert will analyze the course of actions, missions, and activities of terrorist after the critical event occurred. The analysis result will be recorded in a report and incorporated into the actual entity dataset. Another case is that for a critical infrastructure protection a virtual terrorist attacking operation can be planned and performed by a government agency for homeland security. The result from the operation can be the actual entity dataset for learning the
HERALD MTheory. In the HERALD simulation, we assume these cases in which the actual entity dataset contains the historical-critical event data and the virtual operation data. Fig. H.2 shows a HERALD database schema represented by an Enhanced Entity–Relationship (EER) model.
The schema contains 15 relations (i.e., `sensor`, `target`, `time`, `ci_situation`, `reportettarpt`, `sensordof`, `reportettarpt_mintsrpt`, `reportettarpt_geointrpt`, `sensortemporalproperty`, `targettemporalproperty`, `predecessor`, `criticalinfrastructure`, `mti_report`, `imints_report`, and `geints_report`). For example, the `target` relation possesses a primary key (`TargetID`) and three attributes (`TargetType`, `TargetSize`, and `TargetImage`) and indicates a target information observed by the sensor systems.
Appendix I. Use Case 1: HERALD MTheory

HERALD MTheory (MTheory I.1) contains the following 12 MFrags (SensorOf, Predecessor, SensorTemporalProperty, ReportedTarget_MTIRPT, ReportedTarget_IMINTSRPT, ReportedTarget_GEOINTRPT, GEOINTS_Report, MTI_Report, IMINTS_Report, Target, TargetTemporalProperty, and Situation).

MTheory I.1: HERALD MTheory

1 [F: SensorTemporalProperty
2     [C: Isa(tr, TARGET), Isa(sr, SENSOR), Isa(t, TIME)]
3     [R: DistanceToSensor(tr, sr, t)]
4 ]
5 [F: Target
6     [C: Isa(tr, TARGET)]
7     [R: TargetType(tr)]
8     [R: TargetSize(tr)]
9     [RP: TargetType(tr)]
10 ]
11 [R: TargetImage(tr)
12     [RP: TargetType(tr)]
13 ]
14 ]
15 [F: TargetTemporalProperty
16     [C: Isa(tr, TARGET), Isa(t, TIME), Isa(pret, TIME)]
17     [C: Predecessor(pret, t)]
18     [R: Latitude_Velocity(tr, t)
19         [RP: Activity(tr, t), RegionType(tr, t)][IP: TargetType(tr), Latitude_Velocity(tr, pret)]
20     ]
21     [R: Longitude_Velocity(tr, t)
22         [RP: Activity(tr, t), RegionType(tr, t)][IP: TargetType(tr), Longitude_Velocity(tr, pret)]
23     ]
24     [R: Altitude_Velocity(tr, t)
25         [RP: Activity(tr, t), RegionType(tr, t)][IP: TargetType(tr), Altitude_Velocity(tr, pret)]
26     ]
27     [R: Latitude(tr, t)
28         [IP: Latitude(tr, pret), Latitude_Velocity(tr, pret)]
29     ]
30     [R: Longitude(tr, t)
31         [IP: Longitude(tr, pret), Longitude_Velocity(tr, pret)]
32     ]
33     [R: Altitude(tr, t)
34         [IP: Altitude(tr, pret), Altitude_Velocity(tr, pret)]
35     ]
36     [R: DistanceToCl(tr, t)
37         [RP: Activity(tr, t)] [IP: DistanceToCl(tr, pret)]
38     ]
39     [R: DirectionToCl(tr, t)
40         [RP: Activity(tr, t)] [IP: DirectionToCl(tr, pret)]
41 ]
R: Temperature\((tr, t)\) 
[RP: Activity\((tr, t)\)] [IP: TargetType\((tr)\), Temperature\((tr, pret)\)]
The following sub-sections describe the graphic form of each M_frag in HERALD MTheory.

### I.1 SensorOf M_frag

![SensorOf M_frag diagram](image)

**Figure I.1 SensorOf**
I.2 Predecessor M_frag

![Figure I.2 Predecessor](image)

I.3 SensorTemporalProperty M_frag

![Figure I.3 SensorTemporalProperty](image)

I.4 ReportedTarget_MTIRPT M_frag

![Figure I.4 ReportedTarget_MTIRPT](image)
1.5 ReportedTarget_IMINTSRPT MFragment

Figure I.5 ReportedTarget_IMINTSRPT

1.6 ReportedTarget_GEOINTRPT MFragment

Figure I.6 ReportedTarget_GEOINTRPT

1.7 GEOINTS_Report MFragment

Figure I.7 GEOINTS_Report
I.8 MTI_Report MFragment

![Figure I.8 MTI_Report MFragment](image1.png)

I.9 IMINTS_Report MFragment

![Figure I.9 IMINTS_Report MFragment](image2.png)
I.10 Target MFragment

![Figure I.10 Target](image)

I.11 TargetTemporalProperty MFragment

![Figure I.11 TargetTemporalProperty](image)
I.12 Situation MFragment

Figure I.12 Situation
Appendix J. Use Case 1: Confusion Matrix for Categorical Variables of HERALD MTheory

The followings show the experimental results for the categorical variables on the HERALD MTheory in Section 6.1. For the three categorical variables *TargetType*, *Mission*, and *Activity* in the three SSBNs (i.e., SSBN_g1_t2, SSBN_g1_t3, SSBN_g1_t4) generated from the learned HERALD MTheory, each confusion matrix is shown in the followings.

**J.1 SSBN_g1_t2**

<table>
<thead>
<tr>
<th>Table J.1 TargetType_g1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Condition</td>
</tr>
<tr>
<td>Positive</td>
</tr>
<tr>
<td>Condition Positive</td>
</tr>
<tr>
<td>Condition Negative</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table J.2 Mission_g1_t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Condition</td>
</tr>
<tr>
<td>Positive</td>
</tr>
<tr>
<td>Condition Positive</td>
</tr>
<tr>
<td>Condition Negative</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table J.3 Activity_g1_t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Condition</td>
</tr>
<tr>
<td>Positive</td>
</tr>
<tr>
<td>Condition Positive</td>
</tr>
<tr>
<td>Condition Negative</td>
</tr>
</tbody>
</table>
### J.2 SSBN_g1_t3

**Table J.4 TargetType_g1**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Predicted Condition Positive</th>
<th>Predicted Condition Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition Positive</td>
<td>86</td>
<td>14</td>
</tr>
<tr>
<td>Condition Negative</td>
<td>59</td>
<td>141</td>
</tr>
</tbody>
</table>

**Table J.5 Mission_g1_t3**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Predicted Condition Positive</th>
<th>Predicted Condition Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition Positive</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>Condition Negative</td>
<td>3</td>
<td>97</td>
</tr>
</tbody>
</table>

**Table J.6 Activity_g1_t3**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Predicted Condition Positive</th>
<th>Predicted Condition Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition Positive</td>
<td>90</td>
<td>10</td>
</tr>
<tr>
<td>Condition Negative</td>
<td>12</td>
<td>188</td>
</tr>
</tbody>
</table>

### J.3 SSBN_g1_t4

**Table J.7 TargetType_g1**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Predicted Condition Positive</th>
<th>Predicted Condition Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition Positive</td>
<td>89</td>
<td>11</td>
</tr>
<tr>
<td>Condition Negative</td>
<td>50</td>
<td>150</td>
</tr>
</tbody>
</table>
### Table J.8 Mission_g1_t4

<table>
<thead>
<tr>
<th>Condition</th>
<th>Predicted Condition Positive</th>
<th>Predicted Condition Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition Positive</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>Condition Negative</td>
<td>2</td>
<td>98</td>
</tr>
</tbody>
</table>

### Table J.9 Activity_g1_t4

<table>
<thead>
<tr>
<th>Condition</th>
<th>Predicted Condition Positive</th>
<th>Predicted Condition Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition Positive</td>
<td>92</td>
<td>8</td>
</tr>
<tr>
<td>Condition Negative</td>
<td>17</td>
<td>183</td>
</tr>
</tbody>
</table>
Appendix K. Use Case 1: SSBN_g1_t4

defineNode(DistanceToSensor_g1_s1_t1 , Desc);
{ defineState(Discrete, Mid, Long, Short );
p( DistanceToSensor_g1_s1_t1 ) =
{ Mid : .5780996240183319; Long : .0972435426817059;
Short : .3246568332999622; } }
defineNode(TargetType_g1 , Desc);
{ defineState(Discrete, Others, ThreateningGroundTarget, ThreateningAirTarget );
p( TargetType_g1 ) =
{ Others : .8761061946902655; ThreateningGroundTarget : .0619469026548673;
ThreateningAirTarget : .0619469026548673; } }
defineNode(TargetSize_g1 , Desc);
{ defineState(Continuous);
p( TargetSize_g1 | TargetType_g1 ) =
if( TargetType_g1 == ThreateningGroundTarget ) {NormalDist( 2.064385802853249, .0034625044260608); }
else if( TargetType_g1 == Others ) {NormalDist( 3.5288461938253017, 36.0779522048415); }
else if( TargetType_g1 == ThreateningAirTarget ) {NormalDist( 2.042528603915358, .0082069776092345); }
}
defineNode(TargetImage_g1 , Desc);
{ defineState(Discrete, CommercialFWA, Others, HumanWithWeapon, MilitaryFWA, CommercialWheeled );
p( TargetImage_g1 | TargetType_g1 ) =
if( TargetType_g1 == ThreateningGroundTarget ) {CommercialFWA : .0909090909090909; Others : .0909090909090909;
HumanWithWeapon : .6363636363636364; MilitaryFWA : .0909090909090909; CommercialWheeled : .0909090909090909; }
else if( TargetType_g1 == Others ) {CommercialFWA : .0679611650485437;
Others : .786407766902912; HumanWithWeapon : .0097087378640777; MilitaryFWA : .0097087378640777; CommercialWheeled : .1262135922330097; }
else if( TargetType_g1 == ThreateningAirTarget ) {CommercialFWA : .0909090909090909; Others : .0909090909090909;
HumanWithWeapon : .0909090909090909; MilitaryFWA : .6363636363636364; CommercialWheeled : .0909090909090909; }
}
defineNode(Mission_g1_t1 , Desc);
{ defineState(Discrete, Others, Attack );
p( Mission_g1_t1 | TargetType_g1 ) =
if( TargetType_g1 == ThreateningGroundTarget ) {Others : .0002977963073258;
Attack : .9997022036926743; }
else if( TargetType_g1 == Others ) {Others : .999965029667449;
Attack : .0000344970332551; }
else if( TargetType_g1 == ThreateningAirTarget ) {Others : .0008896797153025;
Attack : .999113020846975; }
}
defineNode(Activity_g1_t1 , Desc);
{ defineState(Discrete, HeadingToCI, HeadingToOthers, Staying );
p( Activity_g1_t1 | TargetType_g1 , Mission_g1_t1 ) =
if( Mission_g1_t1 == Attack && TargetType_g1 == ThreateningGroundTarget ) {HeadingToCI : .6957427805894612; HeadingToOthers : .258125334921108;
Staying : .0461446859184281; }
else if( Mission_g1_t1 == Others & TargetType_g1 == ThreateningAirTarget ) {HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333;
Staying : .3333333333333333; } }
else if( Mission_g1_t1 == Others && TargetType_g1 == ThreateningGroundTarget )
    {HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333;  
     Staying : .3333333333333333; }
else if( Mission_g1_t1 == Others && TargetType_g1 == Others )
    {HeadingToCI : .1190451550588154; HeadingToOthers : .7281037634964986;  
     Staying : .1528510814446859;  }
else if( Mission_g1_t1 == Attack && TargetType_g1 == Others )
    {HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333;  
     Staying : .3333333333333333;  }
else if( Mission_g1_t1 == Attack && TargetType_g1 == ThreateningAirTarget )
    {HeadingToCI : .3084444444444445; HeadingToOthers : .6906666666666667;  
     Staying : .0008888888888889;  }
}

defineNode(RegionType_g1_t1 , Desc);
{ defineState(Discrete, OffRoad, road );
  p( RegionType_g1_t1 ) =  
    { OffRoad : .9967429630072312; road : .0032570369927688;  }
}

defineNode(Latitude_Velocity_g1_t1 , Desc);
{ defineState(Continuous);
  p( Latitude_Velocity_g1_t1 | TargetType_g1 , Activity_g1_t1 , RegionType_g1_t1 ) =
    if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToCI && 
        TargetType_g1 == Others ) {NormalDist( .0023046535688136 , .0002151901672446);}
  else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers && 
        TargetType_g1 == ThreateningAirTarget ) {NormalDist( -.0210916384918599, .0012679043286628);}
  else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToOthers && 
        TargetType_g1 == Others ) {NormalDist( .0094773026142154, .0002178506566731);}
  else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToCI && 
        TargetType_g1 == ThreateningAirTarget ) {NormalDist( -.0210916384918599, .0012679043286628);}
  else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == Staying && 
        TargetType_g1 == Others ) {NormalDist( 0.0 , 0.000093750);}
  else if( RegionType_g1_t1 == road && Activity_g1_t1 == Staying && 
        TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.000093750);}
  else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers && 
        TargetType_g1 == ThreateningGroundTarget ) {NormalDist( 0.0 , 0.000093750);}
  else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToOthers && 
        TargetType_g1 == ThreateningGroundTarget ) {NormalDist( 0.0 , 0.000093750);}
  else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == Staying && 
        TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.000093750);}
  else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToCI && 
        TargetType_g1 == ThreateningGroundTarget ) {NormalDist( .0491010044794004, .0019055549264446);}
  else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToCI && 
        TargetType_g1 == ThreateningGroundTarget ) {NormalDist( -.0050458876422215, .0000100790130203);}
  else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers && 
        TargetType_g1 == ThreateningGroundTarget ) {NormalDist( -.0031335033694045, .0000068876630942);}
  else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToOthers && 
        TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.000093750);}
  else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == Staying && 
        TargetType_g1 == ThreateningAirTarget ) {NormalDist( -.0050458876422215, .0000100790130203);}
  else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToCI && 
        TargetType_g1 == ThreateningGroundTarget ) {NormalDist( .0491010044794004, .0019055549264446);}
  else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToCI && 
        TargetType_g1 == ThreateningGroundTarget ) {NormalDist( -.0050458876422215, .0000100790130203);}
  else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers && 
        TargetType_g1 == ThreateningGroundTarget ) {NormalDist( -.0031335033694045, .0000068876630942);}
  else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToOthers && 
        TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.000093750);}
  else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == Staying && 
        TargetType_g1 == ThreateningAirTarget ) {NormalDist( -.0050458876422215, .0000100790130203);}
  else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToCI && 
        TargetType_g1 == ThreateningGroundTarget ) {NormalDist( .0491010044794004, .0019055549264446);}
  else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToCI && 
        TargetType_g1 == ThreateningGroundTarget ) {NormalDist( -.0050458876422215, .0000100790130203);}
  else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers && 
        TargetType_g1 == ThreateningGroundTarget ) {NormalDist( -.0031335033694045, .0000068876630942);}
  else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToOthers && 
        TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.000093750);}
  else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == Staying &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( 0.0 , 0.0000093750);
} else if( RegionType_g1_t1 == road && Activity_g1_t1 == Staying &&
TargetType_g1 == Others ) {NormalDist( 0.0 , 0.0000093750);
}
}
defineNode(Longitude_Velocity_g1_t1 , Desc);
{ defineState(Continuous);
 p( Longitude_Velocity_g1_t1 | TargetType_g1 , Activity_g1_t1 ,
RegionType_g1_t1 ) =
if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToCI &&
TargetType_g1 == Others ) {NormalDist( -.0010418820836344, .0020942460199092);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToCI &&
TargetType_g1 == Others ) {NormalDist( -.0654888456202383 , .02095432494582);
} else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( .0213283889597405, .0188931425484368);
} else if( RegionType_g1_t1 == road && Activity_g1_t1 == Staying &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToCI &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( .0019000520713964, .0019060503729777);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToCI &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( -.0006452952510493, .000136973238477);
} else if( RegionType_g1_t1 == road && Activity_g1_t1 == Staying &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( .0652730541729335, .0014680085717238);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToCI &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( .0010870569942616, .0000009098545841);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( -.0040393646572077, .000006490549193);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( -.001008756942616, .0000009098545841);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == road && Activity_g1_t1 == Staying &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == road && Activity_g1_t1 == Staying &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.001621872);
})
defineNode(Altitude_Velocity_g1_t1 , Desc);
{ defineState(Continuous);
 p( Altitude_Velocity_g1_t1 | TargetType_g1 , Activity_g1_t1 ,
RegionType_g1_t1 ) =
if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToCI &&
TargetType_g1 == Others ) {NormalDist( -.0010418820836344, .0020942460199092);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( -.0654888456202383 , .02095432494582);
} else if( RegionType_g1_t1 == road && Activity_g1_t1 == HeadingToCI &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == Staying &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == Staying &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( -.0010870569942616, .0000009098545841);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( -.0040393646572077, .000006490549193);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( -.001008756942616, .0000009098545841);
} else if( RegionType_g1_t1 == OffRoad && Activity_g1_t1 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == road && Activity_g1_t1 == Staying &&
TargetType_g1 == ThreateningGroundTarget ) {NormalDist( 0.0 , 0.001621872);
} else if( RegionType_g1_t1 == road && Activity_g1_t1 == Staying &&
TargetType_g1 == ThreateningAirTarget ) {NormalDist( 0.0 , 0.001621872);
})
defintenode(Latitude_g1_t1, Desc);
    { definestate(Continuous);
      \( p(\text{Latitude}_g1_t1) = \text{NormalDist}(0.0, 100000000); \)
    }
defineneode(Longitude_g1_t1, Desc);
    { definestate(Continuous);
      \( p(\text{Longitude}_g1_t1) = \text{NormalDist}(0.0, 100000000); \)
    }
defineneode(Altitude_g1_t1, Desc);
    { definestate(Continuous);
      \( p(\text{Altitude}_g1_t1) = \text{NormalDist}(0.0, 100000000); \)
    }
defineneode(DistanceToCI_g1_t1, Desc);
    { definestate(Continuous);
      \( p(\text{DistanceToCI}_g1_t1 | \text{Activity}_g1_t1) = \)
        if( \text{Activity}_g1_t1 == \text{HeadingToCI} ) \{ \text{NormalDist}(1489.2696261407561,
        304527.97024636663); \}
      \text{else if}( \text{Activity}_g1_t1 == \text{Staying} ) \{ \text{NormalDist}(1000.7113066071146,
        161221.2587742367); \}
      \text{else if}( \text{Activity}_g1_t1 == \text{HeadingToOthers} ) \{ \text{NormalDist}(1279.26379454281,
        286823.5807617148); \}
    }
defineNode(DirectionToCI_g1_t1, Desc);
{ defineState(Continuous);
p( DirectionToCI_g1_t1 | Activity_g1_t1 ) =
if( Activity_g1_t1 == HeadingToCI ) {NormalDist( 23.536431924061443,
97.024122477855222 );
} else if( Activity_g1_t1 == Staying ) {NormalDist( 0.0 , 0.0000001 );
} else if( Activity_g1_t1 == HeadingToOthers ) {NormalDist( 93.90814520216249,
1419.2615293644228 );
}
}

defineNode(Temperature_g1_t1, Desc);
{ defineState(Continuous);
p( Temperature_g1_t1 | Activity_g1_t1 ) =
if( Activity_g1_t1 == HeadingToCI ) {NormalDist( 92.09728009556936,
93.38575902146962 );
} else if( Activity_g1_t1 == Staying ) {NormalDist( 95.83531938624884,
21.69607903915628 );
} else if( Activity_g1_t1 == HeadingToOthers ) {NormalDist( 87.28156655197107,
145.50979611125084 );
}
}

defineNode(LatitudeReport_g1_s1_t1, Desc);
{ defineState(Continuous);
p( LatitudeReport_g1_s1_t1 | DistanceToSensor_g1_s1_t1, Latitude_g1_t1 ) =
if( DistanceToSensor_g1_s1_t1 == Short ) { 1.0 * Latitude_g1_t1 +
NormalDist( 0.0 , 0.0000015832463942 );
} else if( DistanceToSensor_g1_s1_t1 == Mid ) { 1.0 * Latitude_g1_t1 +
NormalDist( 0.0 , 0.0000027699623327 );
} else if( DistanceToSensor_g1_s1_t1 == Long ) { 1.0 * Latitude_g1_t1 +
NormalDist( 0.0 , 0.0000053626098128 );
}
}

defineNode(LongitudeReport_g1_s1_t1, Desc);
{ defineState(Continuous);
p( LongitudeReport_g1_s1_t1 | DistanceToSensor_g1_s1_t1, Longitude_g1_t1 ) =
if( DistanceToSensor_g1_s1_t1 == Short ) { 1.0 * Longitude_g1_t1 +
NormalDist( 0.0 , 0.000016091995654 );
} else if( DistanceToSensor_g1_s1_t1 == Mid ) { 1.0 * Longitude_g1_t1 +
NormalDist( 0.0 , 0.000026914371367 );
} else if( DistanceToSensor_g1_s1_t1 == Long ) { 1.0 * Longitude_g1_t1 +
NormalDist( 0.0 , 0.000056339646935 );
}
}

defineNode(AltitudeReport_g1_s1_t1, Desc);
{ defineState(Continuous);
p( AltitudeReport_g1_s1_t1 | DistanceToSensor_g1_s1_t1, Altitude_g1_t1 ) =
if( DistanceToSensor_g1_s1_t1 == Short ) { 1.0 * Altitude_g1_t1 +
NormalDist( 0.0 , 0.000015732463942 );
} else if( DistanceToSensor_g1_s1_t1 == Mid ) { 1.0 * Altitude_g1_t1 +
NormalDist( 0.0 , 0.000026687710333 );
} else if( DistanceToSensor_g1_s1_t1 == Long ) { 1.0 * Altitude_g1_t1 +
NormalDist( 0.0 , 0.000055540737844 );
}
}

defineNode(DistanceToCIReport_g1_s1_t1, Desc);
{ defineState(Continuous);
p( DistanceToCIReport_g1_s1_t1 | DistanceToSensor_g1_s1_t1, DistanceToCI_g1_t1 ) =
if( DistanceToSensor_g1_s1_t1 == Short ) { 1.0 * DistanceToCI_g1_t1 +
NormalDist( 0.0 , 0.000015433910086 );
} else if( DistanceToSensor_g1_s1_t1 == Mid ) { 1.0 * DistanceToCI_g1_t1 +
NormalDist( 0.0 , 0.000027699899884 );
} else if( DistanceToSensor_g1_s1_t1 == Long ) { 1.0 * DistanceToCI_g1_t1 +
NormalDist( 0.0 , 0.000054519677459 );
}
defineNode(DirectionToCIReport_g1_s1_t1, Desc);
{ defineState(Continuous);
p( DirectionToCIReport_g1_s1_t1 | DistanceToSensor_g1_s1_t1 , DirectionToCI_g1_t1 ) =
if( DistanceToSensor_g1_s1_t1 == Short ) { 1.0 * DirectionToCI_g1_t1 +
NormalDist( 0.0 , .000016428204272);
}else if( DistanceToSensor_g1_s1_t1 == Mid ) { 1.0 * DirectionToCI_g1_t1 +
NormalDist( 0.0 , .000027232252329);
}else if( DistanceToSensor_g1_s1_t1 == Long ) { 1.0 * DirectionToCI_g1_t1 +
NormalDist( 0.0 , .000055045471731);
}
}
defineNode(ImageReport_g1_s1_t1, Desc);
{ defineState(Discrete, CommercialFWA, MilitaryWheeled, MilitaryRWA, MilitaryFWA, HumanWithWeapon, MilitaryTracked, CommercialTracked, CommercialWheeled, HumanWithoutWeapon, Others, CommercialRWA);
p( ImageReport_g1_sl_tl | DistanceToSensor_g1_sl_t1 , TargetImage_g1 ) =
if( TargetImage_g1 == Others && DistanceToSensor_g1_sl_t1 == Long )
{CommercialFWA : .0372312532773991; MilitaryWheeled : .0235972732039853;
MilitaryFWA : .0225485055060304; MilitaryFWA : .0214997378080755;
HumanWithWeapon : .0265948085989951; MilitaryTracked : .0256948085989951;
CommercialTracked : .0309386470898669; CommercialWheeled : .0330361824855794;
CommercialWheeled : .02988973917147;  }
else if( TargetImage_g1 == HumanWithWeapon && DistanceToSensor_g1_sl_t1 == Long )
{CommercialFWA : .0909090909090909; MilitaryWheeled : .0909090909090909;
MilitaryRWA : .0909090909090909; MilitaryFWA : .0909090909090909;
HumanWithWeapon : .0909090909090909; MilitaryTracked : .0909090909090909;
CommercialTracked : .0909090909090909; CommercialWheeled : .0909090909090909;
CommercialWheeled : .0909090909090909;  }
else if( TargetImage_g1 == MilitaryFWA && DistanceToSensor_g1_sl_t1 == Long )
{CommercialFWA : .034827586206897; MilitaryWheeled : .0238726790450928;
MilitaryFWA : .01326259469496; MilitaryWheeled : .075066129734748;
HumanWithWeapon : .01326259469496; MilitaryWheeled : .0079575596816976;
CommercialTracked : .0079575596816976; CommercialWheeled : .0055702917718833;
HumanWithoutWeapon : .02917718328912; Others : .0238726790450928;
CommercialWheeled : .0397877984084881;  }
else if( TargetImage_g1 == CommercialWheeled && DistanceToSensor_g1_sl_t1 == Short )
{CommercialFWA : .0909090909090909; MilitaryWheeled : .0909090909090909;
MilitaryFWA : .0909090909090909; MilitaryFWA : .0909090909090909;
HumanWithWeapon : .0909090909090909; MilitaryTracked : .0909090909090909;
CommercialTracked : .0909090909090909; CommercialWheeled : .0909090909090909;
HumanWithoutWeapon : .0909090909090909; Others : .0909090909090909;
CommercialWheeled : .0909090909090909;  }
else if( TargetImage_g1 == CommercialFWA && DistanceToSensor_g1_sl_t1 == Short )
{CommercialFWA : .0909090909090909; MilitaryWheeled : .0909090909090909;
MilitaryFWA : .0909090909090909; MilitaryFWA : .0909090909090909;
HumanWithWeapon : .0909090909090909; MilitaryTracked : .0909090909090909;
CommercialTracked : .0909090909090909; CommercialWheeled : .0909090909090909;
CommercialWheeled : .0909090909090909;  }
else if( TargetImage_g1 == HumanWithWeapon && DistanceToSensor_g1_sl_t1 == Short )
{CommercialFWA : .0909090909090909; MilitaryWheeled : .0909090909090909;
MilitaryFWA : .0909090909090909; MilitaryFWA : .0909090909090909;
HumanWithWeapon : .0909090909090909; MilitaryTracked : .0909090909090909;
CommercialTracked : .0909090909090909; CommercialWheeled : .0909090909090909;
CommercialWheeled : .0909090909090909;  }
else if( TargetImage_g1 == CommercialFWA && DistanceToSensor_g1_sl_t1 == Mid )
{CommercialFWA : .0909090909090909; MilitaryWheeled : .0909090909090909;
MilitaryFWA : .0909090909090909; MilitaryFWA : .0909090909090909;
HumanWithWeapon : .0909090909090909; MilitaryTracked : .0909090909090909;
CommercialTracked : .0909090909090909; CommercialWheeled : .0909090909090909;
CommercialWheeled : .0909090909090909;  }
else if( TargetImage_g1 == HumanWithWeapon && DistanceToSensor_g1_sl_t1 == Short )
{CommercialFWA : .0909090909090909; MilitaryWheeled : .0909090909090909;
MilitaryFWA : .0909090909090909; MilitaryFWA : .0909090909090909;
HumanWithWeapon : .0909090909090909; MilitaryTracked : .0909090909090909;
CommercialTracked : .0909090909090909; CommercialWheeled : .0909090909090909;
CommercialWheeled : .0909090909090909;  }
MilitaryRWA : .0909090909090909; MilitaryFWA : .0909090909090909;
HumanWithWeapon : .0909090909090909; MilitaryTracked : .0909090909090909;
CommercialTracked : .0909090909090909; CommercialWheeled : .0909090909090909;
HumanWithoutWeapon : .0909090909090909; Others : .0909090909090909;
CommercialRWA : .0909090909090909;  }
else if( TargetImage_g1 == Others && DistanceToSensor_g1_s1_t1 == Short )
{CommercialFWA : .0210366025604873; MilitaryWheeled : .027578816761604;
MilitaryRWA : .0213749929502002; MilitaryFWA : .0256612712198974;
HumanWithWeapon : .0244205064576166; MilitaryTracked : .0213749929502002;
CommercialTracked : .0237437256781907; CommercialWheeled : .017314308273645;
HumanWithoutWeapon : .0214877897467712; Others : .7726016581129096;
CommercialRWA : .0234053352884778;  }
else if( TargetImage_g1 == MilitaryFWA && DistanceToSensor_g1_s1_t1 == Short )
{CommercialFWA : .0309951060358891; MilitaryWheeled : .00163132137031;
MilitaryRWA : .0081566068515498; MilitaryFWA : .8303425774877651;
HumanWithWeapon : .0114192495921697; MilitaryTracked : .0309951060358891;
CommercialTracked : .0114192495921697; CommercialWheeled : .0277324632952692;
HumanWithoutWeapon : .0114192495921697; Others : .0146818923327896;
CommercialRWA : .0212207178140294;  }
else if( TargetImage_g1 == CommercialFWA && DistanceToSensor_g1_s1_t1 == Long )
{CommercialFWA : .6992481203007519; MilitaryWheeled : .037593984962406;
MilitaryRWA : .037593984962406; MilitaryFWA : .0526315789473684;
HumanWithWeapon : .0075187969924812; MilitaryTracked : .0526315789473684;
CommercialTracked : .037593984962406; CommercialWheeled : .0075187969924812;
HumanWithoutWeapon : .0526315789473684; Others : .0075187969924812;
CommercialRWA : .0075187969924812;  }
else if( TargetImage_g1 == CommercialWheeled && DistanceToSensor_g1_s1_t1 == Long )
{CommercialFWA : .0909090909090909; MilitaryWheeled : .0909090909090909;
MilitaryRWA : .0909090909090909; MilitaryFWA : .0909090909090909;
HumanWithWeapon : .0909090909090909; MilitaryTracked : .0909090909090909;
CommercialTracked : .0909090909090909; CommercialWheeled : .0909090909090909;
HumanWithoutWeapon : .0909090909090909; Others : .0909090909090909;
CommercialRWA : .0909090909090909;  }
else if( TargetImage_g1 == CommercialFWA && DistanceToSensor_g1_s1_t1 == Mid )
{CommercialFWA : .6448598130841121; MilitaryWheeled : .046728971962168;
MilitaryRWA : .046728971962168; MilitaryFWA : .0654205607476635;
HumanWithWeapon : .046728971962168; MilitaryTracked : .0654205607476635;
CommercialTracked : .0093457943925234; CommercialWheeled : .046728971962168;
HumanWithoutWeapon : .0093457943925234; Others : .0093457943925234;
CommercialRWA : .0093457943925234;  }
else if( TargetImage_g1 == MilitaryFWA && DistanceToSensor_g1_s1_t1 == Mid )
{CommercialFWA : .0361726954492415; MilitaryWheeled : .0338389731621937;
MilitaryRWA : .0338389731621937; MilitaryFWA : .0350338045482483;
HumanWithWeapon : .0268378063010502; MilitaryWheeled : .731621936984983;
HumanWithoutWeapon : .0315052508751459; MilitaryWheeled : .0175029171528588;
CommercialTracked : .0221703617269545; CommercialWheeled : .01516914865811;
HumanWithoutWeapon : .0245040840140023; Others : .029171528588089;
CommercialRWA : .0315052508751459;  }
else if( TargetImage_g1 == HumanWithWeapon && DistanceToSensor_g1_s1_t1 == Mid )
{CommercialFWA : .0620774431468961; MilitaryWheeled : .025199754148747;
MilitaryRWA : .0374923171481254; MilitaryFWA : .031346035648327;
HumanWithWeapon : .675476368162266; MilitaryWheeled : .0350338045482483;
CommercialTracked : .0153657034792317; CommercialWheeled : .0264290104486786;
HumanWithoutWeapon : .0239704978488015; Others : .0460971112476951;
CommercialRWA : .0215119852489244;  }
else if( TargetImage_g1 == Others && DistanceToSensor_g1_s1_t1 == Mid )
{CommercialFWA : .0260949438436808; MilitaryWheeled : .024758827188844;
MilitaryRWA : .029435096655672; MilitaryFWA : .0268464782263789;
HumanWithWeapon : .0266794705857793; MilitaryWheeled : .028934073338733;
CommercialTracked : .028850569135735; CommercialWheeled : .028260431714751;
HumanWithoutWeapon : .0215857375474928; Others : .730950690994113;
defineNode(TemperatureReport_g1_s1_t1, Desc);
{ defineState(Continuous);
p( TemperatureReport_g1_s1_t1 | DistanceToSensor_g1_s1_t1, Temperature_g1_t1 ) =
if( DistanceToSensor_g1_s1_t1 == Short ) { 1.0 * Temperature_g1_t1 +
    NormalDist(0.0, 0.000017080785743); }
else if( DistanceToSensor_g1_s1_t1 == Mid ) { 1.0 * Temperature_g1_t1 +
    NormalDist(0.0, 0.000025816947423); }
else if( DistanceToSensor_g1_s1_t1 == Long ) { 1.0 * Temperature_g1_t1 +
    NormalDist(0.0, 0.00004728836748); }
}

defineNode(SizeReport_g1_s1_t1, Desc);
{ defineState(Continuous);
p( SizeReport_g1_s1_t1 | DistanceToSensor_g1_s1_t1, TargetSize_g1 ) =
if( DistanceToSensor_g1_s1_t1 == Short ) { 1.0 * TargetSize_g1 +
    NormalDist(0.0, 0.000017311652489); }
else if( DistanceToSensor_g1_s1_t1 == Mid ) { 1.0 * TargetSize_g1 +
    NormalDist(0.0, 0.000025376643893); }
else if( DistanceToSensor_g1_s1_t1 == Long ) { 1.0 * TargetSize_g1 +
    NormalDist(0.0, 0.000049496600574); }
}

defineNode(RegionTypeReport_g1_s1_t1, Desc);
{ defineState(Discrete, offroad, road);
p( RegionTypeReport_g1_s1_t1 | DistanceToSensor_g1_s1_t1, RegionType_g1_t1 ) =
if( RegionType_g1_t1 == OffRoad && DistanceToSensor_g1_s1_t1 == Long )
    { offroad : .9550898203592815; road : .449101796407186; }
else if( RegionType_g1_t1 == road && DistanceToSensor_g1_s1_t1 == Short )
    { offroad : .5; road : .5; }
else if( RegionType_g1_t1 == road && DistanceToSensor_g1_s1_t1 == Long )
    { offroad : .5; road : .5; }
else if( RegionType_g1_t1 == OffRoad && DistanceToSensor_g1_s1_t1 == Short )
    { offroad : .9783419391853126; road : .0216580608146873; }
else if( RegionType_g1_t1 == OffRoad && DistanceToSensor_g1_s1_t1 == Mid )
    { offroad : .933857864186633; road : .066141235813367; }
else if( RegionType_g1_t1 == road && DistanceToSensor_g1_s1_t1 == Mid )
    { offroad : .1458333333333333; road : .8541666666666666; }
}

defineNode(DistanceToSensor_g1_s1_t2, Desc);
{ defineState(Discrete, Mid, Long, Short);
p( DistanceToSensor_g1_s1_t2 ) =
    { Mid : .5780996240183319; Long : .0972435426817059;
    Short : .324568332999622; }
}

defineNode(Mission_g1_t2, Desc);
{ defineState(Discrete, Others, Attack);
p( Mission_g1_t2 | TargetType_g1, Mission_g1_t1 ) =
if( Mission_g1_t1 == Attack && TargetType_g1 == ThreateningGroundTarget )
    { Others : .0002977963073258; Attack : .9997022036926743; }
else if( Mission_g1_t1 == Others && TargetType_g1 == ThreateningGroundTarget )
    { Others : .5; Attack : .5; }
else if( Mission_g1_t1 == Others && TargetType_g1 == ThreateningAirTarget )
    { Others : .5; Attack : .5; }
else if( Mission_g1_t1 == Attack && TargetType_g1 == Others )
    { Others : .0000344970332551; Attack : .9999655029667449; }
else if( Mission_g1_t1 == Attack && TargetType_g1 == ThreateningAirTarget )
    { Others : .5; Attack : .5; }
else if( Mission_g1_t1 == Attack && TargetType_g1 == ThreateningGroundTarget )
    { Others : .0000049496600574; Attack : .99995980126090769; }
}
defineNode(Activity_g1_t2 , Desc);
{ defineState(Discrete, HeadingToCI, HeadingToOthers, Staying );
p( Activity_g1_t2 | TargetType_g1, Mission_g1_t2, Activity_g1_t1 ) =
    if( Activity_g1_t1 == HeadingToOthers && Mission_g1_t2 == Others &&
        TargetType_g1 == Others ) {HeadingToCI : .0018475531763703;
        HeadingToOthers : .869155365862902; Staying : .128997112373395; }
    else if( Activity_g1_t1 == HeadingToCI && Mission_g1_t2 == Others &&
             TargetType_g1 == ThreateningGroundTarget ) {HeadingToCI : .3333333333333333;
                HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
    else if( Activity_g1_t1 == HeadingToCI && Mission_g1_t2 == Attack &&
             TargetType_g1 == ThreateningAirTarget ) {HeadingToCI : .936962757016324;
                HeadingToOthers : .0601719707736; Staying : .0028613295128949; }
    else if( Activity_g1_t1 == Staying && Mission_g1_t2 == Others && TargetType_g1 ==
              Others ) {HeadingToCI : .105794283780735;
               HeadingToOthers : .619636814798105; Staying : .27723890142116; }
    else if( Activity_g1_t1 == HeadingToOthers && Mission_g1_t2 == Attack &&
             TargetType_g1 == ThreateningAirTarget ) {HeadingToCI : .0166808616174583;
                HeadingToOthers : .98202841335045; Staying : .0012836970474968; }
    else if( Activity_g1_t1 == Staying && Mission_g1_t2 == Others &&
             TargetType_g1 == ThreateningGroundTarget ) {HeadingToCI : .3333333333333333;
                HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
    else if( Activity_g1_t1 == HeadingToCI && Mission_g1_t2 == Attack &&
             TargetType_g1 == ThreateningGroundTarget ) {HeadingToCI : .3333333333333333;
                HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
    else if( Activity_g1_t1 == HeadingToCI && Mission_g1_t2 == Others &&
             TargetType_g1 == ThreateningAirTarget ) {HeadingToCI : .3333333333333333;
                HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
    else if( Activity_g1_t1 == Staying && Mission_g1_t2 == Attack &&
             TargetType_g1 == ThreateningGroundTarget ) {HeadingToCI : .3333333333333333;
                HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
    else if( Activity_g1_t1 == HeadingToOthers && Mission_g1_t2 == Attack &&
             TargetType_g1 == Others ) {HeadingToCI : .001150747986191;
                HeadingToOthers : .94703697986191; Staying : .0517636598561; }
    else if( Activity_g1_t1 == HeadingToCI && Mission_g1_t2 == Others &&
             TargetType_g1 == ThreateningAirTarget ) {HeadingToCI : .3333333333333333;
                HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
    else if( Activity_g1_t1 == Staying && Mission_g1_t2 == Attack &&
             TargetType_g1 == Others ) {HeadingToCI : .3333333333333333;
                HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
    else if( Activity_g1_t1 == HeadingToOthers && Mission_g1_t2 == Attack &&
             TargetType_g1 == ThreateningGroundTarget ) {HeadingToCI : .001150747986191;
                HeadingToOthers : .94703697986191; Staying : .0517636598561; }
    else if( Activity_g1_t1 == HeadingToCI && Mission_g1_t2 == Others &&
             TargetType_g1 == ThreateningAirTarget ) {HeadingToCI : .3333333333333333;
                HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
    else if( Activity_g1_t1 == Staying && Mission_g1_t2 == Attack &&
             TargetType_g1 == Others ) {HeadingToCI : .3333333333333333;
                HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
    else if( Activity_g1_t1 == HeadingToOthers && Mission_g1_t2 == Attack &&
             TargetType_g1 == ThreateningGroundTarget ) {HeadingToCI : .001150747986191;
                HeadingToOthers : .94703697986191; Staying : .0517636598561; }
    else if( Activity_g1_t1 == HeadingToCI && Mission_g1_t2 == Others &&
             TargetType_g1 == ThreateningAirTarget ) {HeadingToCI : .3333333333333333;
                HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
    else if( Activity_g1_t1 == Staying && Mission_g1_t2 == Attack &&
             TargetType_g1 == Others ) {HeadingToCI : .3333333333333333;
                HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
}
defineNode(RegionType_g1_t2 , Desc);
{ defineState(Discrete, OffRoad, road );
p( RegionType_g1_t2 | RegionType_g1_t1 ) =
if( RegionType_g1_t1 == OffRoad ) {OffRoad : .9994304214880988;
road : .0005695785119012; }
else if( RegionType_g1_t1 == road ) {OffRoad : .1363636363636364;
road : .8636363636363636; }
}

defineNode(Latitude_Velocity_g1_t2 , Desc);
{ defineState(Continuous);
p( Latitude_Velocity_g1_t2 | TargetType_g1 , Activity_g1_t2 , RegionType_g1_t2 ,
Latitude_Velocity_g1_t1 ) =
if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToCI && TargetType_g1
== Others ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0000093750);
}else if( RegionType_g1_t2 == road && Activity_g1_t2 == Staying &&
TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.000011636810599);
}else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToCI &&
TargetType_g1 == ThreateningAirTarget ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0012610798615335);
}else if( RegionType_g1_t2 == road && Activity_g1_t2 == Staying &&
TargetType_g1 == ThreateningAirTarget ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0000093750);
}else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == Staying &&
TargetType_g1 == ThreateningAirTarget ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0000093750);
}else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToCI &&
TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.001261594092696);
}else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == Staying &&
TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0000093750);
}else if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToCI &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.000077584390404);
}else if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToOthers &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0000093750);
}else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToCI &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0019070311194707);
}else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == Staying &&
TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0000093750);
}else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToCI &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.000085380819263);
}else if( RegionType_g1_t2 == road && Activity_g1_t2 == Staying &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.000069016442511);
}else if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToOthers &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0000093750);
}else if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0000093750);
else if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0, 0.001700018869248);
} else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000275399632545);
})

defineNode(Longitude_Velocity_g1_t2 , Desc);
{ defineState(Continuous);
p( Longitude_Velocity_g1_t2 | TargetType_g1 , Activity_g1_t2 , RegionType_g1_t2 , Longitude_Velocity_g1_t1  ) =
if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToCI && TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0, 0.01446954353104);
} else if( RegionType_g1_t2 == road && Activity_g1_t2 == Staying && TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0, 0.0001621872);
} else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0001621872);
} else if( RegionType_g1_t2 == road && Activity_g1_t2 == Staying && TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0001621872);
} else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToCI && TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0, 0.0113133612741177);
} else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == Staying && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0, 0.0000049719336356);
} else if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToCI && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000049719336356);
} else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0, 0.0001621872);
} else if( RegionType_g1_t2 == road && Activity_g1_t2 == Staying && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0, 0.0113133612741177);
} else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == Staying && TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0001621872);
} else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0001621872);
} else if( RegionType_g1_t2 == road && Activity_g1_t2 == Staying && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0, 0.0000049719336356);
} else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == Staying && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0, 0.0000049719336356);
} else if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0001621872);
} else if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0, 0.0000049719336356);
} else if( RegionType_g1_t2 == road && Activity_g1_t2 == Staying && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0001621872);
} else if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0001621872);
} else if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0001621872);
} else if( RegionType_g1_t2 == road && Activity_g1_t2 == Staying && TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0001621872);
} else if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0 , 0.0001621872);
}
TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0, 0.0022134952244327); } else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t1 + NormalDist( 0.0, 0.00193765071369); }
})
defineNode(Altitude_Velocity_g1_t2 , Desc);
{ defineState(Continuous);
p( Altitude_Velocity_g1_t2 | TargetType_g1 , Activity_g1_t2 , RegionType_g1_t2 , Altitude_Velocity_g1_t1 ) =
if( RegionType_g1_t2 == road && Activity_g1_t2 == HeadingToCI && TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000069813111472); }
else if( RegionType_g1_t2 == road && Activity_g1_t2 == Staying && TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000082176 ); }
else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == ThreateningAirTarget ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000082176 ); }
else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == ThreateningAirTarget ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000082176 ); }
else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToCI && TargetType_g1 == ThreateningAirTarget ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000082176 ); }
else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == Staying && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000082176 ); }
else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToCI && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000082176 ); }
else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == Staying && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000082176 ); }
else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToCI && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000082176 ); }
else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000082176 ); }
else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == Staying && TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000082176 ); }
else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000082176 ); }
else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == Staying && TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000082176 ); }
else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToOthers && TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t1 + NormalDist( 0.0, 0.000082176 ); }
else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == Staying && TargetType_g1 == Other
NormalDist( 0.0, .0000085185606717);
} else if( RegionType_g1_t2 == OffRoad && Activity_g1_t2 == HeadingToOthers &&
  TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t1 +
  NormalDist( 0.0, .000082998632358);
}}
}

defineNode(Latitude_g1_t2, Desc);
{ defineState(Continuous);
  p( Latitude_g1_t2 | Latitude_Velocity_g1_t1, Latitude_g1_t1 ) =
  1.0 * Latitude_Velocity_g1_t1 + 1.0 * Latitude_g1_t1 + NormalDist( 0.0,
  0.0000001);
}

defineNode(Longitude_g1_t2, Desc);
{ defineState(Continuous);
  p( Longitude_g1_t2 | Longitude_Velocity_g1_t1, Longitude_g1_t1 ) =
  1.0 * Longitude_Velocity_g1_t1 + 1.0 * Longitude_g1_t1 + NormalDist( 0.0,
  0.0000001);
}

defineNode(Altitude_g1_t2, Desc);
{ defineState(Continuous);
  p( Altitude_g1_t2 | Altitude_Velocity_g1_t1, Altitude_g1_t1 ) =
  1.0 * Altitude_Velocity_g1_t1 + 1.0 * Altitude_g1_t1 + NormalDist( 0.0,
  0.0000001);
}

defineNode(DistanceToCI_g1_t2, Desc);
{ defineState(Continuous);
  p( DistanceToCI_g1_t2 | Activity_g1_t2, DistanceToCI_g1_t1 ) =
  if( Activity_g1_t2 == HeadingToOthers ) {1.002575422880476 * DistanceToCI_g1_t1
  + NormalDist( 0.0, 290571.93849371106);
  } else if( Activity_g1_t2 == HeadingToCI ) {.991535242451743 *
  DistanceToCI_g1_t1 + NormalDist( 0.0, 292618.24472405633);
  } else if( Activity_g1_t2 == Staying ) {1.0000012673057117 * DistanceToCI_g1_t1
  + NormalDist( 0.0, 161584.45531205033);
  }
}

defineNode(DirectionToCI_g1_t2, Desc);
{ defineState(Continuous);
  p( DirectionToCI_g1_t2 | Activity_g1_t2, DirectionToCI_g1_t1 ) =
  if( Activity_g1_t2 == HeadingToOthers ) {.9813365016030442 *
  DirectionToCI_g1_t1 + NormalDist( 0.0, 1422.7267805013084);
  } else if( Activity_g1_t2 == HeadingToCI ) {.1303463969940752 *
  DirectionToCI_g1_t1 + NormalDist( 0.0, 96.94295604746321);
  } else if( Activity_g1_t2 == Staying ) {1.0 * DirectionToCI_g1_t1 +
  NormalDist( 0.0, 0.000000001);
  }
}

defineNode(Temperature_g1_t2, Desc);
{ defineState(Continuous);
  p( Temperature_g1_t2 | Activity_g1_t2, Temperature_g1_t1 ) =
  if( Activity_g1_t2 == HeadingToOthers ) {1.0000008891419039 *
  Temperature_g1_t1 + NormalDist( 0.0, 145.56334034218463);
  } else if( Activity_g1_t2 == HeadingToCI ) {1.0000034017283523 *
  Temperature_g1_t1 + NormalDist( 0.0, 91.81289968886823);
  } else if( Activity_g1_t2 == Staying ) {1.0000068754977992 * Temperature_g1_t1
  + NormalDist( 0.0, 21.75286278479968);
  }
}

defineNode(DistanceToSensor_g1_s1_t3, Desc);
{ defineState(Continuous);
  p( DistanceToSensor_g1_s1_t3 ) =
  if( Temperature_g1_t2 == Attack && TargetType_g1 == ThreateningGroundTarget )
{Others : .0002977963073258; Attack : .9997022036926743; }
else if( Mission_g1_t2 == Others && TargetType_g1 == ThreateningAirTarget )
{Others : .5; Attack : .5; }
else if( Mission_g1_t2 == Others && TargetType_g1 == ThreateningGroundTarget )
{Others : .5; Attack : .5; }
else if( Mission_g1_t2 == Others && TargetType_g1 == Others )
{Others : .9999655029667449; Attack : .0000344970332551; }
else if( Mission_g1_t2 == Attack && TargetType_g1 == Others )
{Others : .5; Attack : .5; }
else if( Mission_g1_t2 == Attack && TargetType_g1 == ThreateningAirTarget )
{Others : .0008896797153025; Attack : .9991103202846975; }

defineNode(Activity_g1_t3 , Desc);
{ defineState(Discrete, HeadingToCI, HeadingToOthers, Staying );
p(Activity_g1_t3 | TargetType_g1 , Mission_g1_t3 , Activity_g1_t2 ) =
if( Activity_g1_t2 == HeadingToOthers && Mission_g1_t3 == Others &&
TargetType_g1 == Others ) {HeadingToCI : .0018475531763703;
HeadingToOthers : .8691553365862902; Staying : .1289971102373395; }
else if( Activity_g1_t2 == HeadingToCI &
Mission_g1_t3 == Others &&
TargetType_g1 == ThreateningGroundTarget ) {HeadingToCI : .3333333333333333;
HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t2 == ThreateningAirTarget )
{HeadingToCI : .93696257016324; HeadingToOthers : .06017197707736; Staying : .00286529512894;
else if( Activity_g1_t2 == Staying & Mission_g1_t3 == Others && TargetType_g1 == Others )
{HeadingToCI : .1057974283780735; HeadingToOthers : .616936814798105; Staying : .277238890412116; }
else if( Activity_g1_t2 == HeadingToOthers &
Mission_g1_t3 == Attack &
TargetType_g1 == ThreateningAirTarget )
{HeadingToCI : .016680616174583; HeadingToOthers : .98208243135045; Staying : .0012836970474968; }
else if( Activity_g1_t2 == Staying & Mission_g1_t3 == Others &&
TargetType_g1 == ThreateningGroundTarget )
{HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t2 == ThreateningAirTarget )
{HeadingToCI : .9572466866182129; HeadingToOthers : .0021376656690894; Staying : .0040156476712697; }
else if( Activity_g1_t2 == Staying & Mission_g1_t3 == Others & TargetType_g1 == Others )
{HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t2 == HeadingToOthers &
Mission_g1_t3 == Attack &
TargetType_g1 == ThreateningGroundTarget )
{HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t2 == HeadingToOthers &
Mission_g1_t3 == Others &
TargetType_g1 == Others )
{HeadingToCI : .841066629018245; HeadingToOthers : .15299497538372; Staying : .0058329521279178; }
else if( Activity_g1_t2 == ThreateningAirTarget )
{HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t2 == Staying & Mission_g1_t3 == Attack &
TargetType_g1 == ThreateningGroundTarget )
{HeadingToCI : .605095414012739; HeadingToOthers : .28662403821656; Staying : .108280254770701; }
else if( Activity_g1_t2 == Staying & Mission_g1_t3 == Others &
TargetType_g1 == ThreateningAirTarget )
{HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t2 == Staying & Mission_g1_t3 == Attack &
TargetType_g1 == Others )
{HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t2 == Staying & Mission_g1_t3 == ThreateningAirTarget )
{HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t2 == Staying & Mission_g1_t3 == ThreateningGroundTarget )
{HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t2 == HeadingToOthers && Mission_g1_t3 == Attack && TargetType_g1 == ThreateningGroundTarget ) {HeadingToCI : .001150747986191; HeadingToOthers : .9470655926352128; Staying : .0517836593785961; }
else if( Activity_g1_t2 == HeadingToCI && Mission_g1_t3 == Attack && TargetType_g1 == ThreateningAirTarget ) {HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t2 == HeadingToCI && Mission_g1_t3 == Attack && TargetType_g1 == Others ) {HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
}

defineNode(RegionType_g1_t3 , Desc);
{ defineState(Discrete, OffRoad, road );
p( RegionType_g1_t3 | RegionType_g1_t2  ) =
if( RegionType_g1_t2 == OffRoad ) {OffRoad : .9994304214880988; road : .0005695785119012; } else if( RegionType_g1_t2 == road ) {OffRoad : .1363636363636364; road : .8636363636363636; }
}

defineNode(Latitude_Velocity_g1_t3 , Desc);
{ defineState(Continuous);
p( Latitude_Velocity_g1_t3 | TargetType_g1 , Activity_g1_t3 , RegionType_g1_t3 , Latitude_Velocity_g1_t2  ) =
if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToCI && TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t2 + NormalDist( 0.0 , 0.0000093750); }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying && TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t2 + NormalDist( 0.0 , 0.0000093750); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToOther &
TargetType_g1 == ThreateningAirTarget ) {1.0 * Latitude_Velocity_g1_t2 + NormalDist( 0.0 , 0.001261594092696); }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToCI && TargetType_g1 == ThreateningAirTarget ) {1.0 * Latitude_Velocity_g1_t2 + NormalDist( 0.0 , 0.0000093750); }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying && TargetType_g1 == ThreateningAirTarget ) {1.0 * Latitude_Velocity_g1_t2 + NormalDist( 0.0 , 0.0000093750); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == Staying && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t2 + NormalDist( 0.0 , 0.0000093750); }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t2 + NormalDist( 0.0 , 0.0000093750); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToOther &
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t2 + NormalDist( 0.0 , 0.0000093750); }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToOthers && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t2 + NormalDist( 0.0 , 0.0019070311194707); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToCI && TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t2 + NormalDist( 0.0 , 0.0000093750); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == Staying && TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t2 + NormalDist( 0.0 , 0.0000093750); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToCI && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t2 + NormalDist( 0.0 , 0.0000093750); }
NormalDist( 0.0, 0.000085380819263);
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToOthers &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.000069016442511);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0000093750);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToOthers &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0000069016442511);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0000093750);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToOthers &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0000069016442511);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0000093750);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToOthers &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0000069016442511);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0000093750);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToOthers &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0000069016442511);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0000093750);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToOthers &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0000069016442511);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0000093750);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToOthers &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0000069016442511);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Latitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0000093750);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToOthers &&
        TargetType_g1 == ThreateningAirTarget )
        { 1.0 * Longitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.001446954353104);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying &&
        TargetType_g1 == Others )
        { 1.0 * Longitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0001621872);
    }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToOthers &&
        TargetType_g1 == ThreateningAirTarget )
        { 1.0 * Longitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.01881367813414);
    }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToCI &&
        TargetType_g1 == ThreateningAirTarget )
        { 1.0 * Longitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.001446954353104);
    }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToOthers &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Longitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.001001621872);
    }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == Staying &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Longitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0001621872);
    }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToCI &&
        TargetType_g1 == ThreateningAirTarget )
        { 1.0 * Longitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.001001621872);
    }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToOthers &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Longitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0001621872);
    }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == Staying &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Longitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0001621872);
    }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToCI &&
        TargetType_g1 == ThreateningAirTarget )
        { 1.0 * Longitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.001001621872);
    }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToOthers &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Longitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0001621872);
    }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == Staying &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Longitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.0001621872);
    }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToCI &&
        TargetType_g1 == ThreateningGroundTarget )
        { 1.0 * Longitude_Velocity_g1_t2 +
          NormalDist( 0.0, 0.000137107925352);
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToOthers &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.000009086095747); }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.0001621872); }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.0001621872); }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToOthers &&
TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.0022134952244327); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToOthers &&
TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.00193765071369); }
}

defineNode(Altitude_Velocity_g1_t3 , Desc); { defineState(Continuous);
p( Altitude_Velocity_g1_t3 | TargetType_g1 , Activity_g1_t3 , RegionType_g1_t3 ,
Altitude_Velocity_g1_t2 ) =
if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToCI && TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.000069813111472); }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying && TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.0000082176 ); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {1.0 * Altitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.000085195886801); }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToCI &&
TargetType_g1 == ThreateningAirTarget ) {1.0 * Altitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.00000082176 ); }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying && TargetType_g1 == ThreateningAirTarget ) {1.0 * Altitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.0000082176 ); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == Staying &&
TargetType_g1 == ThreateningAirTarget ) {1.0 * Altitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.0000849686892); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToCI &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.000086826898311); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToOthers &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.0000821225194323); }
else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToOthers &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.000085170791182); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToCI &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.000083114682263); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToCI &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t2 +
NormalDist( 0.0, 0.000084681077152); }
else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToOthers &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t2 + NormalDist( 0.0 , .0000079108212487 );
) else if( RegionType_g1_t3 == road && Activity_g1_t3 == Staying && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t2 + NormalDist( 0.0 , .0000079108212487 );
) else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToOthers && TargetType_g1 == ThreateningAirTarget ) {1.0 * Altitude_Velocity_g1_t2 + NormalDist( 0.0 , .0000082176388652 );
) else if( RegionType_g1_t3 == road && Activity_g1_t3 == HeadingToOthers && TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t2 + NormalDist( 0.0 , .0000085185606717 );
) else if( RegionType_g1_t3 == OffRoad && Activity_g1_t3 == HeadingToOthers && TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t2 + NormalDist( 0.0 , .0000082176388652 );
)}}
defineNode(Latitude_g1_t3 , Desc);
{ defineState(Continuous);
p( Latitude_g1_t3 | Latitude_Velocity_g1_t2 , Latitude_g1_t2 ) = 1.0 * Latitude_Velocity_g1_t2 + 1.0 * Latitude_g1_t2 + NormalDist( 0.0 , 0.0000001 );
} defineNode(Longitude_g1_t3 , Desc);
{ defineState(Continuous);
p( Longitude_g1_t3 | Longitude_Velocity_g1_t2 , Longitude_g1_t2 ) = 1.0 * Longitude_Velocity_g1_t2 + 1.0 * Longitude_g1_t2 + NormalDist( 0.0 , 0.0000001 );
} defineNode(Altitude_g1_t3 , Desc);
{ defineState(Continuous);
p( Altitude_g1_t3 | Altitude_Velocity_g1_t2 , Altitude_g1_t2 ) = 1.0 * Altitude_Velocity_g1_t2 + 1.0 * Altitude_g1_t2 + NormalDist( 0.0 , 0.0000001 );
} defineNode(DistanceToCI_g1_t3 , Desc);
{ defineState(Continuous);
p( DistanceToCI_g1_t3 | Activity_g1_t3 , DistanceToCI_g1_t2 ) = if( Activity_g1_t3 == HeadingToOthers ) {1.002575422880476 * DistanceToCI_g1_t2 + NormalDist( 0.0 , 290571.93849371106 );
) else if( Activity_g1_t3 == HeadingToCI ) {.991535242451743 * DistanceToCI_g1_t2 + NormalDist( 0.0 , 292618.24472405633 );
) else if( Activity_g1_t3 == Staying ) {1.0000012673057117 * DistanceToCI_g1_t2 + NormalDist( 0.0 , 161584.45531205033 );
)}}
defineNode(DirectionToCI_g1_t3 , Desc);
{ defineState(Continuous);
p( DirectionToCI_g1_t3 | Activity_g1_t3 , DirectionToCI_g1_t2 ) = if( Activity_g1_t3 == HeadingToOthers ) {.9813365016030442 * DirectionToCI_g1_t2 + NormalDist( 0.0 , 1422.7267805013084 );
) else if( Activity_g1_t3 == HeadingToCI ) {.1303463969940752 * DirectionToCI_g1_t2 + NormalDist( 0.0 , 96.94295604746321 );
) else if( Activity_g1_t3 == Staying ) {1.0 * DirectionToCI_g1_t2 + NormalDist( 0.0 , 0.00000001 );
)}}
defineNode(Temperature_g1_t3 , Desc);
{ defineState(Continuous);
p( Temperature_g1_t3 | Activity_g1_t3 , Temperature_g1_t2 ) = if( Activity_g1_t3 == HeadingToOthers ) {1.0000008891419039 * Temperature_g1_t2 + NormalDist( 0.0 , 145.56334034218463 );
) else if( Activity_g1_t3 == HeadingToCI ) {1.0000034017283523 * Temperature_g1_t2 + NormalDist( 0.0 , 91.81289968888888 );
) else if( Activity_g1_t3 == Staying ) {1.0000068754977902 * Temperature_g1_t2 + NormalDist( 0.0 , 21.75286278479968 );
)}}
defineNode(DistanceToSensor_g1_s1_t4, Desc);
{ defineState(Discrete, Mid, Long, Short);
p( DistanceToSensor_g1_s1_t4 ) =
{ Mid : .578096240183319; Long : .0972435426817059;
  Short : .3246568332999622; } }

defineNode(Mission_g1_t4, Desc);
{ defineState(Discrete, Others, Attack);
p( Mission_g1_t4 | TargetType_g1, Mission_g1_t3 ) =
if( Mission_g1_t3 == Attack && TargetType_g1 == ThreateningGroundTarget )
{ Others : .0002977963073258; Attack : .9997022036926743; }
else if( Mission_g1_t3 == Others && TargetType_g1 == ThreateningAirTarget )
{ Others : .5; Attack : .5; }
else if( Mission_g1_t3 == Others && TargetType_g1 == ThreateningGroundTarget )
{ Others : .5; Attack : .5; }
else if( Mission_g1_t3 == Attack && TargetType_g1 == Others )
{ Others : .9999655029667449; Attack : .0000344970332551; }
else if( Mission_g1_t3 == Others && TargetType_g1 == Others )
{ Others : .999910320846975; Attack : .0000896797153025; }
}

defineNode(Activity_g1_t4, Desc);
{ defineState(Discrete, HeadingToCI, HeadingToOthers, Staying);
p( Activity_g1_t4 | TargetType_g1, Mission_g1_t4, Activity_g1_t3 ) =
if( Activity_g1_t3 == HeadingToOthers && Mission_g1_t4 == Others && TargetType_g1 == ThreateningGroundTarget )
{ HeadingToCI : .0018475531763703; HeadingToOthers : .8691553365862902; Staying : .1289971023733959; }
else if( Activity_g1_t3 == HeadingToCI && Mission_g1_t4 == Others && TargetType_g1 == ThreateningGroundTarget )
{ HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t3 == Staying && Mission_g1_t4 == Others && TargetType_g1 == ThreateningGroundTarget )
{ HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t3 == HeadingToCI && Mission_g1_t4 == Others && TargetType_g1 == ThreateningAirTarget )
{ HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t3 == Staying && Mission_g1_t4 == Others && TargetType_g1 == ThreateningAirTarget )
{ HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t3 == HeadingToCI && Mission_g1_t4 == Attack && TargetType_g1 == ThreateningGroundTarget )
{ HeadingToCI : .9572466866182129; HeadingToOthers : .0021376656690894; Staying : .0406156477126977; }
else if( Activity_g1_t3 == HeadingToCI && Mission_g1_t4 == Others && TargetType_g1 == ThreateningAirTarget )
{ HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t3 == Staying && Mission_g1_t4 == Others && TargetType_g1 == ThreateningAirTarget )
{ HeadingToCI : .841876629018245; HeadingToOthers : .022299497538372; Staying : .1358239212279178; }
else if( Activity_g1_t3 == HeadingToCI && Mission_g1_t4 == Others && TargetType_g1 == ThreateningGroundTarget )
{ HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t3 == Staying && Mission_g1_t4 == Others && TargetType_g1 == ThreateningGroundTarget )
{ HeadingToCI : .6050955414012739; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t3 == HeadingToCI && Mission_g1_t4 == Attack && TargetType_g1 == ThreateningGroundTarget )
{ HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
HeadingToOthers : .286624203821656; Staying : .1082802547770701; }
else if( Activity_g1_t3 == Staying && Mission_g1_t4 == Others && TargetType_g1 == ThreateningAirTarget ) {HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t3 == HeadingToOthers && Mission_g1_t4 == Attack && TargetType_g1 == Others ) {HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t3 == Staying && Mission_g1_t4 == Attack && TargetType_g1 == ThreateningAirTarget ) {HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t3 == HeadingToOthers && Mission_g1_t4 == Attack && TargetType_g1 == ThreateningGroundTarget ) {HeadingToCI : .001150747986191; HeadingToOthers : .9470655926352128; Staying : .0517836593785961; }
else if( Activity_g1_t3 == HeadingToCI && Mission_g1_t4 == Others && TargetType_g1 == ThreateningAirTarget ) {HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t3 == HeadingToCI && Mission_g1_t4 == Attack && TargetType_g1 == ThreateningAirTarget ) {HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
else if( Activity_g1_t3 == HeadingToCI && Mission_g1_t4 == Attack && TargetType_g1 == ThreateningGroundTarget ) {HeadingToCI : .3333333333333333; HeadingToOthers : .3333333333333333; Staying : .3333333333333333; }
}
defineNode(RegionType_g1_t4 , Desc);
{ defineState(Discrete, OffRoad, road );
p( RegionType_g1_t4 | RegionType_g1_t3 ) =
if( RegionType_g1_t3 == OffRoad ) {OffRoad : .9994304214880988; road : .0005695785119012; }
else if( RegionType_g1_t3 == road ) {OffRoad : .1363636363636364; road : .8636363636363636; }
}
defineNode(Latitude_Velocity_g1_t4 , Desc);
{ defineState(Continuous);
p( Latitude_Velocity_g1_t4 | TargetType_g1 , Activity_g1_t4 , RegionType_g1_t4 , Latitude_Velocity_g1_t3 ) =
if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToCI && TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.0000093750); }
else if( RegionType_g1_t4 == road && Activity_g1_t4 == Staying && TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.0000093750); }
else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == ThreateningAirTarget ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.001261594092696); }
else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == Staying && TargetType_g1 == ThreateningAirTarget ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.0000093750); }
else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToCI && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.0000093750); }
else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToCI && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.000077584390404); }
else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.0000093750);
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToCI && TargetType_g1 == ThreateningAirTarget ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == Staying && TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.0000093750);
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToCI && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == ThreateningAirTarget ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == Staying && TargetType_g1 == Others ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.0000093750);
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Latitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToCI && TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == Staying && TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.0000093750);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToCI && TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == Staying && TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToCI && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToCI && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , .0000093750);
NormalDist( 0.0 , 0.0001621872);
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == Staying &&
TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t3 +
NormalDist( 0.0 , 0.011313612741177);
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToCI &&
TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 ,
0.0001621872);
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToCI &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t3 +
NormalDist( 0.0 , 0.000137107925352);
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToOthers &&
TargetType_g1 == ThreateningGroundTarget ) {1.0 * Longitude_Velocity_g1_t3 +
NormalDist( 0.0 , 0.000009086095747);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == Staying &&
TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 ,
0.0001621872);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers &&
TargetType_g1 == ThreateningAirTarget ) {1.0 * Longitude_Velocity_g1_t3 +
NormalDist( 0.0 , 0.0000069813111472);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers &&
TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.0022134952244327);
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers &&
TargetType_g1 == Others ) {1.0 * Longitude_Velocity_g1_t3 + NormalDist( 0.0 ,
0.00193765071369);
}
defineNode(Altitude_Velocity_g1_t4 , Desc);
{ defineState(Continuous);
  p( Altitude_Velocity_g1_t4 | TargetType_g1 , Activity_g1_t4 , RegionType_g1_t4 ,
Altitude_Velocity_g1_t3 ) =
    if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToCI && TargetType_g1
== Others ) {1.0 * Altitude_Velocity_g1_t3 +
      NormalDist( 0.0 , 0.000069813111472);
    } else if( RegionType_g1_t4 == road && Activity_g1_t4 == Staying &&
      TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t3 + NormalDist( 0.0 ,
      0.000082176);
    } else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToOthers &&
      TargetType_g1 == ThreateningAirTarget ) {1.0 * Altitude_Velocity_g1_t3 +
      NormalDist( 0.0 , 0.0000851958686801);
    } else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToCI &&
      TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t3 +
      NormalDist( 0.0 , 0.0003082176 );
    } else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == Staying &&
      TargetType_g1 == ThreateningAirTarget ) {1.0 * Altitude_Velocity_g1_t3 +
      NormalDist( 0.0 , 0.00000882176 );
    } else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == Staying &&
      TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t3 + NormalDist( 0.0 ,
      0.00000882176);
    } else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToCI &&
      TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t3 +
      NormalDist( 0.0 , 0.000086826898311);
    } else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToCI &&
      TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t3 +
      NormalDist( 0.0 , 0.000021225194323);
    } else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers &&
      TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t3 +
      NormalDist( 0.0 , 0.000082176 );

298
```plaintext
else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToCI && TargetType_g1 == ThreateningAirTarget ) {1.0 * Altitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.000085170791182 );
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == Staying && TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.000083114682263 );
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToOther && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.000079108212487 );
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == Staying && TargetType_g1 == ThreateningGroundTarget ) {1.0 * Altitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.000070498895006 );
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == ThreateningAirTarget ) {1.0 * Altitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.000084681077152 );
} else if( RegionType_g1_t4 == road && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.000085114682263 );
} else if( RegionType_g1_t4 == OffRoad && Activity_g1_t4 == HeadingToOthers && TargetType_g1 == Others ) {1.0 * Altitude_Velocity_g1_t3 + NormalDist( 0.0 , 0.000079108212487 );
}
}
defineNode(Latitude_g1_t4 , Desc);
{ defineState(Continuous);
p( Latitude_g1_t4 | Latitude_Velocity_g1_t3 , Latitude_g1_t3 ) =
1.0 * Latitude_Velocity_g1_t3 + 1.0 * Latitude_g1_t3 + NormalDist( 0.0 , 0.0000001 );
}
defineNode(Longitude_g1_t4 , Desc);
{ defineState(Continuous);
p( Longitude_g1_t4 | Longitude_Velocity_g1_t3 , Longitude_g1_t3 ) =
1.0 * Longitude_Velocity_g1_t3 + 1.0 * Longitude_g1_t3 + NormalDist( 0.0 , 0.0000001 );
}
defineNode(Altitude_g1_t4 , Desc);
{ defineState(Continuous);
p( Altitude_g1_t4 | Altitude_Velocity_g1_t3 , Altitude_g1_t3 ) =
1.0 * Altitude_Velocity_g1_t3 + 1.0 * Altitude_g1_t3 + NormalDist( 0.0 , 0.0000001 );
}
defineNode(DistanceToCI_g1_t4 , Desc);
{ defineState(Continuous);
p( DistanceToCI_g1_t4 | Activity_g1_t4 , DistanceToCI_g1_t3 ) =
if( Activity_g1_t4 == HeadingToOthers ) {1.002575422880476 * DistanceToCI_g1_t3 + NormalDist( 0.0 , 290571.93849371106 );
} else if( Activity_g1_t4 == HeadingToCI ) {0.991535242451743 * DistanceToCI_g1_t3 + NormalDist( 0.0 , 292618.244720563 );
} else if( Activity_g1_t4 == Staying ) {1.0 * DistanceToCI_g1_t3 + NormalDist( 0.0 , 161584.4553120503 );
}
}
defineNode(DirectionToCI_g1_t4 , Desc);
{ defineState(Continuous);
p( DirectionToCI_g1_t4 | Activity_g1_t4 , DirectionToCI_g1_t3 ) =
if( Activity_g1_t4 == HeadingToOthers ) {0.981336501694075 * DirectionToCI_g1_t3 + NormalDist( 0.0 , 290571.93849371106 );
} else if( Activity_g1_t4 == HeadingToCI ) {0.130346396994075 * DirectionToCI_g1_t3 + NormalDist( 0.0 , 292618.244720563 );
} else if( Activity_g1_t4 == Staying ) {1.0 * DirectionToCI_g1_t3 + NormalDist( 0.0 , 0.0000001 );
}
}
```
defineNode(Temperature_g1_t4 , Desc);
{ defineState(Continuous);
p( Temperature_g1_t4 | Activity_g1_t4 , Temperature_g1_t3 ) =
if( Activity_g1_t4 == HeadingToOthers ) {1.000008891419039 * Temperature_g1_t3
+ NormalDist( 0.0 , 145.5634034218463);}
else if( Activity_g1_t4 == HeadingToCI ) {1.0000034017283523 *
Temperature_g1_t3 + NormalDist( 0.0 , 91.81289968886823);}
else if( Activity_g1_t4 == Staying ) {1.0000068754977902 * Temperature_g1_t3 +
NormalDist( 0.0 , 21.75286278479968);}}
Appendix L. Use Case 2: PROGNOS PO via UMP-ST

To develop the original PROGNOS PO [Carvalho et al., 2011][Costa et al., 2012], three iterations of the four steps in UMP-ST (*Requirement*, *Analysis* & *Design*, *Implementation*, and *Test*) were performed. The following subsections summarize the four steps in UMP-ST to develop the PROGNOS PO.

**L.1 Requirements**

The *Requirement* step identifies requirements containing goals, queries, and evidence for a probabilistic ontology. The requirements for the PROGNOS PO were developed gradually over the three iterations. In the first iteration, a simple requirement regarding a ship of interest was identified [Carvalho et al., 2010]. In the second iteration, requirements for two types of terrorist-ships were defined. In the third iteration, requirements for crew members in a ship of interest were specified. The following list shows part of the resulting requirements [Carvalho, 2011].

1. Identify if a ship is of interest,
   1.1 Is the ship being used to exchange illicit cargo?
   1.1.1 Was the ship hijacked?
   1.1.2 Does the ship have a terrorist crew member?
   1.1.2.1 Is the crew member associated with any terrorist organization?
   ...
   1.2 Is the ship being used as a suicide ship to bomb a port?
   ...

The main goal was to identify a ship of interest (i.e., a terrorist-ship). In this requirement, we assumed the ship of interest may exchange illicit cargo and/or be used as
a suicide ship to bomb a port. To support this goal, we needed to identify the type of a crew member of a ship. If the type of a crew member is a terrorist, the ship is highly likely to be a terrorist-ship. To identify whether a crew member is a terrorist, we can check whether the crew member is associated with any terrorist organization.

L.2 Analysis & Design

This step defines the types of entities, their properties and relationships, and the rules that apply to them, i.e., the semantics of the domain model. The Unified Modeling Language (UML) diagrams can provide a convenient and understandable visualization of the classes and relationships for the model semantics. The requirements defined in the previous step are used to develop the model semantics. Thus, entities, attributes for the entities, and relationships between the entities were identified. For example, from Requirement 1, an entity was derived (i.e., a ship) and an attribute of the entity was derived (i.e., the type of a ship). From Requirement 1.1.2, a new entity was derived (i.e., a (terrorist) person) and a relationship between the entities was derived (i.e., a ship has a crew (terrorist) member). In the second iteration, Carvalho [Carvalho, 2011] developed the model represented by UML as shown in Fig. L.1.

The classes and relationships form six natural groups (i.e., Electronics, Behavior, Ship, Position, Plan, and Social Network). The ship types are NavyShip, FishingShip, and MerchantShip. Ship routes are UnusualRoute and UsualRoute. Two ships can meet each other at a position. A ship can use electronic devices such as Radio, Radar, and AIS (Automatic Identification System). A ship can show behavior such as Aggressive, Erratic, Evasive, and Normal. A ship can have a (terrorist) crewmember who may belong to a
(terrorist) organization. A ship can have a terrorist plan such as *BombPort* and *ExchangeIllicitCargo*.

Figure L.1 Entities, their attributes, and relations for the MDA model after the second iteration (This figure provided by permission of Carvalho [2011]).

After developing the model semantics, conditional rules were identified. There were three iterations of this process. The following list shows a few of the conditional rules from [Carvalho, 2011].

1. **(a)** If a crew member is a member of a terrorist organization, then it is more likely that he is a terrorist.

1. **(b)** If an organization has a terrorist member, it is more likely that it is a terrorist organization.

...  

4. **(a)** Research shows that if a crew member has a relationship with terrorists, there is a 68% chance that he has a friend who is a terrorist.

...
These conditional rules were derived from extensive research about terrorism [Sageman, 2004] and from the knowledge provided by a domain expert. These rules were used to develop the PROGNOS PO.

L.3 Implementation

In the Implementation step, the PROGNOS PO was designed. The PROGNOS PO can be found in [Carvalho, 2011][Costa et al., 2012]. Fig. L.2 shows the PROGNOS PO containing five groups of MFrags.

![Original PROGNOS probabilistic ontology](image)

**Figure L.2 Original PROGNOS probabilistic ontology**

The first set of MFrags is for a ship of interest. It includes nine MFrags: *Aggressive Behavior, Terrorist Plan, Evasive Behavior, Erratic Behavior, Unusual Route, Bomb Port Plan, Ship Of Interest, Electronics Status,* and *Exchange Illicit Cargo Plan.* These MFrags are used to reason about properties of a ship (e.g., unusual behavior and an illegal plan). The second set of MFrags is for a person of interest. It includes four MFrags:
Person Communications, Person Background Influences, Person Cluster Associations, and Person Relations. These MFrags are used to identify a person who may communicate with a terrorist, has a suspicious background and history, and has a relationship with a terrorist. The third set of MFrags is for information of relationships between two ships. It includes two MFrags, Radar and Meeting. These MFrags are used to identify whether one ship is within radar range of another ship and whether two ships are meeting. The fourth set of MFrags is for information about the relationship between a person and an organization. It includes one M_frag Terrorist Person in which a person who belongs to an organization is identified. The last set of MFrags is for information about a relationship between a person and a ship. It includes two MFrags Has Terrorist Crew and Ship Characteristics. These MFrags are used to link a person and a ship, and to identify whether a ship has a terrorist crew member.

The following list shows part of a partial PROGNOS PO containing information about MFrags (F), context nodes (C), resident nodes (R), resident parent nodes (RP), and input parent nodes (IP). Note that a partial probabilistic ontology doesn't contain a class local distribution and domain information for a random variable.

**PO L.1:** Original PROGNOS probabilistic ontology

```plaintext
1  [F: ErraticBehavior_MFrag
2    [C: isA(ship,Ship)]
3    [R: hasErraticBehavior(ship) [IP: hasExchangeIllicitCargoPartition(ship)]]
4    [R: hasEquipmentFailure(ship)]
5    [R: isCrewVisible(ship) [RP: hasErraticBehavior(ship)] [RP: hasEquipmentFailure(ship)]]
6  ]
7  [F: TerroristPerson_MFrag
8    [C: isA(person,Person), isA(org,Organization)]
9    [R: isTerroristOrganization(org) [IP: isTerroristPerson(person)] [RP: isMemberOfOrganization(person, org)]]
10   [R: isTerroristPerson(person) [RP: isMemberOfOrganization(person, org)]]
11  ]
12  [F: ShipCharacteristics_MFrag
```
[C: isA(ship,Ship), isA(person,Person)]

[R: hasCrewMember(ship, person)]

[R: hasTypeOfShip(ship)][R: isHijacked(ship)]

[F: EvasiveBehavior_MFrag
[C: isA(ship,Ship)]

[R: hasEvasiveBehavior(ship)][IP: hasExchangeIllicitCargoPartition(ship)]

] [F: PersonCommunications_MFrag
[C: isA(person,Person)]

[R: communicatesWithTerrorist(person)][IP: isTerroristPerson(person)]

[R: usesChatroom(person) [RP: communicatesWithTerrorist(person)]]

[R: usesEmail(person) [RP: communicatesWithTerrorist(person)]]

[R: usesCellular(person) [RP: communicatesWithTerrorist(person)]]

[R: usesWeblog(person) [RP: communicatesWithTerrorist(person)]]

] [F: PersonBackgroundInfluences_MFrag
[C: isA(person,Person)]

[R: hasInfluencePartition(person) [IP: isTerroristPerson(person)]]

[R: knowsPersonImprisonedInOIForOEF(person) RP: hasOIForOEFInfluence(person)]]

[R: hasFamilyStatus(person) [RP: hasInfluencePartition(person)]]

[R: hasOIForOEFInfluence(person) [RP: hasInfluencePartition(person)]]

[R: knowsPersonKilledInOIForOEF(person) RP: hasOIForOEFInfluence(person)]]

] [F: AggressiveBehavior_MFrag
[C: isA(ship,Ship)]

[R: hasAggressiveBehavior(ship) [IP: hasBombPortPlan(ship)][IP: hasExchangeIllicitCargoPartition(ship)]]

[R: hasWeaponVisible(ship) [RP: hasAggressiveBehavior(ship)]]

[R: isJettisoningCargo(ship) [RP: hasAggressiveBehavior(ship)]]

[R: speedChange(ship) [RP: hasAggressiveBehavior(ship)]]

[R: propellerTurnCount(ship) [RP: speedChange(ship)]]

[R: cavitation(ship) [RP: speedChange(ship)][RP: turnRate(ship)]]

[R: shipRCSchange(ship) [RP: turnRate(ship)]]

] [F: ShipOfInterest_MFrag
[C: isA(ship,Ship)]

[R: isShipOfInterest(ship) [IP: hasTerroristPlan(ship)]]

] [F: ExchangeIllicitCargoPlan_MFrag
[C: isA(ship,Ship)]

[R: hasExchangeIllicitCargoPlan(ship) [IP: hasTerroristPlan(ship)]]

[R: hasExchangeIllicitCargoPartition(ship)]

[IP: hasTypeOfShip(ship)][RP: hasExchangeIllicitCargoPlan(ship)]]

] [F: PersonRelations_MFrag
[C: isA(person,Person)]

[R: hasKinshipToTerrorist(person) [RP: hasTerroristBeliefs(person)]]

[R: hasFriendshipWithTerrorist(person) [RP: hasTerroristBeliefs(person)]]

[R: hasTerroristBeliefs(person) [IP: isTerroristPerson(person)]]

] [F: Meeting_MFrag
[C: isA(ship1,Ship), isA(ship2,Ship)]

[C: (¬(ship1 = ship2) )]

[R: areMeeting(ship1, ship2) [IP: hasExchangeIllicitCargoPartition(ship1)]]

[R: areMeetingReport(ship1, ship2) [RP: areMeeting(ship1, ship2)]]

] [F: BombPortPlan_MFrag
[C: isA(ship,Ship)]

[R: hasBombPortPlan(ship) [IP: hasTerroristPlan(ship)]]

]
PO L.1 shows the context nodes and the resident nodes in the MFrags, and the causal relationship between the resident nodes. For example, the MFragment *ErraticBehavior_MFrag* (Line 1–6) contains an *isA* context node and three resident nodes *hasErraticBehavior*, *hasEquipmentFailure*, and *isCrewVisible*. The resident node *hasErraticBehavior* is influenced by an input node *hasExchangeIllicitCargoPartition*.
The resident node *isCrewVisible* is influenced by the resident nodes *hasErraticBehavior* and *hasEquipmentFailure*. This PROGNOS PO was tested in the next step.

**L.4 Test**

In this step, the PROGNOS PO was evaluated to determine whether to accept it. To do this, the case-based evaluation, in which various scenarios were defined and used to examine the reasoning implications of the probabilistic ontology, was used. For example, given a scenario which was developed by a subject matter expert (SME), some information (e.g., history of a target) from the scenario for a target was used as evidence for inference of the PROGNOS PO to identify some properties (e.g., whether the target is a terrorist) of the target. If the result of inference coincided exactly with the scenario from SME, we could think that the probabilistic ontology was reasonable. For this test, three qualitatively different scenarios were used [Carvalho, 2011].

After three iterations for UMP-ST, an overall test for the PROGNOS PO was performed using a simulation. In the real world situation, it is very difficult to acquire a real dataset to develop such a probabilistic ontology which contains rare events. For this reason, the simulation was used to produce a test dataset given different scenarios generated randomly. Carvalho [2011] and Costa et al [2012] introduced some results for this test. In such a test, it is important that knowledge used to develop a probabilistic ontology and knowledge used to develop a simulation for testing the probabilistic ontology should not be same. If they are same, the test is meaningless, because the probabilistic ontology and the simulation are same models, but just in different forms.
Appendix M. Use Case 3: Predictive Situation Awareness Model for Smart Manufacturing

M.1 Smart Manufacturing

In this section, we begin with a brief description of smart manufacturing and introduce factors for smart manufacturing.

M.1.1 General Manufacturing

Manufacturing is the process of transforming (raw) materials and energy, by means of workers and machinery, into products that address manufacturing requirements from stakeholders. Fig. M.1 shows a general manufacturing function represented as IDEF0\(^16\).

The general manufacturing function begins with manufacturing requirements from stakeholders. Such requirements are realized by producing the products which the

\(\text{IDEF is an acronym for “Icam DEFinition for Function Modeling”}.\) IDEF0 contains a box representing a function and four classes of arrow (Input Arrow, Output Arrow, Control Arrow, and Mechanism Arrow) denoting data, control, and functional flows.
stakeholders need and/or want. To produce the products, the general manufacturing function requires various inputs (e.g., raw materials, components, and other resources). Also, it requires some enablers (e.g., man, machine, method, technology, and environment) to achieve its purpose. Commonly, the examples of a measure of effectiveness (MOE) for general manufacturing include product quality, manufacturing time, and manufacturing cost.

**M.1.2 Smart Manufacturing**

Various manufacturing paradigms were discussed in [Lu et al., 2016]. One of them is smart manufacturing, which has the following characteristics: (1) Digitization of manufacturing enterprises, (2) Connected devices and distributed intelligence, (3) Collaborative supply chain, (4) Integrated and optimal decision making, and (5) Sensors and big data analytics [Lu et al., 2016]. For this research, we consider smart manufacturing to include autonomous functions associated with OODA loop [Boyd, 1976][Boyd, 1987] to respond proactively, responsively, and adaptively to market requirements. OODA is a high-level concept for decision making, which comprises four steps (Observe, Orient, Decide, and Act). In the Observe step, data and/or signals from external and internal systems are observed. In the Orient step, observations become information and are used to form a model for plans and COAs (Courses Of Action). In the Decide step, hypotheses or alternatives for models are decided according to the preference of a decision maker. In the Act step, the chosen COAs are implemented.

Smart manufacturing performs OODA with little or no human interference. Vagia et al. [2016] presented 8 autonomy levels (ranging from Level 1: No computer assistance
to Level 8: Operations by computers for everything without human interference). The aim of smart manufacturing is to approach Level 8. Therefore, smart manufacturing requires the following: (1) Autonomous Observe: autonomous data gathering by sensors, (2) Autonomous Orient: autonomous information and knowledge construction by AI systems, (3) Autonomous Decide: autonomous decision making by cooperation of AI systems and human, and (4) Autonomous Act: autonomous manufacturing operation by actuator and equipment. Fig. M.2 extends Fig. M.1 to a smart manufacturing function.

![Figure M.2 Smart Manufacturing](image)

For smart manufacturing, an autonomous OODA function is required to perform OODA. The autonomous OODA observes the overall situations for the manufacturing function /system and decides action plans for manufacturing by using IoT, AI, and CPS technologies.
M.1.3 Predictive Manufacturing Situation Awareness Factors for Smart Manufacturing

To perform smart manufacturing, Predictive Manufacturing Situation Awareness (MSAW) is required to estimate and predict situations. MSAW provides a big picture for an overall manufacturing situation. With such a big picture, decision makers (human and/or machine) in manufacturing can make better decisions. Therefore, for better smart manufacturing, precise, comprehensive, and efficient MSAW is required.

We consider factors which must be represented to provide MSAW. These MSAW factors for smart manufacturing include:

- Mission/Goal
- System/Function
- Input/Output
- Time

These MSAW factors include missions for systems to perform functions at a given time to use inputs in order to produce outputs.

M.2 Representation for Uncertainty in Manufacturing

Supporting autonomous MSAW requires a computable representation of the MSAW factors described above. In this section, we introduce an MSAW-MEBN model that does this.

M.2.1 Basic Uncertainty Model for Manufacturing Process

As a first step toward developing the MSAW-MEBN model, we begin by considering a basic manufacturing model.
A system that performs processes in manufacturing produces outputs (e.g., products and materials) by receiving inputs (e.g., energy and resources). Therefore, the basic unit of manufacturing can be defined as systems, inputs, and outputs. The inputs and outputs, commonly called items, are defined as entities with similar properties. We consider the systems and items (i.e., inputs and outputs) to be the basic elements of manufacturing as shown in Fig. M.3.

![Figure M.3 Basic Model for Manufacturing Process](image)

Each system has its own properties (or attributes), and each item also has properties. Examples of system properties include operating time and production cost. An example of an item property is quality.

Next, consider an uncertainty model corresponding to this model. The uncertainty model contains random variables (RVs) which represent uncertain properties, as well as causal relationships among these RVs. Fig. M.4 below shows the system properties and the item properties represented as RVs.
System properties and item properties can be influenced by each other. Item properties can affect other item properties. For example, in a steel plate manufacturing factory, a heater is used to preheat slabs before rolling. Heating cost and heating time are properties of the heater (i.e., System). Slab temperature and slab size are properties of items. The working time of the heater may depend on the size of the slab. Thus, the input item property (i.e., the slab size) influences the system property (i.e., the heating time). In addition, the slab temperature before heating affects the slab temperature after heating. Thus, the input item property influences the output item property. These causal relationships can be used to make decisions about specific operations.

M.2.2 MEBN Model for Basic Manufacturing Process

Uncertain situations in manufacturing can be represented through MEBN. The basic uncertainty model for the manufacturing process in Fig. M.4 does not contain entity information (e.g., a system entity, an item entity, or a time entity). Such entity information can be used to provide clear information for what entity is associated with the RVs. That is, we can use the basic MSAW-MEBN model to represent situations with multiple items and/or multiple systems. This basic MSAW-MEBN model can be applied
to support MSAW. Fig. M.5 shows the basic MSAW-MEBN model containing three MFrags: (1) the Mfrag \textit{System}, (2) the Mfrag \textit{Input}, and (3) the Mfrag \textit{Output}. The MEBN model also contains three kinds of entities: \textit{System}, \textit{Item}, and \textit{Time}. The \textit{System} entity is an entity denoting a system performing processes or functions. The \textit{Item} entity is an entity denoting an input to a system and an output from a system. The \textit{Time} entity is an entity denoting a time stamp indicating a time interval for system operation.

(1) The Mfrag \textit{System} represents probabilistic knowledge of properties of systems. It contains five \textit{isA} context nodes for entities \textit{Sending System}, \textit{Sending Time}, \textit{Sending Item}, \textit{Receiving System}, and \textit{Receiving Time}. The entities \textit{Sending System}, \textit{Sending Time}, and \textit{Sending Item} identify what system is sending what item at what time. The entities \textit{Receiving System}, and \textit{Receiving Time} identify what system is receiving an item at what time. The Mfrag contains a context node \textit{Input_to_System} which is used to guarantee valid entities for the above five entities and the distributions for RVs in the Mfrag. In the Mfrag, an input node \textit{Item_Property}, defined in the Mfrag \textit{Output}, influences a resident node \textit{System_Property}. This is the same as the item property affects the system property in Fig. M.4. The resident node \textit{System_Property} (\textit{receivingSys}, \textit{receivingTime}) has a receiving system ordinary variable \textit{receivingSys} and a receiving time ordinary variable \textit{receivingTime} as arguments. It indicates the system properties for a receiving system at a receiving time.
Figure M.5 Basic MSAW-MEBN Model for Manufacturing Process

(2) The MFrags Input represents probabilistic knowledge of relationships between items and systems. It contains six isA context nodes for entities Sending System, Sending Time, Sending Item, Receiving System, Receiving Time, and Receiving Item. The MFrags contain two resident nodes Input_to_Output and Input_to_System. The resident node Input_to_Output is used to represent a condition under which the relationship between an input and an output item is valid. The resident node Input_to_System is used to represent a condition under which the relationship between an input item and a system is valid.
(3) The MFrag Output represents probabilistic knowledge of properties of items. It contains two input nodes Item\_Property and System\_Property, and one resident node Item\_Property. The input node Item\_Property is the property of an input item, while the resident node Item\_Property is the property of an output item. The resident node Item\_Property is affected by the input node System\_Property. This is the same as the item property and system property affects the item property in Fig. M.4.

The above basic MSAW-MEBN model can be used to construct various situation-specific Bayesian Networks (SSBNs) for supporting MSAW. In the following, we introduce an illustrative simple example for MSAW. For the example, suppose there is a simple item flow between two systems as shown Fig. M.6 below. Item0 produced by an external system is input into System1. System1 produces Item1 as an output. Item1 is an input for System1 and/or System2. System2 produces Item2 using Item1. System1 will produce Item1, when it receives Item0 or (recursively) Item1, generated by System1.

An SSBN constructed from the basic MSAW-MEBN model is shown below in Fig. M.7. The SSBN represents the situation depicted in the model of Fig. M.6. The upper box shows the first flow denoted by Flow1 and the lower box shows the second flow denoted by Flow2. In the first flow, there is an item property
(Item\_Property\_i0\_s0\_t1) of Item0 (i0) produced by an external System0 (s0), which is out of the overall system boundary, at Time1 (t1). The item property of Item0 influences an item property (Item\_Property\_i1\_s1\_t1) of Item1 (i1) produced by System1 (s1) at Time1 (t1). In the similar way, the item property Item\_Property\_i1\_s1\_t1 influences an item property Item\_Property\_i2\_s2\_t1 in Flow1. In some situations, Item1 can affect System1 to change a system property as shown Fig. M.7. Flow2 shows such a situation where the item property Item\_Property\_i1\_s1\_t1 in Flow1 influences a system property System\_Property\_s1\_t2 in Flow2. Then a flow similar to Flow1 also occurs in Flow2.

![Figure M.7 SSBN Representing a Two-Flows Situation](image)

The above SSBN in Fig. M.7 was constructed from the basic MSAW-MEBN model given entity information (i.e., i0, i1, i2, s0, s1, s2, t1, and t2) and evidence for the relationships between entities as shown the following list.

**For Flow1**

\[ Input\_to\_Output(i0, s0, t1, i1, s1, t1) = true \]
Input_to_Output(i1, s1, t1, i2, s2, t1) = true

For Flow2

Input_to_Output(i0, s0, t2, i1, s1, t2) = true
Input_to_Output(i1, s1, t2, i2, s2, t2) = true
Input_to_System(i1, s1, s1, s1, t2) = true

The above SSBN is just one example of the use of the MSAW-MEBN model to represent a manufacturing situation. We can use the basic MSAW-MEBN model in Fig. M.5 to represent and reason about various manufacturing situations, each particular situation being represented by a different SSBN.

M.3 Use Case

In this section, we present an illustrative use case for the MSAW-MEBN model supporting MSAW for a steel plate manufacturing situation.

In this illustrative use case, the MSAW-MEBN model for steel plate manufacturing is used to predict the production time, production cost, and product quality for steel plates. The steel plate manufacturing system contains various equipment (e.g., a reheating furnace, a roughing mill, and a finishing mill). A slab is taken as an input material and it is pressed by the roughing mill and the finishing mill. Then as a final product, a steel plate is produced. The goal of the steel plate manufacturing system is to produce products of good quality (e.g., few defects and required flatness) in a short time and at low cost. To address such goals, we should be aware of the manufacturing...
situations first. The MSAW-MEBN can be used to support such manufacturing situation awareness.

Fig. M.8 shows an uncertainty model for the steel plate manufacturing system. The uncertainty model below was developed by using the MSAW-MEBN model in Section 6.3.2. Properties of the steel plate are taken from a Steel Plates Faults Data Set [Steel Plates Faults Data Set, 2017]. The data set contains 7 fault types for a steel plate: Pastry, Z_Scratch, K_Scratch, Stains, Dirtiness, Bumps and Other_Faults. It contains 27 independent variables (e.g., X_Minimum and X_Maximum) as shown in Fig. M.8.

![Figure M.8 Uncertainty Model for Steel Plate Manufacturing System](image)

The above model is divided into two groups: A situation group and a target object group. The target object group represents the system and the items in the factory, and the
situation group represents situation variables which are used for decision making. The above model describes the system from a top-level viewpoint. That is, in the target object group, there is one overall system, one input item, and one output item. The system and items in the target object group have various properties. For example, the steel type, the slab size, and the slab temperature are properties of the input item. The system has a cost property, a time property, and a system process type property. The various variables of a steel plate (e.g., size, thickness, and luminosity) are properties of the output item. Also, various steel plate fault variables (e.g., Pastry, Stains, Dirtiness, and Bumps [Steel Plates Faults Data Set, 2017]) are properties of the output item. Note that for the above model, we omitted the observing condition group and the report group in Fig. 6.6 in Section 6.3.2. Other relevant datasets or information for such groups could also be added to Fig. M.8.

In the situation group, there are state properties representing total cost, total time, and total quality rate. The total cost and total time are the total system operating cost and the total time respectively, after all steel plates have been produced. The total quality rate [Blanchard et al., 1990] is the overall quality rate value for all steel plates manufactured. The total quality rate is defined as the following.

\[
\text{Total Quality Rate} = 1 - \frac{\sum_{i \in I} \sum_{f \in F} fault_{i,f}}{|I||F|},
\]

where \(I\) denotes all output items, \(|S|\) denotes the number of the elements in a set \(S\), \(F\) denotes all fault types, and \(fault_{i,f}\) means an \(f\)-th fault value (zero denotes no-fault and one indicates that a fault has occurred) for the \(i\)-th output item.
The goal of inference from the MSAW-MEBN model for steel plate manufacturing is to predict the total cost, total time, and total quality rate of all manufactured products given the input item properties (e.g., the steel type, the slab size, and the slab temperature). To do this, we first need to create an MSAW-MEBN model. We used MEBN learning (Chapter 5) to create the MSAW-MEBN model using a training dataset. We used a holdout sample to test how well the MSAW-MEBN model predicts cases outside the training set. Such datasets for the MSAW-MEBN model were modified from the Steel Plates Faults Data Set. The dataset from only had data associated with the properties of the manufactured steel plate (i.e., the output item properties in Fig. M.8). However, there was no input item dataset (e.g., type of steel, slab thickness, and slab temperature) and no system dataset (e.g., production cost, production time, and process type). Such datasets are used to test how the MSAW-MEBN model (the contribution of this research) performs in the steel plate manufacturing situation. In order to generate such datasets, models for the input item and the system were created by expert knowledge. Note that expert knowledge can be directly used to create some of the local distributions of RVs in the MSAW-MEBN model instead of generating the datasets via the models from expert knowledge. The reasons why we use the datasets derived from the models are:

(1) We tried to make the test case as similar to reality as possible. Thus, in most cases for actual manufacturing situations, sensor data from the input item, output item, and system are given to perform data analysis (i.e., analysis using data, but not model);
(2) In this use case scenario, the goal of evaluation is to predict the production time, production cost, and product quality given input item values, so we needed test cases for input values as well as the time and cost values. The datasets were generated according to the dataset from [Steel Plates Faults Data Set, 2017]. The generated datasets were divided into two datasets: a training dataset and a test dataset. The training dataset was used for MEBN learning (Chapter 5) and the test dataset was used to evaluate predictions for a learned MSAW-MEBN model. For MEBN learning, the structure (i.e., the conditional relationships) in Fig. M.8 was used. For example, the RV $X_{Minimum}$ in the output item was influenced by the RVs in the input item (e.g., $Type_{of}_{Steel}$, $Slab_X$, and $Slab_Temperature$) and RVs in the system ($Cost$, $Time$, and $Type_{of}_{Process}$). Also, the RV $X_{Minimum}$ was influenced by the RVs in the fault properties in the output item (e.g., $Pastry$, $Z_{Scratch}$, and $K_{Scatch}$). Given this learned structure, parameters of the distributions (e.g., the slab size, the system cost, the plate size, and total cost) were learned under the assumption that continuous RVs are linear conditional Gaussian [Sun & Chang, 2010].

To predict the steel plate manufacturing situation, we assumed the situation in which three plates would be produced over time. The learned MSAW-MEBN model was used to construct an SSBN containing RVs for one system and three steel plate products. Also, the SSBN contained RVs for the total cost, total time, and total quality rate in the situation group in Fig. M.8. Then, we tested prediction for the total cost, total time, and
total quality rate given the input item values (e.g., the steel type, the slab size, and the slab temperature in the target object group in Fig. M.8) by comparing the actual results (i.e., test data for the total cost, total time, and total quality rate) and the predicted results (i.e., predicted Gaussian probability distribution for the total cost, total time, and total quality rate). To measure the performance of the predictions, continuous ranked probability score (CRPS) (Appendix C) was used. A good continuous ranked probability score (i.e., yielding a score of zero for a better model) means a fitness score for the MSAW-MEBN model to the test dataset. For inference of a conditional hybrid Bayesian network (i.e., the SSBN), no discrete node may have a continuous parent node, a Hybrid Message Passing inference algorithm [Sun & Chang, 2010] was used.

<table>
<thead>
<tr>
<th></th>
<th>Total Cost</th>
<th>Total Time</th>
<th>Total Quality Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Continuous Ranked Probability Score</td>
<td>0.0193 (0.0535)</td>
<td>0.0152 (0.0436)</td>
<td>1.2088 (0.4415)</td>
</tr>
</tbody>
</table>

The average CRPSs for the total cost, total time, and total quality rate were 0.0193, 0.0152, and 1.2088, respectively (numbers in parentheses are standard deviations). Note that a comparison between other prediction algorithm and the BN inference algorithm which we used in this research is beyond the scope of this research. The purpose of this use case is to show how to use the MSAW-MEBN model that supports MSAW (Predictive Manufacturing Situation Awareness).
M.4 Conclusion

In this research, we introduced an MSAW-MEBN model, developed via the PSAW-MEBN reference model, to reason about complex and complicated situations in manufacturing. Also, we presented a use case for the MSAW-MEBN model supporting MSAW for a steel plate manufacturing process to predict the total cost, total time, and total quality rate. Such an MSAW-MEBN model can be used for self-awareness and self-prediction, which are required for smart manufacturing.

The following summarizes future research issues. (1) This research provided a simple use case; however, actual manufacturing situations are more complex. As future research, the MSAW-MEBN model will be applied to more complex real-world manufacturing situations. (2) Prediction accuracy is important for better decision in manufacturing. In the use case, one inference algorithm [Sun & Chang, 2010] was tested. In the future, various inference algorithms will be tested in terms of prediction accuracy. (3) Situation information estimated and predicted using the MSAW-MEBN model can be used to perform self-maintenance and self-reconfiguration [Renzi et al., 2014]. The development of a system performing such self-processes using the MSAW-MEBN model continues to be a challenging research topic.
Appendix N. Experimental Comparison of HMLP with UMP-ST

N.1 Slides for the Lectures

The following shows the lecture slides for Task 1. Obtain relevant knowledge (BN, MEBN, the script form of MEBN, and UMP-ST) in Section 6.4.1.
In real-world situations, BN should deal with:
- Repeated same RVs
- Unknown number of RVs
- RVs interacting with each other in varied ways

BN is not expressive enough to handle:
- Different numbers of entities for different situations
- Uncertainty about relationship between entities
- Situation evolving in time

Therefore, we need a more expressive representation.
We introduce a script representing an MTheory

```
[VehicleSpeedReportM disaggregate]
[Vehicle (obj) Vehicle]
[Speed (obj)]
[L: Speed + NormalDist(0, 2)]
[
]
[Speed (obj)]
[L: NormalDist(0, 2)]
[
]
```

CLD Functions

- **Sum**
  - E.g., `Sum(X)`
  - Given `x1, x2, and x3, x1 + x2 + x3`

- **Average**
  - E.g., `Average(X)`
  - Given `x1, x2, and x3, (x1 + x2 + x3) / 3`
The following shows the lecture slides for Task 4. Obtain HMLP knowledge.

HMLP

- Manual MEBN modeling by a domain expert is a labor-intensive and insufficiently agile process.
- Therefore, it is necessary to implement MEBN modeling automation using a machine learning method.

What is MEBN learning?

What are the steps in developing a MEBN model?

1. Analyze Requirements
2. Design World Model & Rules
3. Construct Reasoning Model
4. Test Reasoning Model

MEBN, an extension of UPP-RT (Cavalcante et al., 2016), is a process to develop a MEBN model by combining expert knowledge and a machine learning (i.e., data-driven approach) to develop a PSAW-MEBN model.

---

The following shows the slides for **Task 5. Obtain stakeholder requirements and domain knowledge**.
Cost prediction for a heater in a steel plate system

Goal
- Develop a total cost prediction system for the heater which heats a slab

Query
- Predict a total cost ($) to heat three slabs

Prior knowledge
- The heater system is associated with two infrared thermal imaging camera sensors (each error ~N(0, 3))
- The heater system contains an actuator which is used to control an energy value to heat a slab (i.e., the actuator calculates the energy value given the input slab temperature)
- There is no energy loss when the energy value is used in the heater
- All manufacturing factors (e.g., the temperature, energy value, and cost) are normally distributed continuous values
- Energy unit: kWh (kiloWatt-hour)
- Slab weight: 100kg (Fixed)
- Ordered temperature: 1200 °C (Fixed)
- Energy cost is 20cent/kWh

Situation

Database used in the system
N.2 Data given to Participants
The following excel files were given to the participants in both groups.

<table>
<thead>
<tr>
<th>time.csv</th>
<th>slabinput_item.csv</th>
<th>heateractuator_item.csv</th>
<th>heater_item.csv</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeID</td>
<td>TimeID temperature</td>
<td>TimeID energy</td>
<td>TimeID temperature</td>
</tr>
<tr>
<td>0</td>
<td>14.841084</td>
<td>0</td>
<td>1199.682</td>
</tr>
<tr>
<td>1</td>
<td>21.365812</td>
<td>1</td>
<td>1195.878</td>
</tr>
<tr>
<td>2</td>
<td>18.133339</td>
<td>2</td>
<td>1201.93</td>
</tr>
<tr>
<td>3</td>
<td>16.819091</td>
<td>3</td>
<td>1198.864</td>
</tr>
<tr>
<td>4</td>
<td>18.389012</td>
<td>4</td>
<td>1195.2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>99</td>
<td>15.404255</td>
<td>99</td>
<td>1197.926</td>
</tr>
</tbody>
</table>

Figure N.1 Excel files were given to both groups.
Also, same data sets as above which are stored in MySQL were given to the participants in both groups.

![MySQL schema for data sets which were given to both groups](image)

**N.3 MEBN Model Results from Participants**

The following six MTheories are the results from the participants.

---

**MTheory N.1**

1. [F: slabinput_itemM_frag]
2.  [C: Isa (t, Time)]
3.  [R: Temperature_beg (t)]
4.    [L: NormalDist(15, 9)]
5.  ]
6. ]
7. [F: heateractuator_itemM_frag]
8.  [C: Isa (t, Time)]
9.  [R: CalculateEnergy (t)]
10. [IP: Temperature_beg (t)]
11. [L: (1200 – Sum(Temperature_beg)) * 100 + NormalDist(0, 83765)]
12. ]
13. ]
14. [F: heater_itemM_frag]
15.  [C: Isa (t, Time)]
16.  [R: Temperature_end (t)]
17.  [L: NormalDist(1200, 7)]
MTheory N.2

[F: SlabInputMFragment]
[C: Isa(t, Time)]
[R: Temperature(t)]
[L: NormalDist(15.6, 6.9)]

[R: TemperatureSensor(t)]
[RP: Temperature(t)]
[L: Sum(Temperature) + NormalDist(0, 3)]

[F: HeaterActuatorMFragment]
[C: Isa(t, Time)]
[R: Energy(t)]
[IP: Temperature(t)]
[L: 120000 - (100*Sum(Temperature))+NormalDist(0, 0.000001)]

[F: HeaterMFragment]
[C: Isa(t, Time)]
[R: ChangedTemperature(t)]
[IP: Energy(t)]
[IP: Temperature(t)]
[L: 0.01* Sum(Energy)+Sum(Temperature)+ NormalDist(1199.5, 9.9)]

[F: TotalCostMFragment]
[C: Isa(tc, TotalCost), Isa(t, Time)]
[R: Cost(tc)]
MTheory N.3

1 [F: TemperatureReportM_frag]
2   [C: Isa (Obj, Slab)] [C: Isa (rpt, Report)] [C: Obj = ReportedObject(rpt)]
3   [R: TemperatureReport(rpt)
4     [RP: SlabTemperature (Obj)]
5     [L: NormalDist (0, 3) + Sum(SlabTemperature)]
6   ]
7   [R: SlabTemperature (Obj)
8     [L: NormalDist (15.55784, 6.89249)]
9   ]
10 ]
11 [F: EnergyReportM_frag]
12   [C: Isa (Obj, Slab), Isa (Eg, Energy)]
13   [R: TotalEnergyCost (Eg)
14     [RP: Energy (Eg)]
15     [L: Sum(Energy) * 0.20 + NormalDist(0, 0.000001)]
16   ]
17   [R: Energy (Eg)
18     [IP: SlabTemperature (Obj)]
19     [L: 99.8414 * (1200 – Sum(SlabTemperature)) + NormalDist(0, 0.000001)]
20   ]
21 ]
22 [F: ReferenceM_frag]
23   [C: Isa (rpt, Report)]
24   [R: ReportedObject(rpt)]
25 ]

MTheory N.4

1 [F: heater_item
2   [C: IsA(heater_item_TimeID, TIME)]
3   [R: HI_temperature(heater_item_TimeID)
4     [L: NormalDist(1200.201336434659, 6.192662550762393)]

337
MTheory N.5

```plaintext
[F: slabinput_item
[C: IsA(slabinput_item_TimeID, TIME)]
[R: SII_temperature(slabinput_item_TimeID)]
[L: NormalDist(15.557843606667822, 6.892498282240208)]
[R: SIIReported_temperature(slabinput_item_TimeID)]
[RP: SII_temperature(slabinput_item_TimeID)]
[L: Sum(SII_temperature) + NormalDist(0, 3)]
]

[F: HAI_energy_SII_temperature
[C: IsA(heateractuator_item_TimeID, TIME)]
[R: HAI_energy(heateractuator_item_TimeID)]
[IP: SII_temperature(heateractuator_item_TimeID)]
[L: -99.99999814540547 * Sum(SII_temperature) +
NormalDist(119999.99995954607, .0000000006984919)]
]

[F: TotalCost
[C: IsA(item_TimeID, TIME), IsA(costID, TOTALCOST)]
[R: Cost(costID)]
[IP: HAI_energy(item_TimeID)]
[L: Sum(HAI_energy) * 0.2 + NormalDist(0, 3)]
]
```
MTheory N.6

[F: heater_item
[C: IsA(heater_item_TimeID, TIME)]
[R: HI_temperature(heater_item_TimeID)]

[IP: HI_temperature(heater_item_TimeID), SII_temperature(heater_item_TimeID)]
[L: -12765.620178222656 * Sum(HAI_energy) + -1276562.453125 *
Sum(SII_temperature) + NormalDist(1531875628.4338844, 1.734375)]

[F: Slabs
[C: IsA(Slabs_TimeID, TIME)]
[R: S_temperature(Slabs_TimeID)]

[IP: SII_temperature(Slabs_TimeID)]
[L: Sum(SII_temperature) + NormalDist(0, 3)]

[F: H_temperature_report
[C: IsA(heater_item_TimeID, TIME)]
[R: H_temperature(heater_item_TimeID)]

[IP: HI_temperature(heater_item_TimeID)]
[L: Sum(HI_temperature) + NormalDist(0, 3)]

[F: Cost
[C: IsA(CostID, COST), IsA(Slabs_TimeID, TIME)]
[R: C_HAI(CostID)]

[IP: HAI_energy(Slabs_TimeID)]
[L: 0.2 * Sum(HAI_energy) + NormalDist(0, 3)]
]
F: heateractuator_item
[C: IsA(heateractuator_item_TimeID, TIME)]
[R: HAI_energy(heateractuator_item_TimeID)]
[IP: SII_temperature(heateractuator_item_TimeID)]
[L: -99.99999814540547 * Sum(SII_temperature) + NormalDist(119999.99995954607,.0000000006984919)]

F: slabinput_item
[C: IsA(slabinput_item_TimeID, TIME)]
[R: SII_temperature(slabinput_item_TimeID)]
[L: NormalDist(15.557843606667822,6.892498282240208)]

F: energy_cost
[C: IsA(energy_cost_TimeID, TIME), IsA(process_ID, PROCESS)]
[R: TotalEnergyCost(process_ID)]
[IP: HAI_energy(energy_cost_TimeID)]
[L: Sum(HAI_energy) * 0.2 + NormalDist(0, 0.000001)]

F: sensor
[C: IsA(sensor_TimeID, TIME)]
[R: sensor_temperature(sensor_TimeID)]
[IP: SII_temperature(sensor_TimeID)]
[L: Sum(SII_temperature) + NormalDist(0, 3)]
Appendix O. Hybrid Message Passing with Gaussian Mixture Reduction with Optimal Settings

Hybrid Bayesian Networks (HBNs), which contain both discrete and continuous variables, arise naturally in many application areas (e.g., image understanding, data fusion, medical diagnosis, fraud detection). This chapter concerns inference in an important subclass of HBNs, the conditional Gaussian (CG) networks, in which all continuous random variables have Gaussian distributions and all children of continuous random variables must be continuous. Inference in CG networks can be NP-hard even for special-case structures, such as poly-trees, where inference in discrete Bayesian networks (BN) can be performed in polynomial time. Therefore, approximate inference is required. In approximate inference, it is often necessary to trade off accuracy against solution time. This chapter presents an extension to the Hybrid Message Passing inference algorithm [Sun, 2007] for general CG networks and an algorithm for optimizing its accuracy given a bound on computation time. The extended algorithm uses Gaussian mixture reduction to prevent an exponential increase in the number of Gaussian mixture components. The trade-off algorithm performs pre-processing to find optimal run-time settings for the extended algorithm. Experimental results for four CG networks compare performance of the extended algorithm with existing algorithms and show the optimal settings for these CG networks.
0.1 Introduction

A Bayesian Network (BN) [Pearl, 1988] is a probabilistic graphical model that represents a joint distribution on a set of random variables in a compact form that exploits conditional independence relationships among the random variables. The random variables (RVs) are represented as nodes in a directed acyclic graph (DAG) in which a directed edge represents a direct dependency between two nodes and no directed cycles are allowed in the graph. Bayesian Networks have become a powerful tool for representing uncertain knowledge and performing inference under uncertainty. They have been applied in many domains, such as Image Understanding, Data fusion, Medical diagnosis, and Fraud detection, and have become a powerful tool in inference for the real world. Furthermore, an important application of BN is PSAW [Laskey et al., 2000][Wright et al., 2002][Costa et al., 2005][Suzic, 2005][Costa et al., 2009][Carvalho et al., 2010], where an SSBN is constructed from an MTheory to answer queries about a situation.

Hybrid Bayesian Network (HBNs) can contain both discrete and continuous RVs. An important subclass, the conditional linear Gaussian (CLG) networks, consists of networks in which all discrete random variables have only discrete parents, all continuous random variables have Gaussian distributions, and the conditional distribution of any Gaussian RV is linear in its Gaussian parents. Exact inference methods exist for CLG networks [Lauritzen, 1992][Lauritzen & Jensen, 2001]. However, even in special cases for which exact inference in discrete Bayesian Networks (BNs) is tractable, exact inference in CLG networks can be NP-hard [Lerner & Parr, 2001]. In particular, the posterior marginal distribution for each individual Gaussian random variable is a mixture
of Gaussian distributions, and the number of components needed to compute the exact
distribution for a given random variable may be exponential in the number of categorical
variables in the network. Furthermore, no exact algorithms exist for general CG networks
in which no discrete node may have a continuous parent node and all continuous nodes
have Gaussian probability distributions. Therefore, approximate inference for CG
networks is an important area of research.

Approximate inference algorithms for HBNs can be roughly classified into five
categories: (1) Sampling (SP), (2) Discretization (DS), (3) Structure Approximation (SA),
(4) Clustering (CL), and (5) Message Passing (MP) approaches.

SP algorithms draw random samples to use for inference and can handle BNs of
arbitrary structure. Henrion [1988] presented a basic sampling approach, called logic
sampling, for approximate inference in discrete Bayesian networks. Logic sampling
generates samples beginning at root nodes and following links to descendant nodes,
terminating at leaf nodes of the graph. If a sampled realization contains an evidence node
whose value does not match the observed value, it is rejected. The result is a sample from
the conditional distribution of the sampled nodes given the observed values of the
evidence nodes. This rejection strategy may require a very large number of samples to
converge to an acceptable inference result. Further, this strategy cannot be applied when
there are continuous evidence nodes, because in this case all samples would be rejected.
Fung & Chang [1989] suggested a method that sets all evidence variables to their
observed values, samples only the non-evidence variables, and weights each sample by
the likelihood of the evidence variables given the sampled non-evidence variables. This
likelihood weighting algorithm, which can be applied when evidence nodes are continuous, has become very popular. However, when the evidence configuration is highly unlikely, this method can result in very poor accuracy. Pearl [1988] proposed a Gibbs sampling approach for Bayesian networks. His algorithm is a special case of the more general class of Markov Chain Monte Carlo algorithms [Gilks et al., 1996]. Efficiency of Gibbs sampling can be dramatically improved by sampling only a subset (called a cutset) of random variables that breaks all loops in the graph, and performing exact inference on the remaining singly connected network [Bidyuk & Dechter, 2007]. Nevertheless, for any SP algorithm very large numbers of samples may be required for challenging BNs, such as those with complex topologies, very unlikely evidence configurations, and/or deterministic or near-deterministic relationships.

DS algorithms change a hybrid BN to a discrete BN by discretizing all continuous RVs in the hybrid BN. This approach changes a continuous variable to a set of intervals, called a bin. After the change, the discretized BN is handled by a discrete inference algorithm (e.g., [Pearl, 1988]). Kozlov & Koller [1997] provided an improved discretization by efficiently adjusting the shape of a continuous RV. However, DS algorithms start with approximation for discretization and this approximation can cause inaccurate posterior distributions. Accuracy can be improved with finer discretization, but at the cost of possibly major additional cost in time and space. Furthermore, there is a time cost for discretization, and a need for methods to choose the granularity of the distribution to balance accuracy against computation cost.
SA algorithms change an intractable hybrid BN (e.g., conditional nonlinear Gaussian network) to a tractable hybrid BN (e.g., conditional linear Gaussian network). After changing to a tractable hybrid BN, a hybrid inference algorithm that can handle the tractable hybrid BN is used for inference. Shenoy [2006] proposed a SA algorithm in which any type of a continuous RV can be approximated by a mixture of Gaussian distributions, thus converting an arbitrary hybrid BN to a CG BN. He showed how various hybrid BNs (e.g., non-Gaussian HBN, nonlinear HBN, a HBN with a continuous parent and a discrete child, and a HBN with non-constant variance) can be converted to a CG BN. Although SA algorithms can treat various types of HBNs, they require an appropriate CG inference algorithm for the converted HBN.

CL algorithms handle loops by converting the original BN to a graph of clusters in which each node corresponds to a cluster of nodes from the original BN, such that the graph of clusters contains no loops. A conversion step is required to form clusters from the original BN. Among CL approaches, the popular Junction Tree (JT) algorithm has been adapted for inference in CG networks [Lauritzen, 1992][Lauritzen & Jensen, 2001]. However, constraints required by the Lauritzen algorithm [Lauritzen, 1992][Lauritzen & Jensen, 2001] on the form of the junction tree tend to result in cliques containing very many discrete nodes. Because inference is exponential in the number of discrete nodes in a cluster, the algorithm is often intractable even when a tractable clustering approach exists for a discrete network of the same structure [Lerner & Parr, 2001]. For this reason, it is typically necessary to resort to approximate inference. Gaussian mixture reduction
(GMR) has been suggested as an approximation approach [Lerner, 2002]. GMR approximates a Gaussian mixture model (GMM) with a GMM having fewer components.

In MP algorithms, each node in the BN sends messages to relevant nodes along paths between the relevant nodes. The messages contain information to update the distributions of the relevant nodes. After updating, each of the nodes computes its marginal distribution. If the BN has loops, message passing may not converge. MP algorithms are also subject to the problem of uncontrolled growth in the number of mixture components. A GMR approach has been proposed to address this issue [Sun et al., 2010][Sun & Chang, 2010]. However, they provided no general algorithm for applying GMR within the MP algorithm. Park et al. [2015] introduced a general algorithm for MP using GMR, but included no guidance on how to trade off between accuracy and computational resources in hybrid MP using GMR.

This chapter presents a complete solution to the hybrid inference problem by providing two algorithms: Hybrid Message Passing (HMP) with Gaussian Mixture Reduction (GMR) and Optimal Gaussian Mixture Reduction (Optimal GMR).

The HMP-GMR algorithm prevents exponential growth of Gaussian mixture components in MP algorithms for inference in CG Bayesian networks. We present an extension of the algorithm of [Sun et al., 2010][Sun & Chang, 2010] that incorporates GMR to control complexity, and examine its performance relative to competing algorithms.

Each inference algorithm has its own characteristics. For example, some algorithms are faster and some are more accurate. Further, accuracy and speed can
depend on the Bayesian network and the specific pattern of evidence. These characteristics can be used as guidance for choosing an inference method for a given problem. Metrics for evaluating an inference algorithm include speed, accuracy, and resource usage (e.g., memory or CPU usage). In some situations, algorithm speed is the most important factor. In other cases, accuracy may be more important. For example, early stage missile tracking may require a high speed algorithm for estimating the missile trajectory, while matching faces in a security video against a no-fly database may prioritize accuracy over speed. The HMP-GMR algorithm requires a maximum number of Gaussian components as an input parameter. This maximum number of components influences both accuracy and execution time of the HMP-GMR algorithm. We introduce a preprocessing algorithm called HMP-GMR with Optimal Settings (HMP-GMR-OS), which optimizes the initial settings for HMP-GMR to provide the best accuracy on a given HBN under a bound on computation time. The HMP-GMR-OS algorithm is intended for cases in which a given HBN will be used repeatedly in a time-limited situation, and a pre-processing step is desired to balance accuracy against speed of inference. Sampling approaches have been used for such situations, because of their anytime property. That is, sampling always provides an answer even if it runs out of time. In some cases, our algorithm can result in better accuracy than a sampling approach for the same execution time.

The layout of this chapter is as follows. Section O.2 introduces Hybrid Message Passing Inference and Gaussian Mixture Reduction. Section O.3 presents the HMP-GMR algorithm, which combines the two methods introduced in Section O.2. Section O.4
proposes the OCB algorithm to find the optimal number of allowable components in any
given mixture distribution. Section O.5 presents experimental results on the advantages
and disadvantages of the new algorithm. Section O.6 draws conclusions.

0.2 Preliminaries

In this section, we introduce message passing inference for CG BNs and
component reduction for Gaussian mixture models.

0.2.1 Structure of Hybrid Bayesian Network

A general hybrid BN can contain both discrete and continuous nodes. A node in a
hybrid BN can be categorized according to its type (i.e., discrete or continuous), its parent
node type(s) (i.e., discrete, continuous, or hybrid with at least one discrete and one
continuous node), and its child node type(s) (i.e., discrete, continuous, or hybrid). The
following table shows all possible classifications of nodes in a hybrid BN (D stands for
discrete; C stands for continuous; and H stands for hybrid).

<table>
<thead>
<tr>
<th>Node type</th>
<th>Parent node type(s)</th>
<th>D</th>
<th>C</th>
<th>H</th>
<th>D</th>
<th>C</th>
<th>H</th>
<th>D</th>
<th>C</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
</tbody>
</table>

As shown in Table O.1, there are 18 node categories in a general hybrid BN. Various special cases impose restrictions eliminating some of the 18 categories. A hybrid
BN in which no discrete node may have a continuous parent node is called a conditional hybrid BN [Sun, 2007]. That is, a conditional hybrid BN may contain Types 1, 2, 3, 11, 14, and 17 from Table O.1. These six cases are shown in Fig. O.1 below. In the figure, a rectangle indicates a discrete node and a circle indicates a continuous node. For example, Type 1 has a discrete node $B$ with its discrete parent node $A$ and discrete child node $C$, while Type 3 has a discrete node $B$ with its discrete parent node $A$ and hybrid child nodes $C$ and $Y$.

A general hybrid BN places no restriction on the type of probability distribution for a continuous node. If all continuous nodes in a hybrid BN have Gaussian probability distributions, the BN is called Gaussian hybrid BN. A BN that is both a conditional hybrid BN and a Gaussian hybrid BN is called a conditional Gaussian (CG) BN. CG BNs can be further classified into two sub-categories: conditional linear Gaussian (CLG) BNs and conditional nonlinear Gaussian (CNG) BNs. For the CLG BNs, the Gaussian conditional distributions are always linear functions of the Gaussian parents. That is, if $X$ is a continuous node with $n$ continuous parents $U_1, \ldots, U_n$ and $m$ discrete parents $A_1, \ldots, A_m$, then the conditional distribution $p(X \mid u, a)$ given parent states $U=u$ and $A=a$ has the following form:

$$p(X \mid u, a) = \mathcal{N}(L^{(a)}(u), \sigma^{(a)})$$

(O.1)
where \( L^{(a)}(u) = m^{(a)} + b_1^{(a)}u_1 + \cdots + b_n^{(a)}u_n \) is a linear function of the continuous parents, with intercept \( m^{(a)} \), coefficients \( b_i^{(a)} \), and standard deviation \( \sigma^{(a)} \) that depend on the state \( a \) of the discrete parents.

A Gaussian conditional distribution for a continuous node in a CNG BN can be any function of the Gaussian and discrete parents. The form is similar to Equation O.1 except that \( L^{(a)}(u) \) can be a nonlinear function.

Note that the types of Fig. O.1 cover any number of parent and child nodes. Thus, the discrete node \( B \) can have a set of discrete parent nodes (i.e., \( A = \{A_1, A_2, ..., A_l\} \)), a set of discrete child nodes (i.e., \( C = \{C_1, C_2, ..., C_m\} \)), and/or a set of continuous child nodes (i.e., \( Y = \{Y_1, Y_2, ..., Y_n\} \)). The continuous node \( X \) can have a set of discrete parent nodes (i.e., \( A = \{A_1, A_2, ..., A_l\} \)), a set of continuous parent nodes (i.e., \( U = \{U_1, U_2, ..., U_m\} \)), and a set of continuous child nodes (i.e., \( Y = \{Y_1, Y_2, ..., Y_n\} \)). We use this notation to
introduce message passing inference for a discrete BN, a continuous BN, and a hybrid BN.

**O.2.2 Message Passing Inference for Discrete BN**

Message passing inference for a discrete BN was introduced in [Pearl, 1988]. A discrete BN contains only discrete nodes (i.e., Type 1 in Fig. O.1). The objective of inference is to compute the function

$$\text{BEL}(B) = P(B \mid e)$$

for each node $B$ in the Bayesian network. Here, $B$ denotes a node with its associated RV, $e$ is a set of evidence events consisting of state assignments for nodes in the network, and $\text{BEL}(B)$ is the conditional distribution of $B$ given the values of the evidence RVs. If the BN is a polytree (has no undirected cycles), the evidence $e$ can be split into two components, $e^+_B$ and $e^-_B$, where $e^+_B$ relates only to non-descendants of $B$ and $e^-_B$ relates only to descendants of $B$. Equation O.2 can be decomposed as follows:

$$P(B \mid e) = \alpha P(B \mid e^+_B, e^-_B)$$

$$= \alpha P(B \mid e^+_B)P(B \mid e^-_B)$$

$$= \alpha \pi(B) \lambda(B)$$

where $\alpha$ denotes a normalizing constant. The second line is valid because in a polytree, $e^+_B$ and $e^-_B$ are independent given $B$. The factors relating to non-descendants and
descendants are denoted $\pi(B)$ and $\lambda(B)$, as shown in the third line of the equation. These factors are called the Pi and Lambda functions, respectively.

The Pi function $\pi(B)$ and Lambda function $\lambda(B)$ can be written as follows:

$$\pi(B) = \sum_{A} P(B \mid A) \prod_{i} \pi_B(A_i) \tag{O.3}$$

and

$$\lambda(B) = \prod_{j} \lambda_C(B) \tag{O.4}$$

where $\pi_B(A_i)$ and $\lambda_C(B)$ denote a Pi message from the parent $A_i$ to $B$ and Lambda message from the child $C_j$ to $B$, respectively. These messages can be written as follows:

$$\pi_C(B) = \alpha \left[ \prod_{k \neq j} \lambda_C(B) \right] \pi(B) \tag{O.5}$$

and

$$\lambda_B(A_i) = \sum_{B} \lambda(B) \sum_{A_k: k \neq i} P(B \mid A) \prod_{k \neq i} \pi_B(A_k) \tag{O.6}$$

Note that the Lambda function $\lambda(B)$ is similar to the Pi message $\pi_C(B)$ except that the Pi message includes a factor of $\pi(B)$, includes Lambda messages all children of the parent except the target child node $j$, and includes a normalizing constant $\alpha$. A similar
relationship exists between the Pi function \( \pi(B) \) and the Lambda message \( \lambda_B(A_i) \). The Lambda message multiplies by \( \lambda(B) \) and includes Pi messages only from parent nodes other than the target parent node \( i \).

The MP algorithm is given as follows.

- **Initialization:** For any evidence node \( B=b \), set \( \pi(B) = \lambda(B) \) to 1 for \( B=b \) and 0 for \( B\neq b \). For any non-evidence node with no parents, set \( \pi(B) \) to the prior distribution for \( B \). For any non-evidence node \( B \) with no children, set \( \lambda(B) \) uniformly equal to 1.

- **Iterate until no change occurs:**
  - For each node \( B \), if \( B \) has received Pi messages from all its parents, then calculate \( \pi(B) \).
  - For each node \( B \), if \( B \) has received Lambda messages from all its parents, then calculate \( \lambda(B) \).
  - For each node \( B \), if \( \pi(B) \) has been calculated and \( B \) has received Lambda messages from all its children except \( C \), calculate and send the Pi message from \( B \) to \( C \).
  - For each node \( B \), if \( \lambda(B) \) has been calculated and \( B \) has received Pi messages from all its parents except \( A \), calculate and send the Lambda message from \( B \) to \( A \).

- **Calculate** \( \mathbb{P}(B \mid e) = \alpha \pi(B) \lambda(B) \) for each node \( B \).
This algorithm finds exact values of all $\pi(B)$ if the network is a polytree. The algorithm can also be applied to BNs containing undirected cycles. It is not guaranteed to converge, but when it converges, it often results in a good approximation to the correct posterior probabilities [Murphy et al, 1999].

### O.2.3 Message Passing Inference for Continuous BNs

Type 14 in Fig. O.1 is a BN in which all nodes are continuous. The MP algorithm can be extended to this case by defining Pi/Lambda messages as follows [Sun, 2007]. These messages can be computed exactly for linear Gaussian networks. For general Gaussian networks, the messages can be approximated using the unscented transformation [Uhlmann, 1995] to project mean and covariance estimates through nonlinear transformations.

The Pi and Lambda functions for a continuous node $X$ in a Gaussian network can be written as follows:

\[
\pi(X) = \int_{\mathbf{U}} P(X \mid \mathbf{U}) \prod_i \pi_X(U_i) \ d\mathbf{U} \tag{O.7}
\]

and

\[
\lambda(X) = \prod_j \lambda_{Y_j}(X) \tag{O.8}
\]

where $d\mathbf{U}$ is the m-dimensional differential with $\mathbf{U} = \{U_1, U_2, \ldots, U_m\}$, $\pi_X(U_i)$ denotes the Pi message from the continuous parent $U_i$ to $X$, and $\lambda_{Y_j}(X)$ denotes the Lambda message...
from the continuous child $Y_j$ to $X$. These Pi and Lambda messages can be written as follows:

$$\pi_{Y_j}(X) = \alpha \left[ \prod_{k \neq j} \lambda_{Y_j}(X) \right] \pi(X) \tag{O.9}$$

and

$$\lambda_X(U_j) = \int_X \lambda(X) \int_{\bar{U}} P(X \mid U) \prod_{k \neq j} \pi_X(U_k) \ d\bar{U} \ dX \tag{O.10}$$

where $d\bar{U}$ is the $(m - 1)$-dimensional differential with $\bar{U} = \{s \mid s \in U \text{ and } s \neq U_j\}$.

**O.2.4 Message Passing Inference for Hybrid BNs**

For a conditional hybrid BN (i.e., Types 1, 2, 3, 11, 14, and 17 in Fig. O.1), the Pi/Lambda functions and the Pi/Lambda messages can be extended as follows [Sun, 2007].

The Pi function for a discrete node $B$ (i.e., Types 1, 2, and 3) is given by Equation O.3. The Pi function for the continuous node $X$ of Type 17 is given as follows:

$$\pi(X) = \sum_A \int_{U} P(X \mid A, U) \prod_i \pi_X(A_i) \prod_j \pi_X(U_j) \ dU \tag{O.11}$$
where $d\mathbf{U}$ is the m-dimensional differential, $\mathbf{U} = \{ U_1, U_2, ..., U_m \}$, $\pi_X(A_i)$ denotes the Pi message from the discrete parent $A_i$ to $X$ and $\pi_X(U_j)$ denotes the Pi message from the continuous parent $U_j$ to $X$. Derivation of Pi functions for Types 11 and 14 is straightforward by use of Equation O.11.

The Lambda function for Types 11, 14, and 17 is given by Equation O.8. The Lambda function for the discrete node $B$ of Type 3 is given as follows:

$$\lambda(B) = \prod_i \lambda_{C_i}(B) \prod_j \lambda_{Y_j}(B) \quad \text{(O.12)}$$

where $\lambda_{C_i}(B)$ denotes the Lambda message from the discrete child node $C_i$ to $B$ and $\lambda_{Y_j}(B)$ denotes the Lambda message from the continuous child node $Y_j$ to $B$. Derivation of Lambda functions for Types 1 and 2 is straightforward by use of Equation O.12.

The Pi message for Types 11, 14, and 17 is given by Equation O.9. The Pi message for Type 3 is given as follows:

$$\pi_{Y_j}(B) = \alpha \left[ \prod_i \lambda_{C_i}(B) \prod_{k \neq j} \lambda_{Y_k}(B) \right] \pi(B) \quad \text{(O.13)}$$

The Lambda message for Types 1, 2, and 3 is given by Equation O.6. The Lambda message for the continuous node $X$ of Type 17 can be written as follows:
\[ \lambda_X(U_j) = \int_X \lambda(X) \sum_A \int_\mathbf{\bar{U}} \mathbf{P}(X \mid A, U) \prod_i \pi_X(A_i) \prod_{k \neq j} \pi_X(U_k) \ d\mathbf{\bar{U}} dX \quad (O.14) \]

and

\[ \lambda_X(A_i = a) = \int_X \lambda(X) \sum_A \int_\mathbf{\bar{U}} \mathbf{P}(X \mid A_i = a, \mathbf{\bar{A}}, U) \prod_i \pi_X(A_i) \prod_{k \neq i} \pi_X(A_k) \prod_j \pi_X(U_j) \ d\mathbf{\bar{U}} dX \quad (O.15) \]

where \( d\mathbf{\bar{U}} \) is the \((m - 1)\)-dimensional differential with \( \mathbf{\bar{U}} = \{s \mid s \in U \text{ and } s \not\in U_j\} \), \( \mathbf{\bar{A}} = \{s \mid s \in A \text{ and } s \not\in A_i\} \), and \( a \) denotes a state of \( A_i \). The first equation is for the message to the continuous parent \( U_j \) and the second equation is for the message to the discrete parent \( A_i \). These apply respectively to Types 11 and 14.

### 0.2.5 Gaussian Mixture Reduction

Gaussian mixture reduction (GMR) approximates an \( M \)-component GMM with a reduced number \( N < M \) of components. Several methods for GMR have been proposed, e.g., [Salmond, 1990][West, 1993][Williams, 2003][Williams \& Maybeck, 2003][Schrempf et al., 2005][Runnalls, 2007].

A straightforward and basic method for performing GMR [West, 1993] is the following:

1. Find the two closest components in a GMM according to a distance criterion.
2. Merge the two selected components into one component.
(3) Update to a GMM with one fewer component.

(4) Repeat steps 1-3 until a stopping criterion is reached (e.g., a predefined number of components, and a predefined precision).

As a distance criterion, Runnalls [2007] proposed the Kullback-Leibler (KL) divergence [Kullback & Leibler, 1951]. The distance criterion using the KL divergence is written as follows [Runnalls, 2007].

\[
d\left((w_i, \mu_i, \sigma_i), (w_j, \mu_j, \sigma_j)\right) = \frac{\left(\frac{(w_i+w_j) \log \det (\sigma_{ij})-(w_i) \log \det (\sigma_i)-(w_j) \log \det (\sigma_j)}{2}\right)}{\sigma_i}
\]  

(0.16)

where \(i\) and \(j\) denote the \(i\)-th and \(j\)-th component of a GMM, respectively, and \(w_k, \mu_k, \sigma_k\) are the weight, mean, and covariance of the \(k\)-th component, respectively.

Recently, a more efficient algorithm using constraint optimization was proposed [Chen et al., 2010][Chang & Sun, 2010]. This more complicated algorithm will be utilized in future work.

**O.3 Extended Hybrid Message Passing Algorithm**

The previous sections introduced message passing inference and Gaussian mixture reduction. This section combines these methods into an extended hybrid message passing algorithm for CG BNs.

The GMR operation is denoted as a function, \(\tau(gmm, max_nc)\) that applies Equation O.16, where \(gmm\) is a Gaussian mixture model and \(max_nc\) is the maximum number of allowable mixture components.
To specify the algorithm, we need to define where in the inference process GMR will be applied. For this, the Pi/Lambda functions for $X$ and Pi/Lambda messages $U \rightarrow X$ in Type 14 and $\{A, U\} \rightarrow X$ in Type 17 are chosen. Hence, the function $\tau(gmm, max_{nc})$ is applied to Equations O.7, O.8, O.9, O.10, O.11, O.14, and O.15. For the extended algorithm, these equations become:

\[
\pi(X) = \tau \left( \int_U P(X \mid U) \prod_i \pi_X(U_i) \ dU, M \right) \tag{O.17}
\]

\[
\pi(X) = \tau \left( \sum_A \int_U P(X \mid A, U) \prod_i \pi_X(A_i) \prod_j \pi_X(U_j) \ dU, M \right) \tag{O.18}
\]

\[
\lambda(X) = \tau \left( \prod_j \lambda_{Y_j}(X), M \right) \tag{O.19}
\]

\[
\pi_{Y_j}(X) = \alpha \tau \left( \prod_{k \neq j} \lambda_{Y_k}(X), M \right) \pi(X) \tag{O.20}
\]

\[
\lambda_X(U_j) = \int_X \lambda(X) \tau \left( \int_U P(X \mid U) \prod_k \pi_X(U_k) \ dU, M \right) dX \tag{O.21}
\]

\[
\lambda_X(U_j) = \int_X \lambda(X) \tau \left( \sum_A \int_U P(X \mid A, U) \prod_i \pi_X(A_i) \prod_{k \neq j} \pi_X(U_k) \ dU, M \right) dX \tag{O.22}
\]

\[
\lambda_X(A_i = a) = \int_X \lambda(X) \tau \left( \sum_{A_i} P(X \mid A_i = a, \bar{A}, U) \prod_k \pi_X(A_k) \prod_j \pi_X(U_j) \ dU, M \right) dX \tag{O.23}
\]

where $M = max_{nc}$ denotes the maximum allowable number of components.
The above equations are implemented in the following algorithm, called *Hybrid Message Passing Algorithm with Gaussian Mixture Reduction* (HMP-GMR). The following algorithm is an extension of a Hybrid Message Passing algorithm (HMP) from [Sun, 2007] to which we apply GMR. of their anytime property. An initial version of this algorithm was introduced in [Park et al., 2015]. In contrast with the initial version, this is an anytime algorithm that can provide a solution even if it is interrupted before completion.

**Algorithm 1:** Hybrid Message Passing (HMP) with Gaussian Mixture Reduction (GMR)

**Algorithm**

**Procedure** HMP-GMR (\
\[\text{net,} \quad \text{max\_time,} \quad \text{max\_iteration,} \quad \text{max\_nc,} \quad \text{max\_prcs}\]
\[\text{// a Hybrid BN} \quad \text{// a maximum execution time} \quad \text{// a maximum number of iterations} \quad \text{// a maximum allowable number of components} \quad \text{// a maximum precision}\]
\)

\[\text{1 for } i = 1, \ldots \text{ until max\_iteration}\]

\[\text{2 for } j = 1, \ldots \text{ until the number of nodes in net}\]

\[\text{3 } \pi_j \leftarrow \text{ComputeAllPiMsgs}(j, \text{max\_nc}, \text{max\_time})\]

\[\text{4 } \lambda_i \leftarrow \text{ComputeAllLambdaMsgs}(j, \text{max\_nc}, \text{max\_time})\]

\[\text{5 SendPiMsg}(j, \text{max\_nc}, \text{max\_time})\]

\[\text{6 SendLambdaMsg}(j, \text{max\_nc}, \text{max\_time})\]

\[\text{7 for } j = 1, \ldots \text{ until the number of nodes in net}\]

\[\text{8 bel}_{ij} \leftarrow \text{compute belief function using } \lambda_i \text{ and } \pi_j \text{ (O.2)}\]

\[\text{9 diff}_{ij} \leftarrow \text{compare distribution difference between } \text{bel}_{ij} \text{ and } \text{bel}_{i-1,j}\]

\[\text{10 max\_diff} \leftarrow \text{get maximum difference between } \text{diff}_{ij} \text{ and } \text{max\_diff}\]

\[\text{11 if } \text{max\_diff < max\_prcs then break}\]

\[\text{12 return a set of } \text{bel}_{ij}\]

**Procedure** ComputeAllPiMsgs (\
\[\text{j,} \quad \text{// a current node} \quad \text{max\_time,} \quad \text{// a maximum execution time} \quad \text{max\_nc} \quad \text{// a maximum allowable number of components}\]
\)

\[\text{1 if } j \text{ is discrete then do (O.3)}\]

\[\text{2 else if } j \text{ is continuous then}\]
if parent of \( j \) is discrete then do (O.11)
if parent of \( j \) is continuous then do (O.17) with \( M = max\_nc \)
if parent of \( j \) is hybrid then do (O.18) with \( M = max\_nc \)

\( exe\_time \leftarrow \) get a current execution time
if \( max\_time < exe\_time \) do not update \( \pi \)
return \( \pi \)

Procedure ComputeAllLambdaMsgs ( \( j \), \( max\_time \), \( max\_nc \))

1 if \( j \) is discrete then do (O.4)
2 else if \( j \) is continuous then do (O.19) with \( M = max\_nc \)
3 \( exe\_time \leftarrow \) get a current execution time
4 if \( max\_time < exe\_time \) do not update \( \lambda \)
5 return \( \lambda \)

Procedure SendPiMsg( \( j \), \( max\_time \), \( max\_nc \))

1 for \( k = 1, \ldots \) until number of children of \( j \)
2 if \( j \) is discrete then do (O.5) for \( k \)
3 else if \( j \) is continuous then do (O.20) for \( k \) with \( M = max\_nc \)
4 \( exe\_time \leftarrow \) get a current execution time
5 if \( max\_time < exe\_time \) do not update the Pi message from one of (O.5) and (O.20)

Procedure SendLambdaMsg ( \( j \), \( max\_time \), \( max\_nc \))

1 for \( k = 1, \ldots \) until number of parents of \( j \)
2 if \( j \) is discrete then do (O.6) for \( k \)
3 else if \( j \) is continuous then
4 if parent of \( j \) is discrete then do (O.15)
5 if parent of \( j \) is continuous then (O.21) with \( M = max\_nc \)
6 if parent of \( j \) is hybrid then do (O.22), (O.23) with \( M = max\_nc \)
7 \( exe\_time \leftarrow \) get a current execution time
8 if \( max\_time < exe\_time \) do not update the Lambda message from one of (O.6), (O.15), (O.21), (O.22), and (O.23)
HMP-GMR has five inputs. The first input, net, is the Hybrid BN with specified evidence nodes and their values. The second input, max_time, is the maximum execution time used to control how long this algorithm runs by comparing to a current execution time, exe_time, representing a period from the algorithm starting time to the current time. The third input, max_iteration, is the maximum number of iterations allowed, where an iteration is one round in which all nodes in the BN perform their operations. The fourth input, max_nc, is the maximum number of Gaussian components that may be output by the GMR function $\tau$. The fifth input, max_prcs, is a threshold on the distance between posterior distributions of nodes in the current and previous iterations. The algorithm terminates when the distance is lower than the threshold. HMP-GMR outputs approximate posterior distributions of all nodes.

Given these inputs, the algorithm proceeds as follows: (1) The algorithm iterates message passing from 1 to the maximum number of iterations or until it is interrupted due to exceeding the time limit. (2) The algorithm cycles through all nodes in the BN. (3) For the $j$-th node, all Pi messages from its parents are computed to calculate the Pi value $\pi_j$. If the RV is discrete, Equation O.3 is used, while if it is continuous and has only discrete, only continuous, or hybrid parents, Equation O.11, O.17, or O.18, is used, respectively. (4) All Lambda messages from children of the $j$-th node are computed to calculate the Lambda value $\lambda_j$. If the RV is discrete, Equation O.4 is used, while if it is continuous Equation O.19 is used. (5) A Pi message is sent from the $j$-th node to its children. If the node is discrete, Equation O.5 is used, while if it is continuous Equation O.20 is used. (6)
A Lambda message is sent from the \( j \)-th node to its parents. If the node is discrete, Equation O.6 is used, while if it is continuous and has only discrete, only continuous, or hybrid parents, Equation O.15, O.21, or O.22 (for continuous parent) / O.23 (for discrete parent) is used, respectively. For each of these functions in Lines 3, 4, 5, and 6, if the current execution time exceeds the maximum execution time, the result from the function is not updated and the for-loop in Line 2 of the HMP-GMR procedure is stopped. After all nodes have passed their messages (i.e., Line 2 ~ 6), (7) the belief function is computed for all nodes. (8) The Lambda and Pi values are multiplied and normalized for all nodes to calculate the belief function \( \text{bel}_{ij} \). (9) The difference \( \text{diff}_{ij} \) between the current and previous beliefs are computed for all nodes. (10) The maximum difference \( \text{max\_diff} \) between current and previous belief is selected. (11) If the maximum difference \( \text{max\_diff} \) is less than the maximum precision \( \text{max\_prcs} \), the iteration of the message passing is stopped. (12) Upon stopping, the algorithm outputs approximate posterior marginal distributions for all nodes.

There are three exit points from the iteration: (1) when the maximum number of allowable iterations is reached, (2) when the maximum difference is less than the maximum precision, and (3) when the current execution time for the algorithm exceeds the maximum execution time.

**0.4 Optimizing the Settings of HMP-GMR**

In some situations, the Hybrid Message Passing with Gaussian Mixture Reduction (HMP-GMR) algorithm performs better than other algorithms. For example, although in theory a sampling algorithm can be made as accurate as desired, for a given problem,
HMP-GMR may have higher accuracy for a given limit on computation time. However, HMP-GMR requires initial settings before it executes. The performance of HMP-GMR depends on these initial settings. More specifically, HMP-GMR requires that the maximum allowable number of components $\textit{max}\_nc$ and the maximum number of allowable iterations $\textit{max}\_\text{iteration}$ are specified as inputs. If the maximum allowable number of components is too small, accuracy may be too low; but if it is too large, execution time may be unacceptably long. Also, the maximum number of allowable iterations can influence accuracy and execution time. The number of components required to achieve a given accuracy depends on the network topology, the placement of continuous and discrete nodes, and the conditional distributions. As noted above, when the BN contains loops, the HMP-GMR algorithm may not converge. Thus, in some problems, HMP-GMR may spend many iterations without a significant improvement in accuracy.

Therefore, there is a need to trade off accuracy against execution time depending on the maximum allowable number of components and the maximum number of iterations. Different applications pose different requirements on execution time. It is assumed that the maximum allowable execution time for inference is an input parameter that is specified before the inference algorithm runs. Therefore, the optimization problem is defined as attaining the best achievable accuracy for a given constraint on execution time, by varying the maximum allowable number of components and the maximum allowable number of iterations.
Finding an exact optimum would be infeasible in the general case. Therefore, this section presents a Monte Carlo method to find approximately optimal values for a specific conditional Gaussian Bayesian network. The algorithm is appropriate for problems in which a given HBN is specified \textit{a priori}, and inference on the HBN will be performed repeatedly in a time-restricted setting with limits on execution time for inference. The optimization can be performed offline as a preprocessing step to find good initial settings for performing HMP-GMR inference at run time. For example, a real-time threat detection system might use a CG HBN to process sensor inputs automatically and issue alarms when suspicious combinations of sensor readings occur. Because the system runs in real time, fast execution is essential. At design time, an offline optimization can be run using the algorithm presented here to determine the best settings for the maximum number of components and the maximum number of iterations.

An optimization problem for this situation can be formulated as shown Equation O.24. In this setting, we assume that a specific HBN is given.

\[
\min_{nc \in \mathcal{NC}, it \in \mathcal{IT}} f(nc, it)
\]

subject to: \( t < \text{max\_time} \)

where \( \mathcal{NC} \) means a set of the candidate maximum allowable numbers of components \( \{nc_1, nc_2, ..., nc_n\} \), \( \mathcal{IT} \) means a set of the candidate maximum iteration numbers \( \{it_1, it_2, ..., it_m\} \), \( f(.) \) is an objective function measuring error of HMP-GMR, \( t \) means the current execution time for inference of HMP-GMR, and \( \text{max\_time} \) means the maximum
execution time. We call this as HMP-GMR with Optimal Settings (HMP-GMR-OS), which finds the values \((nc\ \text{and} \ it)\) that achieve the best accuracy under a given time restriction. Equation O.25 shows the objective function \(f(.)\) of HMP-GMR-OS.

\[
f(nc, it) = \frac{\sum_{e \in E} \text{err}(s(net, e), h(net, e, max\_time, nc, it))}{|E|}
\]  

(O.25)

where \(E\) means a set of the candidate evidence \(\{e_1, e_2, ..., e_l\}\), which are randomly selected, for a Bayesian network \(net\), \(err(.)\) means a function resulting in an error (e.g., KL-divergence [Kullback & Leibler, 1951]) between an inference result from sampling and an inference result from HMP-GMR, \(s(.)\) means a sampling inference algorithm used for exact inference, \(h(.)\) means the HMP-GMR algorithm with a maximum execution time \(max\_time\), a candidate maximum allowable number of components \(nc\), and a candidate maximum iteration number \(it\).

The Monte Carlo method for HMP-GMR, called an HMP-GMR-OS algorithm (Algorithm 2), finds the best values for the two decision variables given a Hybrid BN, a maximum execution time, a number of samples, an upper limit on the maximum number of iterations, and an upper limit on the maximum allowable number of components.

The HMP-GMR-OS algorithm has five inputs. The first input \(net\) is a Hybrid BN. The second input \(max\_time\) is the maximum execution time for inference of HMP-GMR. The third input \(num\_samples\) is the number indicating how many times the simulation should be repeated. The fourth input \(ul\_max\_it\) is the number of maximum iterations used
for inference of HMP-GMR. The fifth input \( ul\_max\_nc \) is the number indicating how many the number of maximum allowable number of components will be investigated.

**Algorithm 2:** HMP-GMR with Optimal Settings (HMP-GMR-OS) Algorithm

**Procedure** HMP-GMR-OS ( )

\[
\begin{align*}
\text{net}, & \quad // \text{a Hybrid BN} \\
\text{max\_time}, & \quad // \text{a maximum execution time} \\
\text{num\_samples}, & \quad // \text{a number of samples} \\
\text{ul\_max\_it}, & \quad // \text{an upper limit on the maximum number of iterations} \\
\text{ul\_max\_nc} & \quad // \text{an upper limit on the maximum allowable number of components}
\end{align*}
\]

1. for \( i = 1, \ldots \) until \( \text{num\_samples} \)
2. \( e_i \leftarrow \) generate randomly a set of evidence values from \( \text{net} \)
3. \( s_i \leftarrow \) inference using a sampling algorithm with \( \text{net} \) and \( e_i \)
4. for \( j = 1, \ldots \) until \( \text{ul\_max\_nc} \)
5. for \( k = 1, \ldots \) until \( \text{ul\_max\_it} \)
6. \( h_{jk} \leftarrow \) inference using HMP-GMR with \( \text{net}, e_i, \text{max\_time}, j, \) and \( k \)
7. \( r_{ijk} \leftarrow \) calculate an inference error value between \( s_i \) and \( h_{jk} \)
8. \( r_{jk} \leftarrow \) add \( r_{ijk} \) to a set of inference error values \( r_{jk} \) for \( j \) and \( k \)
9. for \( j = 1, \ldots \) until \( \text{ul\_max\_nc} \)
10. for \( k = 1, \ldots \) until \( \text{ul\_max\_it} \)
11. \( \text{avg\_r}_{jk} \leftarrow \) calculate an average inference error for \( r_{jk} \)
12. [\( nc, it \)] \leftarrow \text{select best values for} \ nc \text{ and} \ it \text{ in (O.25) using} \ \text{avg\_r}_{jk}
13. return [\( nc, it \)]

Given these inputs, the algorithm proceeds as follows: (1) The algorithm simulates the given number of samples. (2) The algorithm randomly selects some evidence nodes from the Hybrid BN \( \text{net} \). Also, it randomly selects a reasonable evidence value for each evidence node (i.e., a highly unlikely value is not used for the evidence value) and provides an \( i \)-th set of evidence values \( e_i \). (3) The set of the evidence values is used for inference of a sampling algorithm by which nearly correct results of inference \( s_i \).
(i.e., posterior distributions) are found. (4) The maximum allowable number of components, denoted by \( j \), is varied from 1 to the upper limit on the maximum allowable number of components \( ul\_max\_nc \). (5) The maximum number of iterations, denoted by \( k \), is varied from 1 to the upper limit on the maximum number of iterations \( ul\_max\_it \). (6) This algorithm uses the HMP-GMR algorithm with the Hybrid BN net, the set of the evidence values \( e_i \), the maximum execution time \( max\_time \), the maximum allowable number of components \( j \), and the maximum number of iterations \( k \). Then, the HMP-GMR algorithm provides the results \( h_{jk} \) (i.e., posterior distributions). (7) An inference error value \( r_{ijk} \) between the nearly correct results \( s_i \) and the HMP-GMR’s result \( h_{jk} \) is computed by using a distance function (e.g., KL-divergence [Kullback & Leibler, 1951]). (8) The inference error value \( r_{ijk} \) at \( i \)-th sample for \( j \) and \( k \) is stored at a set of inference error values \( r_{jk} \) for \( j \) and \( k \). After simulating all samples, for all \( j \) (9) and \( k \) (10), (11) an average inference error \( \text{avg}_{r_{jk}} \) is calculated using the set of the inference error values \( r_{jk} \). (12) A best maximum allowable number of components \( nc \) and a best maximum number of iterations \( it \) are selected by finding a minimum average inference error from \( \text{avg}_{r_{jk}} \). (13) The algorithm outputs the best values \( nc \) and \( it \).

In summary, the HMP-GMR-OS algorithm is a preprocessing algorithm finding the optimal settings for HMP-GMR given an HBN to improve accuracy before HMP-GMR for the HBN executes for a practical situation.

0.5 Experiment

This section presents experiments to evaluate the performance of the HMP-GMR algorithm and the Optimal GMR algorithm. For evaluation of the HMP-GMR algorithm,
Park et al. [2015] presented simple experiments to demonstrate scalability and efficiency using two hybrid BNs. Here, more extensive experiments are performed on four BNs. These four BNs would be representative BNs in terms of various numbers of discrete parent nodes and various numbers of loopy structures in a given network.

Fig. O.2 shows two illustrative Conditional Gaussian (CG) BNs (i.e., 1 and 2) containing a discrete node $A$ with 4 states, a continuous node $X_j$, and another continuous node $Y_j$. These two BNs differ in the links between $Y_j$ and $Y_{j+1}$. The first, shown on the left in Fig. O.2, has no undirected cycles involving only continuous nodes, while the second, shown on the right in Fig. O.2, has undirected cycles among. For example, between $X_1$ and $Y_2$, there are two paths: $X_1 \rightarrow X_2 \rightarrow Y_2$ and $X_1 \rightarrow Y_1 \rightarrow Y_2$.

![Figure O.2 Conditional Gaussian BN 1 in the left figure and Conditional Gaussian BN 2 in the right figure](image)

Fig. O.3 shows two additional cases in which the BNs contain a large number of discrete nodes. Each discrete node $A_i$ has four states and the continuous nodes $X_i$ and $Y_j$ are conditional Gaussians. Again, the difference between the two BNs is that the third has no undirected cycles involving only continuous nodes, while the fourth has loopy structure for the continuous nodes.

369
The experiments examined both CLG and CNG BNs with these four CG BNs. The size of the four CG BNs was varied by adjusting $n$. Some of the leaf continuous nodes $\{Y_1, \ldots, Y_n\}$ were randomly selected as evidence nodes. All other nodes were unobserved.

Table O.2 shows characteristics of these four CG BNs. In these BNs, the discrete nodes have four states. Therefore, in CG BNs 1 and 2, there are four discrete states for discrete node $A$, while in CG BNs 3 and 4, there are $4^n$ total configurations of the discrete states. This is $4^7 = 16384$ configurations when $n = 7$. When $n = 7$, CG BNs 1 and 4 contain 21 cycles, while CG BN 2 contains 501 cycles\textsuperscript{17}. CG BN 3 contains no cycles.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
 & CG BN 1 & CG BN 2 & CG BN 3 & CG BN 4 \\
\hline
Cycles & 21 & 501 & 0 & 21 \\
\hline
Combinations of discrete states & 4 & 4 & 16384 & 16384 \\
\hline
\end{tabular}
\caption{Characteristics of four CG BNs with $n = 7$}
\end{table}

\textsuperscript{17} Cycles were derived by a cycle finding algorithm [Johnson, 1975].
The following factors were varied in the experiment: (1) Type of hybrid BN (i.e., BN 1, 2, 3, or 4; CLG or CNG BN), (2) type of inference algorithm (i.e., Hybrid Junction Tree (Hybrid-JT) [Lauritzen, 1992][Lauritzen & Jensen, 2001], original Hybrid MP (HMP) [Sun, 2007], Hybrid MP with Gaussian Mixture Reduction (HMP-GMR) [Park et al., 2015], or Likelihood Weighting (LW) sampling [Fung & Chang, 1989]), (3) number of repeated structures $n$, (4) algorithm characteristics (i.e., the number of GMM components allowed, the number of allowable message passing iterations, and the maximum precision for the message passing algorithms). The dependent variables were accuracy of result and execution time. For all experiments, the convergence criterion for HMP and HMP-GMR was $max_{prs} = 10^{-3}$.

Using these settings, we conducted three experiments: (1) A comparison between HMP, HMP-GMR, and Hybrid-JT investigated scalability of HMP-GMR to complex networks (i.e., larger values of $n$), (2) the HMP-GMR itself was evaluated for posterior distribution accuracy and execution time, and (3) optimal settings derived from the HMP-GMR-OS algorithm were evaluated according to inference accuracy. The experiments were run on a 3.40GHz Intel Core i7-3770 processor. The algorithms were implemented in the Java programming language. In a Java code for HMP-GMR, there was another exit point from the iteration of the HMP-GMR algorithm in Section O.3. In some cases, computation of Lambda values and Pi values in the HMP-GMR algorithm could not be completed due to numeric underflow. This happened when the HMP-GMR algorithm diverged. When this occurred, the HMP-GMR algorithm stopped and provided its current solution.
0.5.1 Scalability of HMP-GMR

The first experiment examined improvement in scalability of HMP-GMR over HMP and Hybrid-JT.

The initial setting of this experiment consists of (1) maximum of 4 components output by GMR and (2) 100 iteration limit for each of HMP and HMP-GMR. Eight CG BNs (i.e., conditional linear/nonlinear cases for CG BNs 1, 2, 3, and 4) were run with HMP, HMP-GMR, and Hybrid-JT using the following inputs and outputs. The input value of $n$ for both BNs was varied from 1 to 10. The output value is the execution time.

Fig. O.4 and O.5 show the results of this experiment summarizing the execution times for the CLG case as the number of nodes $n$ is varied. The X axis for each figure denotes the number of nodes $n$. The Y axis for each figure denotes the execution time in milliseconds. The solid line denotes the HMP-GMR results. The dashed line denotes the HMP results. The dotted line denotes the Hybrid-JT results.

![Figure O.4 Execution Times over n on CG BNs 1 and 2](image)

Results for CG BNs 1 and 2 show a similar pattern. The HMP algorithm with no GMR exceeded the time limit at $n = 7$ and $n = 4$, respectively. Execution times for HMP-
GMR were higher than those for Hybrid-JT in both cases. The increase in execution time for both HMP-GMR and Hybrid-JT was linear in $n$.

![Figure O.5 Execution Times over $n$ on CG BNs 3 and 4](image)

In Fig. O.5, results from HMP and Hybrid-JT show exponential growth in execution time, while execution time of HMP-GMR increased linearly. For HMP, the execution time limit for CG BNs 3 and 4 was exceeded at $n = 7$ and $n = 4$, respectively. For Hybrid-JT, the execution time limit for CG BNs 3 and 4 was exceeded at $n = 9$.

The initial setting of this experiment consists of (1) maximum of 4 components output by GMR and (2) 100 iteration limit for each of HMP and HMP-GMR. Four CNG BNs (i.e., conditional nonlinear cases for CG BNs 1, 2, 3, and 4) were run with HMP, HMP-GMR, and Hybrid-JT using the following inputs and outputs. The input value of $n$ for both BNs was varied from 1 to 10. The output value is the execution time.
Results for the CNG networks showed similar patterns. These experiments showed that HMP-GMR is scalable to large BNs for both linear and nonlinear CG networks. However, scalability alone is not sufficient. Accuracy and good operational performance are also essential.

0.5.2 Accuracy and Efficiency of HMP-GMR
In this experiment, we investigated the accuracy and convergence of HMP-GMR for the four CLG BNs. To evaluate the accuracy of HMP-GMR, exact inference results using Hybrid-JT inference were used. Some of the runs using Hybrid-JT stopped because
of the exponential growth of components, so Hybrid-JT produced posterior distributions only for \( n \leq 7 \). For this reason, this experiment used \( n = 7 \) for the four CLG BNs. Accuracy was measured by KL-divergence [Kullback & Leibler, 1951] (lower values mean better accuracy). We calculated the KL-divergence between exact and approximate results for each unobserved node, and summed them (henceforth, we use KL-divergences to mean the sum of KL-divergences over unobserved nodes). The number of runs in the experiment was 100. The maximum allowable number of components was \( nc = 2 \). The maximum number of iterations was \( it = 10000 \). The maximum execution time was \( max\_time = 200000 \) millisecond (ms). In the experiment, there were three exit points: (1) When the algorithm converged, (2) when the time limit was exceeded, and (3) when the algorithm diverged. When the algorithm did not converge, the algorithm halted and provided its current solution.

![Figure O.6 Percentages for each model case when the algorithm converged, diverged, and ran out of time](image)

Fig. O.6 shows percentages for each CLG BN for which the algorithm converged, diverged, and ran out of time. For CLG BN 1, the algorithm converged in 97% of the
runs and ran out of time in 3% of the cases (execution time > 200000 ms). For CLG BN 2, the algorithm converged in 52% of the runs, ran out of time in 3% of the runs, and diverged in 45% of the runs. For CLG BN 3, the algorithm converged in 100% of the runs. For CLG BN 4, the algorithm converged in only 31% of the runs and ran out of time in 69% of the runs.

We observed that a large number of cycles (CLG BN 2) could cause many situations in which the algorithm did not converge (i.e., diverged or ran out of time). Also, the algorithm for the cases with no cycles (CLG BN 3) always converged. For the cases in which the algorithm ran out of time, if more time had been allowed it might have converged, diverged, or failed to either converge or diverge (i.e., oscillated). In some cases for CLG BNs 1, 2, and 4, the algorithm oscillated until reaching the maximum execution time and halting. When there were many cycles and many discrete states (i.e., for CLG BN 4), the algorithm often ran out of time.

Table O.3 Average accuracies and average execution times for the four CLG BNs

<table>
<thead>
<tr>
<th></th>
<th>CLG BN 1</th>
<th>CLG BN 2</th>
<th>CLG BN 3</th>
<th>CLG BN 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Converged</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>avg. KL-divergence</td>
<td>0.0001 (0.0001)</td>
<td>1.0352 (1.4604)</td>
<td>2.2469 (1.452)</td>
<td>3.6409 (3.1155)</td>
</tr>
<tr>
<td>avg. Time</td>
<td>1514 (363.53)</td>
<td>16547 (24113)</td>
<td>2163.7 (415.6)</td>
<td>9584 (9739.4)</td>
</tr>
<tr>
<td>Diverged</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>avg. KL-divergence</td>
<td>-</td>
<td>106.3524 (16.0374)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>avg. Time</td>
<td>-</td>
<td>9677.7 (5703.3)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Out of time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>avg. KL-divergence</td>
<td>0.0528 (0.0545)</td>
<td>3.0795 (2.3361)</td>
<td>-</td>
<td>82.0464 (121.4148)</td>
</tr>
</tbody>
</table>
Table O.3 shows averages (avg.) for KL-divergence over runs and average execution times for three cases (converged, diverged, and ran out of time) on the four CLG BNs (numbers in parentheses are standard deviations). Fig. O.7 shows the accuracy results from this experiment, when the algorithm converged. In Fig. O.7, the four lanes denote the four CLG BNs. The vertical axis on the upper chart denotes KL-divergence values. For the case of convergence, the averages for KL-divergence over runs of the experiment were 0.0001, 1.04, 2.25, and 3.64 for CLG BNs 1, 2, 3, and 4, respectively. Again for the case of convergence, the average execution times were 1514, 16547, 2164, and 9584 for CLG BNs 1, 2, 3, and 4, respectively. For the case of convergence, because of the large number (501) of cycles in CLG BN 2, the inference algorithm required more time (avg. 16547 ms) to converge than others. Also, the algorithm for CLG BN 4 spent more time (avg. 9584 ms) in comparison with the case (avg. 2164 ms) of CLG BN 3, because of the large number (21) of cycles in CLG BN 4. In the case of divergence, the algorithm for CLG BN 2 halted with numeric underflow in the Lambda or Pi value computation. In this case, the algorithm performed with very poor accuracy (avg. 106.35) and had a long execution time (avg. 9678 ms).

Some cases for CLG BNs 1, 2, and 4 stopped because the maximum execution time was exceeded. This never happened in CLG BN 3, which contained no cycles. In addition, accuracies for CLG BNs 1 and 2 were better than those for CLG BN 4.
The four sets of results depicted in Table O.3 illustrated how network topology influences accuracy and execution time. In this experiment, we used arbitrary settings (i.e., $nc = 2$ and $it = 10000$) for HMP-GMR. In the next section, we investigate whether the performance of HMP-GMR can be improved by optimizing these settings.

**0.5.3 Optimal Settings for HMP-GMR**

HMP-GMR requires initial settings (i.e., $nc$ and $it$), which influence accuracy and execution time. To find good initial settings, the HMP-GMR-OS algorithm was introduced in Section O.4. With this algorithm, we use the following experiment setting: (1) the Hybrid BNs (i.e., CLG BNs 1, 2, 3, and 4 with $n = 7$), (2) the maximum execution time (i.e., $max\_time = 3000$ ms), (3) the number of samples (i.e., $num\_samples = 50$), (4) the upper limit on the maximum allowable number of components (i.e., $ul\_max\_nc = 10$), (5) the upper limit on the maximum number of iterations (i.e., $ul\_num\_it = 10$), and (6) the Hybrid-JT algorithm to obtain correct inference results.
Fig. O.8 shows the results from this experiment obtained by the HMP-GMR-OS algorithm. Results for CLG BNs 1, 2, 3 and 4 are shown at the top-left, top-right, bottom-left, and bottom-right, respectively. The vertical axes on the four charts denote average KL-divergence values. The bottom-left axis for each chart denotes the maximum number of iterations, while the bottom-right axis denotes the maximum allowable number of components.

Table O.4 shows minimum averages for KL-divergences in Fig. O.8 and best values for nc and it (numbers in parentheses are standard deviations). For example, for
CLG BN 1, the minimum average for KL-divergence was 0.78 and its standard deviation was 3.37 at $nc = 5$ and $it = 10$. For CLG BN 4, the minimum average for KL-divergence was 11.37 and its standard deviation was 8.44 at $nc = 1$ and $it = 8$.

For CLG BNs 1 and 3, the optimal number of iterations was the upper limit of 10. A better value might have been found if a larger number of iterations had been investigated. For CLG BNs 2 and 4, the best value for the maximum number of iterations was smaller than 10. Note that better results might be obtained, if the number of samples was increased and/or the ranges of $nc$ and/or $it$ were expanded.

From this experiment, we observed that good accuracy can be achieved with a small number of components. Although the best values found by our experiment for $nc$ were 5, 4, 6, and 1 for CLG BNs 1, 2, 3, and 4, respectively, the accuracy was not much better than using a single component. For example, the average of KL-divergence for CLG BN 1 was 0.78 at $nc = 5$ and $it = 10$, while the average of KL-divergence for CLG BN 1 was 1.09 at $nc = 1$ and $it = 10$. To check whether the difference was statistically significant, paired t-tests were performed at the 5% significance level. Table O.5 shows confidence intervals from the paired t-tests. From these tests, we observed that the

<table>
<thead>
<tr>
<th></th>
<th>CLG BN 1</th>
<th>CLG BN 2</th>
<th>CLG BN 3</th>
<th>CLG BN 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum average</td>
<td>0.7771 (3.3748)</td>
<td>15.1868 (20.999)</td>
<td>1.9617 (1.72)</td>
<td>11.3698 (8.4352)</td>
</tr>
<tr>
<td>Best $nc$</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Best $it$</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>
difference between the setting found by HMP-GMR-OS and the case with \( nc = 1 \) and \( it = 10 \) was not statistically significant for any of the CLG BNs.

<table>
<thead>
<tr>
<th>Confidence Interval</th>
<th>CLG BN 1</th>
<th>CLG BN 2</th>
<th>CLG BN 3</th>
<th>CLG BN 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-1.7623, 1.1444</td>
<td>-12.7994, 5.5235</td>
<td>-1.1426, 0.1924</td>
<td>-4.4534, 2.7875</td>
</tr>
</tbody>
</table>

This result suggests choosing \( nc = 1 \) as a default setting for HMP-GMR. This default setting for HMP-GMR was evaluated for accuracy against the Likelihood Weighting (LW) algorithm. Fig. O.9 shows accuracy comparison between HMP-GMR with the default setting \( nc = 1 \) and LW sampling for the four CLG BNs. We use the following Experiment setting: (1) type of the hybrid BNs (i.e., CLG BNs 1, 2, 3, and 4 with \( n = 7 \)), (2) type of inference algorithm (i.e., HMP-GMR at \( nc = 1 \) and \( it = 10 \), and LW sampling), (3) 100 samples, (4) the maximum execution time (i.e., \( max\_time = 3000 \) ms), and (5) the Hybrid-JT algorithm to obtain correct inference results.

Fig. O.9 shows results from this experiment. When HMP-GMR didn’t converge, it stopped at the maximum execution time and provided its current solution. The vertical axis denotes KL-divergence values. The chart contains eight lanes for four groups (CLG BNs 1, 2, 3, and 4). In the two adjoined lanes for each group, the left lane denotes the HMP-GMR case, while the right lane denotes the LW case. For example, the first lane in Fig. O.9 denotes the HMP-GMR case for CLG BN 1 (i.e., HG 1), while the second lane in Fig. O.9 denotes the LW case for CLG BN 1 (i.e., LW 1). The execution times for the
two cases in each group were set to similar values. That is, the number of samples for LW was controlled to achieve similar execution times as HMP-GMR.

![Figure O.9 KL-divergences of HMP-GMR (HG) and LW for four CLG BNs](image)

Table O.6 shows averages of KL-divergences for the two algorithms (numbers in parentheses are standard deviations). For example, for CLG BN 1, an average KL-divergence from HMP-GMR was 0.57. For CLG BN 4, an average KL-divergence from LW was 14.29. The fourth row denotes a natural-log ratio between HMP-GMR and LW. In the comparison between HMP-GMR and LW, LW was better than HMP-GMR for CLG BN 2.

![Table O.6 Comparison between three algorithms on averages of KL-divergences](table)

<table>
<thead>
<tr>
<th></th>
<th>CLG BN 1</th>
<th>CLG BN 2</th>
<th>CLG BN 3</th>
<th>CLG BN 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMP-GMR</td>
<td>0.5665 (2.4692)</td>
<td>12.224 (18.0106)</td>
<td>2.6175 (1.6649)</td>
<td>11.6846 (10.8626)</td>
</tr>
<tr>
<td>LW</td>
<td>7.8211 (4.127)</td>
<td>6.1218 (5.0357)</td>
<td>4.7952 (6.865)</td>
<td>14.2882 (24.955)</td>
</tr>
<tr>
<td>LN(HMP-GMR/LW)</td>
<td>-2.6251</td>
<td>0.6915</td>
<td>-0.6054</td>
<td>-0.2012</td>
</tr>
</tbody>
</table>
Fig. O.10 shows accuracy comparison between LW and HMP-GMR for the four CLG BNs. The X axis denotes KL-divergence for HMP-GMR. The Y axis denotes KL-divergence for LW. For CLG BN 1, HMP-GMR provided much better accuracy than LW. For CLG BN 2, LW provided better accuracy than HMP-GMR. For CLG BN 3, HMP-GMR provided better accuracy than LW. For CLG BN 4, LW and HMP-GMR performed similarly, but as can be seen in Fig. O.9, the results for LW were more variable.

Accuracy from HMP-GMR was lower in comparison with LW, for the CLG BN containing many loops. CLG BN 2 contained 501 cycles, while CLG BNs 1 and 4 contained 21 cycles and CLG BN 3 contained no cycles. More extensive investigations
would be needed to determine whether this superiority of LW over HMP-GMR generalizes to arbitrary BNs with many loops. Accuracies from HMP-GMR did not depend much on the number of discrete states. CLG BN 1 contained four discrete states, while the CLG BN 3 contained 16384 configurations of the discrete states. For these networks, HMP-GMR provided better accuracy than LW regardless of how many discrete states a CLG BN contains.

We can consider whether to use HMP-GMR or LW according to the features of the CLG BN. The following list shows suggestions in which HMP-GMR can be chosen or not in terms of the number of configurations of the discrete states and the number of cycles.

**0.5.3.1 Small number of configurations of the discrete states and small number of cycles**

In this case, our experiment showed better accuracy from HMP-GMR in comparison with LW under a given time restriction. LW requires many samples to improve accuracy, while HMP-GMR uses message passing approach, which can provide exact results for a polytree network [Pearl, 1988]. In a simple-topology network (i.e., small number of configurations of the discrete states and small number of cycles), when HMP-GMR converges in time, it can provide high accuracy.

**0.5.3.2 Small number of configurations of the discrete states and large number of cycles**

When there are many cycles, HMP-GMR can diverge. This behavior depends on the network topology, the placement of continuous and discrete nodes, the conditional distributions, and the pattern of evidence. When divergence occurs, HMP-GMR halts
during message passing because of numeric underflow. A feature to detect when the pi and lambda messages are going out of bounds and stop the algorithm may be useful, but intermediate results from before the algorithm diverges are of doubtful usefulness. In the case of a large number of cycles, LW can perform better than HMP-GMR.

**0.5.3.3 Large number of configurations of the discrete states and small number of cycles**

HMP-GMR reduces the maximum allowable number of components, which is influenced by the number of configurations of the discrete states. So, HMP-GMR can tolerate many numbers of configurations of the discrete states, while LW requires many samples for many numbers of configurations of the discrete states. In this case, we observed that for the four CLG BNs, HMP-GMR could converge and provide good accuracy.

**0.5.3.4 Large number of configurations of the discrete states and large number of cycles**

Although LW can tolerate many cycles, it can perform poorly when the necessary number of samples to achieve good accuracy is too large for the available time limit. Also, HMP-GMR can perform poorly because of many cycles. In this case, the better choice between LW and HMP-GMR may vary depending on specific features of the problem. For example, when the number of configurations of the discrete states is relatively smaller than the number of cycles, LW can be used. When the number of configurations of the discrete states is relatively larger than the number of cycles, HMP-GMR can be used. However, in any case, accuracies for both approaches may be low.
**0.6 Conclusion**

We have developed an extended message passing algorithm for CG hybrid Bayesian networks to overcome the exponential growth in components of the Gaussian mixture model. Our experiments demonstrated scalability, accuracy, and optimal settings for the complex hybrid BNs. In our experiments, both the original hybrid message passing inference and the hybrid junction tree inference showed exponential growth in execution time. The Gaussian mixture reduction method presented in this chapter addressed this problem. Another issue we should address was to define ways to choose the optimal settings to achieve desired accuracy and execution time. For this, we presented a preprocessing algorithm to optimize the maximum allowable number of components and the optimal maximum iteration. The algorithm enables HMP-GMR to provide better accuracy within a predefined time. For the four CLG BNs we investigated, we observed that the accuracy from using a single Gaussian component was nearly as good as the setting found by our optimization method, and the difference in accuracy was not statistically significant. Note that other networks with complex topologies, very unlikely evidence configurations, and/or deterministic or near-deterministic relationships might be different. We observed that the accuracy results for a loopy CG BN were less than for a poly CG BN. To address this, we can use a clustering inference (e.g., Hybrid-JT) with Gaussian mixture reduction. These issues will be addressed in future work.
Appendix P. Test HBN for HMP-GMR

**CLG BN 1 (n = 5)**

```plaintext
defineNode(A, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4); p(A) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X1, DescriptionY);{ defineState(Continuous); p(X1 | A) = if (A == a1) {NormalDist(0, 100); } else if (A == a2) {NormalDist(200, 100); } else if (A == a3) {NormalDist(400, 100); } else if (A == a4) {NormalDist(600, 100); } }
defineNode(Y1, DescriptionY);{ defineState(Continuous); p(Y1 | X1) = X1 + NormalDist(0, 100); }
defineNode(X2, DescriptionY);{ defineState(Continuous); p(X2 | X1, A) = if (A == a1) {2* X1 + NormalDist(0, 100); } else if (A == a2) {2* X1 + NormalDist(200, 100); } else if (A == a3) {2* X1 + NormalDist(400, 100); } else if (A == a4) {2* X1 + NormalDist(600, 100); } }
defineNode(Y2, DescriptionY);{ defineState(Continuous); p(Y2 | X2) = X2 + NormalDist(0, 100); }
defineNode(X3, DescriptionY);{ defineState(Continuous); p(X3 | X2, A) = if (A == a1) {2* X2 + NormalDist(0, 100); } else if (A == a2) {2* X2 + NormalDist(200, 100); } else if (A == a3) {2* X2 + NormalDist(400, 100); } else if (A == a4) {2* X2 + NormalDist(600, 100); } }
defineNode(Y3, DescriptionY);{ defineState(Continuous); p(Y3 | X3) = X3 + NormalDist(0, 100); }
defineNode(X4, DescriptionY);{ defineState(Continuous); p(X4 | X3, A) = if (A == a1) {2* X3 + NormalDist(0, 100); } else if (A == a2) {2* X3 + NormalDist(200, 100); } else if (A == a3) {2* X3 + NormalDist(400, 100); } else if (A == a4) {2* X3 + NormalDist(600, 100); } }
defineNode(Y4, DescriptionY);{ defineState(Continuous); p(Y4 | X4) = X4 + NormalDist(0, 100); }
defineNode(X5, DescriptionY);{ defineState(Continuous); p(X5 | X4, A) = if (A == a1) {2* X4 + NormalDist(0, 100); } else if (A == a2) {2* X4 + NormalDist(200, 100); } else if (A == a3) {2* X4 + NormalDist(400, 100); } else if (A == a4) {2* X4 + NormalDist(600, 100); } }
defineNode(Y5, DescriptionY);{ defineState(Continuous); p(Y5 | X5) = X5 + NormalDist(0, 100); }
```

**CLG BN 2 (n = 5)**

```plaintext
defineNode(A, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4); p(A) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X1, DescriptionY);{ defineState(Continuous); p(X1 | A) = if (A == a1) {NormalDist(0, 100); } else if (A == a2) {NormalDist(200, 100); } else if (A == a3) {NormalDist(400, 100); } else if (A == a4) {NormalDist(600, 100); } }
defineNode(Y1, DescriptionY);{ defineState(Continuous); p(Y1 | X1) = X1 + NormalDist(0, 100); }
defineNode(X2, DescriptionY);{ defineState(Continuous); p(X2 | X1, A) = if (A == a1) {2* X1 + NormalDist(0, 100); } else if (A == a2) {2* X1 + NormalDist(200, 100); } else if (A == a3) {2* X1 + NormalDist(400, 100); } else if (A == a4) {2* X1 + NormalDist(600, 100); } }
defineNode(Y2, DescriptionY);{ defineState(Continuous); p(Y2 | X2) = X2 + NormalDist(0, 100); }
defineNode(X3, DescriptionY);{ defineState(Continuous); p(X3 | X2, A) = if (A == a1) {2* X2 + NormalDist(0, 100); } else if (A == a2) {2* X2 + NormalDist(200, 100); } else if (A == a3) {2* X2 + NormalDist(400, 100); } else if (A == a4) {2* X2 + NormalDist(600, 100); } }
defineNode(Y3, DescriptionY);{ defineState(Continuous); p(Y3 | X3) = X3 + NormalDist(0, 100); }
defineNode(X4, DescriptionY);{ defineState(Continuous); p(X4 | X3, A) = if (A == a1) {2* X3 + NormalDist(0, 100); } else if (A == a2) {2* X3 + NormalDist(200, 100); } else if (A == a3) {2* X3 + NormalDist(400, 100); } else if (A == a4) {2* X3 + NormalDist(600, 100); } }
defineNode(Y4, DescriptionY);{ defineState(Continuous); p(Y4 | X4) = X4 + NormalDist(0, 100); }
defineNode(X5, DescriptionY);{ defineState(Continuous); p(X5 | X4, A) = if (A == a1) {2* X4 + NormalDist(0, 100); } else if (A == a2) {2* X4 + NormalDist(200, 100); } else if (A == a3) {2* X4 + NormalDist(400, 100); } else if (A == a4) {2* X4 + NormalDist(600, 100); } }
defineNode(Y5, DescriptionY);{ defineState(Continuous); p(Y5 | X5) = X5 + NormalDist(0, 100); }
```
else if (A == a4) { 2* X1 + NormalDist(600, 100); } }
defineNode(Y2, DescriptionY);{ defineState(Continuous); p( Y2 | Y1, X2 ) = 2*Y1+ X2 + NormalDist(0, 100); }
defineNode(X3, DescriptionY);{ defineState(Continuous); p( X3 | X2, A ) = if (A == a1) { 2* X2 + NormalDist(0, 100); } else if (A == a2) { 2* X2 + NormalDist(200, 100); } else if (A == a3) { 2* X2 + NormalDist(400, 100); } else if (A == a4) { 2* X2 + NormalDist(600, 100); } }
defineNode(Y3, DescriptionY);{ defineState(Continuous); p( Y3 | Y2, X3 ) = 2*Y2+ X3 + NormalDist(0, 100); }
defineNode(X4, DescriptionY);{ defineState(Continuous); p( X4 | X3, A ) = if (A == a1) { 2* X3 + NormalDist(0, 100); } else if (A == a2) { 2* X3 + NormalDist(200, 100); } else if (A == a3) { 2* X3 + NormalDist(400, 100); } else if (A == a4) { 2* X3 + NormalDist(600, 100); } }
defineNode(Y4, DescriptionY);{ defineState(Continuous); p( Y4 | Y3, X4 ) = 2*Y3+ X4 + NormalDist(0, 100); }
defineNode(X5, DescriptionY);{ defineState(Continuous); p( X5 | X4, A ) = if (A == a1) { 2* X4 + NormalDist(0, 100); } else if (A == a2) { 2* X4 + NormalDist(200, 100); } else if (A == a3) { 2* X4 + NormalDist(400, 100); } else if (A == a4) { 2* X4 + NormalDist(600, 100); } }
defineNode(Y5, DescriptionY);{ defineState(Continuous); p( Y5 | Y4, X5 ) = 2*Y4+ X5 + NormalDist(0, 100); }

CLG BN 3 (n = 5)
defineNode(A, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4); p( A ) = { a1: 0.1; a2: 0.2; a3: 0.3; a4: 0.4; } }
defineNode(X1, DescriptionY);{ defineState(Continuous); p( X1 | A ) = if (A == a1) {NormalDist(0, 100); } else if (A == a2) {NormalDist(200, 100); } else if (A == a3) {NormalDist(400, 100); } else if (A == a4) {NormalDist(600, 100); } }
defineNode(Y1, DescriptionY);{ defineState(Continuous); p( Y1 | X1 ) = X1 + NormalDist(0, 100); }
defineNode(A2, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4); p( A2 ) = { a1: 0.1; a2: 0.2; a3: 0.3; a4: 0.4; } }
defineNode(X2, DescriptionY);{ defineState(Continuous); p( X2 | X1, A2 ) = if (A2 == a1) {2*X1 + NormalDist(0, 100); } else if (A2 == a2) {2*X1 + NormalDist(200, 100); } else if (A2 == a3) {2*X1 + NormalDist(400, 100); } else if (A2 == a4) {2*X1 + NormalDist(600, 100); } }
defineNode(Y2, DescriptionY);{ defineState(Continuous); p( Y2 | X2 ) = X2 + NormalDist(0, 100); }
defineNode(A3, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4); p( A3 ) = { a1: 0.1; a2: 0.2; a3: 0.3; a4: 0.4; } }
defineNode(X3, DescriptionY);{ defineState(Continuous); p( X3 | X2, A3 ) = if (A3 == a1) {2*X2 + NormalDist(0, 100); } else if (A3 == a2) {2*X2 + NormalDist(200, 100); } else if (A3 == a3) {2*X2 + NormalDist(400, 100); } else if (A3 == a4) {2*X2 + NormalDist(600, 100); } }
defineNode(Y3, DescriptionY);{ defineState(Continuous); p( Y3 | X3 ) = X3 + NormalDist(0, 100); }
defineNode(A4, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4); p( A4 ) = { a1: 0.1; a2: 0.2; a3: 0.3; a4: 0.4; } }
defineNode(X4, DescriptionY);{ defineState(Continuous); p( X4 | X3, A4 ) = if (A4 == a1) {2*X3 + NormalDist(0, 100); } else if (A4 == a2) {2*X3 + NormalDist(200, 100); }
else if (A4 == a3) {  2*X3 + NormalDist(400, 100); }
else if (A4 == a4) {  2*X3 + NormalDist(600, 100); }
defineNode(Y4, DescriptionY);{ defineState(Continuous);
p( Y4 |  X4 ) = X4 + NormalDist(0, 100); }
defineNode(A5, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4);
p( A5 ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X5, DescriptionY);{ defineState(Continuous);
p( X5 |  X4, A5 ) = if (A5 == a1) {  2*X4 + NormalDist(0, 100); }
else if (A5 == a2) {  2*X4 + NormalDist(200, 100); }
else if (A5 == a3) {  2*X4 + NormalDist(400, 100); }
else if (A5 == a4) {  2*X4 + NormalDist(600, 100); }
defineNode(Y5, DescriptionY);{ defineState(Continuous);
p( Y5 |  X5 ) = X5 + NormalDist(0, 100); }

CLG BN 4  (n = 5)

defineNode(A, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4);
p( A ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X1, DescriptionY);{ defineState(Continuous);
p( X1 | A ) = if { A == a1 } {NormalDist(0, 100); }
else if (A == a2 ) {NormalDist(200, 100); }
else if ( A == a3 ) {NormalDist(400, 100); }
else if ( A == a4 ) {NormalDist(600, 100); }
defineNode(Y1, DescriptionY);{ defineState(Continuous);
p( Y1 | X1) = X1 + NormalDist(0, 100); }
defineNode(A2, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4);
p( A2 ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X2, DescriptionY);{ defineState(Continuous);
p( X2 |  X1, A2 ) = if (A2 == a1) {  2*X1 + NormalDist(0, 100); }
else if (A2 == a2) {  2*X1 + NormalDist(200, 100); }
else if (A2 == a3) {  2*X1 + NormalDist(400, 100); }
else if (A2 == a4) {  2*X1 + NormalDist(600, 100); }
defineNode(Y2, DescriptionY);{ defineState(Continuous);
p( Y2 |  Y1, X2 ) = 2*Y1+ X2 + NormalDist(0, 100); }
defineNode(A3, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4);
p( A3 ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X3, DescriptionY);{ defineState(Continuous);
p( X3 |  X2, A3 ) = if (A3 == a1) {  2*X2 + NormalDist(0, 100); }
else if (A3 == a2) {  2*X2 + NormalDist(200, 100); }
else if (A3 == a3) {  2*X2 + NormalDist(400, 100); }
else if (A3 == a4) {  2*X2 + NormalDist(600, 100); }
defineNode(Y3, DescriptionY);{ defineState(Continuous);
p( Y3 |  Y2, X3 ) = 2*Y2+ X3 + NormalDist(0, 100); }
defineNode(A4, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4);
p( A4 ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X4, DescriptionY);{ defineState(Continuous);
p( X4 |  X3, A4 ) = if (A4 == a1) {  2*X3 + NormalDist(0, 100); }
else if (A4 == a2) {  2*X3 + NormalDist(200, 100); }
else if (A4 == a3) {  2*X3 + NormalDist(400, 100); }
else if (A4 == a4) {  2*X3 + NormalDist(600, 100); }
defineNode(Y4, DescriptionY);{ defineState(Continuous);
p( Y4 |  Y3, X4 ) = 2*Y3+ X4 + NormalDist(0, 100); }
defineNode(A5, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4);
p( A5 ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X5, DescriptionY);{ defineState(Continuous);
p( X5 |  X4, A5 ) = if (A5 == a1) {  2*X4 + NormalDist(0, 100); }
else if (A5 == a2) {  2*X4 + NormalDist(200, 100); }
else if (A5 == a3) {  2*X4 + NormalDist(400, 100); }
else if (A5 == a4) {  2*X4 + NormalDist(600, 100); }
}
defineNode(Y5, DescriptionY);{ defineState(Continuous); p( Y5 | Y4, X5 ) = 2*Y4 + X5 + NormalDist(0, 100); }

CNG BN 1 (n = 5)

defineNode(A, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4); p( A ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X1, DescriptionY);{ defineState(Continuous); p( X1 | A ) = if ( A == a1 ) {NormalDist(100, 100); } else if ( A == a2 ) {NormalDist(200, 100); } else if ( A == a3 ) {NormalDist(400, 100); } else if ( A == a4 ) {NormalDist(600, 100); } }
defineNode(Y1, DescriptionY);{ defineState(Continuous); p( Y1 | X1 ) = X1 + NormalDist(10, 100); }
defineNode(X2, DescriptionY);{ defineState(Continuous); p( X2 | X1, A ) = if (A == a1) { Log(X1, e) + NormalDist(100, 100); } else if (A == a2) { Log(X2, e) + NormalDist(200, 100); } else if (A == a3) { Log(X2, e) + NormalDist(400, 100); } else if (A == a4) { Log(X2, e) + NormalDist(600, 100); } }
defineNode(Y2, DescriptionY);{ defineState(Continuous); p( Y2 | X2 ) = X2 + NormalDist(10, 100); }
defineNode(X3, DescriptionY);{ defineState(Continuous); p( X3 | X2, A ) = if (A == a1) { Log(X2, e) + NormalDist(100, 100); } else if (A == a2) { Log(X3, e) + NormalDist(200, 100); } else if (A == a3) { Log(X3, e) + NormalDist(400, 100); } else if (A == a4) { Log(X3, e) + NormalDist(600, 100); } }
defineNode(Y3, DescriptionY);{ defineState(Continuous); p( Y3 | X3 ) = X3 + NormalDist(10, 100); }
defineNode(X4, DescriptionY);{ defineState(Continuous); p( X4 | X3, A ) = if (A == a1) { Log(X3, e) + NormalDist(100, 100); } else if (A == a2) { Log(X4, e) + NormalDist(200, 100); } else if (A == a3) { Log(X4, e) + NormalDist(400, 100); } else if (A == a4) { Log(X4, e) + NormalDist(600, 100); } }
defineNode(Y4, DescriptionY);{ defineState(Continuous); p( Y4 | X4 ) = X4 + NormalDist(10, 100); }
defineNode(X5, DescriptionY);{ defineState(Continuous); p( X5 | X4, A ) = if (A == a1) { Log(X4, e) + NormalDist(100, 100); } else if (A == a2) { Log(X5, e) + NormalDist(200, 100); } else if (A == a3) { Log(X5, e) + NormalDist(400, 100); } else if (A == a4) { Log(X5, e) + NormalDist(600, 100); } }
defineNode(Y5, DescriptionY);{ defineState(Continuous); p( Y5 | X5 ) = X5 + NormalDist(10, 100); }

CNG BN 2 (n = 5)

defineNode(A, DescriptionY);{ defineState(Discrete, a1, a2, a3, a4); p( A ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X1, DescriptionY);{ defineState(Continuous); p( X1 | A ) = if ( A == a1 ) {NormalDist(100, 100); } else if ( A == a2 ) {NormalDist(200, 100); } else if ( A == a3 ) {NormalDist(400, 100); } else if ( A == a4 ) {NormalDist(600, 100); } }
defineNode(Y1, DescriptionY);{ defineState(Continuous); p( Y1 | X1 ) = X1 + NormalDist(10, 100); }

390
defineNode(X2, DescriptionY); { defineState(Continuous);
  p(x2 | x1, A ) = if (A == a1) {Log(X1, e) + NormalDist(100, 100);}
  else if (A == a2) {Log(X1, e) + NormalDist(200, 100);}
  else if (A == a3) {Log(X1, e) + NormalDist(400, 100);}
  else if (A == a4) {Log(X1, e) + NormalDist(600, 100);}
}
defineNode(Y2, DescriptionY); { defineState(Continuous);
  p(y2 | y1, x2) = 2*y1 + x2 + NormalDist(10, 100);
}
defineNode(X3, DescriptionY); { defineState(Continuous);
  p(x3 | x2, A ) = if (A == a1) {Log(X2, e) + NormalDist(100, 100);}
  else if (A == a2) {Log(X2, e) + NormalDist(200, 100);}
  else if (A == a3) {Log(X2, e) + NormalDist(400, 100);}
  else if (A == a4) {Log(X2, e) + NormalDist(600, 100);}
}
defineNode(Y3, DescriptionY); { defineState(Continuous);
  p(y3 | y2, x3) = 2*y2 + x3 + NormalDist(10, 100);
}
defineNode(X4, DescriptionY); { defineState(Continuous);
  p(x4 | x3, A ) = if (A == a1) {Log(X3, e) + NormalDist(100, 100);}
  else if (A == a2) {Log(X3, e) + NormalDist(200, 100);}
  else if (A == a3) {Log(X3, e) + NormalDist(400, 100);}
  else if (A == a4) {Log(X3, e) + NormalDist(600, 100);}
}
defineNode(Y4, DescriptionY); { defineState(Continuous);
  p(y4 | y3, x4) = 2*y3 + x4 + NormalDist(10, 100);
}
defineNode(X5, DescriptionY); { defineState(Continuous);
  p(x5 | x4, A ) = if (A == a1) {Log(X4, e) + NormalDist(100, 100);}
  else if (A == a2) {Log(X4, e) + NormalDist(200, 100);}
  else if (A == a3) {Log(X4, e) + NormalDist(400, 100);}
  else if (A == a4) {Log(X4, e) + NormalDist(600, 100);}
}
defineNode(Y5, DescriptionY); { defineState(Continuous);
  p(y5 | y4, x5) = 2*y4 + x5 + NormalDist(10, 100);
}

defineNode(A, DescriptionY); { defineState(Discrete, a1, a2, a3, a4);
  p(A ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; }
}
defineNode(X1, DescriptionY); { defineState(Continuous);
  p(x1 | A ) = if (A == a1) {NormalDist(100, 100);}
  else if (A == a2) {NormalDist(200, 100);}
  else if (A == a3) {NormalDist(400, 100);}
  else if (A == a4) {NormalDist(600, 100);}
}
defineNode(Y1, DescriptionY); { defineState(Continuous);
  p(y1 | x1) = x1 + NormalDist(10, 100);
}
defineNode(A2, DescriptionY); { defineState(Discrete, a1, a2, a3, a4);
  p(A2 ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; }
}
defineNode(X2, DescriptionY); { defineState(Continuous);
  p(x2 | x1, A2 ) = if (A2 == a1) {Log(X1, e) + NormalDist(100, 100);}
  else if (A2 == a2) {Log(X1, e) + NormalDist(200, 100);}
  else if (A2 == a3) {Log(X1, e) + NormalDist(400, 100);}
  else if (A2 == a4) {Log(X1, e) + NormalDist(600, 100);}
}
defineNode(Y2, DescriptionY); { defineState(Continuous);
  p(y2 | x2) = x2 + NormalDist(10, 100);
}
defineNode(A3, DescriptionY); { defineState(Discrete, a1, a2, a3, a4);
  p(A3 ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; }
}
defineNode(X3, DescriptionY); { defineState(Continuous);
  p(x3 | A3 ) = if (A3 == a1) {Log(X2, e) + NormalDist(100, 100);}
  else if (A3 == a2) {Log(X2, e) + NormalDist(200, 100);}
  else if (A3 == a3) {Log(X2, e) + NormalDist(400, 100);}
  else if (A3 == a4) {Log(X2, e) + NormalDist(600, 100);}
}
defineNode(Y3, DescriptionY); { defineState(Continuous);
  p(y3 | x3) = x3 + NormalDist(10, 100);
}
defineNode(A4, DescriptionY); { defineState(Discrete, a1, a2, a3, a4);
p(A4 ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X4, DescriptionY); { defineState(Continuous);
p(X4 | X3, A4 ) = if (A4 == a1) { Log(X3, e) + NormalDist(100, 100); }
ext else if (A4 == a2) { Log(X3, e) + NormalDist(200, 100); }
ext else if (A4 == a3) { Log(X3, e) + NormalDist(400, 100); }
ext else if (A4 == a4) { Log(X3, e) + NormalDist(600, 100); } }
defineNode(Y4, DescriptionY); { defineState(Continuous);
p(Y4 | X4 ) = X4 + NormalDist(10, 100); }
defineNode(A5, DescriptionY); { defineState(Discrete, a1, a2, a3, a4);
p(A5 ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X5, DescriptionY); { defineState(Continuous);
p(X5 | X4, A5 ) = if (A5 == a1) { Log(X4, e) + NormalDist(100, 100); }
ext else if (A5 == a2) { Log(X4, e) + NormalDist(200, 100); }
ext else if (A5 == a3) { Log(X4, e) + NormalDist(400, 100); }
ext else if (A5 == a4) { Log(X4, e) + NormalDist(600, 100); } }
defineNode(Y5, DescriptionY); { defineState(Continuous);
p(Y5 | X5 ) = X5 + NormalDist(10, 100); }

CNG BN 4 (n = 5)
defineNode(A, DescriptionY); { defineState(Discrete, a1, a2, a3, a4);
p(A ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X1, DescriptionY); { defineState(Continuous);
p(X1 | A ) = if (A == a1) { NormalDist(100, 100); }
ext else if (A == a2) { NormalDist(200, 100); }
ext else if (A == a3) { NormalDist(400, 100); }
ext else if (A == a4) { NormalDist(600, 100); } }
defineNode(Y1, DescriptionY); { defineState(Continuous);
p(Y1 | X1 ) = X1 + NormalDist(10, 100); }
defineNode(A2, DescriptionY); { defineState(Discrete, a1, a2, a3, a4);
p(A2 ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X2, DescriptionY); { defineState(Continuous);
p(X2 | X1, A2 ) = if (A2 == a1) { Log(X1, e) + NormalDist(100, 100); }
ext else if (A2 == a2) { Log(X1, e) + NormalDist(200, 100); }
ext else if (A2 == a3) { Log(X1, e) + NormalDist(400, 100); }
ext else if (A2 == a4) { Log(X1, e) + NormalDist(600, 100); } }
defineNode(Y2, DescriptionY); { defineState(Continuous);
p(Y2 | X1, X2 ) = 2*Y1+ X2 + NormalDist(10, 100); }
defineNode(A3, DescriptionY); { defineState(Discrete, a1, a2, a3, a4);
p(A3 ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X3, DescriptionY); { defineState(Continuous);
p(X3 | X2, A3 ) = if (A3 == a1) { Log(X2, e) + NormalDist(100, 100); }
ext else if (A3 == a2) { Log(X2, e) + NormalDist(200, 100); }
ext else if (A3 == a3) { Log(X2, e) + NormalDist(400, 100); }
ext else if (A3 == a4) { Log(X2, e) + NormalDist(600, 100); } }
defineNode(Y3, DescriptionY); { defineState(Continuous);
p(Y3 | X2, X3 ) = 2*Y2+ X3 + NormalDist(10, 100); }
defineNode(A4, DescriptionY); { defineState(Discrete, a1, a2, a3, a4);
p(A4 ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X4, DescriptionY); { defineState(Continuous);
p(X4 | X3, A4 ) = if (A4 == a1) { Log(X3, e) + NormalDist(100, 100); }
ext else if (A4 == a2) { Log(X3, e) + NormalDist(200, 100); }
ext else if (A4 == a3) { Log(X3, e) + NormalDist(400, 100); }
ext else if (A4 == a4) { Log(X3, e) + NormalDist(600, 100); } }
defineNode(Y4, DescriptionY); { defineState(Continuous);
p(Y4 | X3, X4 ) = 2*Y3+ X4 + NormalDist(10, 100); }
defineNode(A5, DescriptionY); { defineState(Discrete, a1, a2, a3, a4);
p(A5 ) = { a1 : 0.1; a2 : 0.2; a3 : 0.3; a4 : 0.4; } }
defineNode(X5, DescriptionY); { defineState(Continuous);
    p( X5 | X4, A5 ) = if (A5 == a1) { Log(X4, e) + NormalDist(100, 100); } 
    else if (A5 == a2) { Log(X4, e) + NormalDist(200, 100); } 
    else if (A5 == a3) { Log(X4, e) + NormalDist(400, 100); } 
    else if (A5 == a4) { Log(X4, e) + NormalDist(600, 100); } 
} defineNode(Y5, DescriptionY); { defineState(Continuous);
    p( Y5 | Y4, X5 ) = 2*Y4 + X5 + NormalDist(10, 100); 
} defineNode(Y5, DescriptionY); { defineState(Continuous);
    p( Y5 | Y4, X5 ) = 2*Y4 + X5 + NormalDist(10, 100); 
}
REFERENCES


396


Date, C. J. (2012). Database Design and Relational Theory: Normal Forms and All That Jazz. O'Reilly Media, Inc.


Hackthorn, R. (2003). The Link is the Thing. DM Review


Hustadt, U. (1994). Do we need the closed world assumption in knowledge representation?. In KRDB.


403


Motion Imagery Standards Board (MISB). (2010). Video Moving Target Indicator Local Data Set. MISB Recommended Practice 0903.2.


Vagia, M., Transeth, A. A., & Fjerdingen, S. A. (2016). A literature review on the levels of automation during the years. What are the different taxonomies that have been proposed?. Applied ergonomics, 53, 190-202.


Williams, J. L. (2003). Gaussian mixture reduction for tracking multiple maneuvering targets in clutter (no. afit/ge/eng/03-19). Air Force Inst of Tech Wright-Patterson AFB OH School of Engineering and Management.


Cheol Young Park served at South Korean Air Force for 3 years. After leaving the service, he was hired at a software company in 1998. For five years after 1998, he had worked on software development (e.g., web browsers, messengers, and games). From his industrial experience, he has focused on systems engineering research to find solutions to develop a system efficiently and improve system performance using operational research and systems engineering. To do that, he joined Information Technology and System Engineering program at GMU (2008). Since 2008, he focused on a research on automatic system modeling under uncertainty from data. He has published 13 peer-reviewed papers related to his research goal. In his PhD program, he has developed representation formalisms, inference algorithms, and machine learning algorithms for MEBN (Multi-Entity Bayesian Networks) to support predictive situation awareness. He served as a reviewer for IEEE Transactions on Cybernetics and he volunteers to teach artificial intelligence and software programming to high school students. His PhD research has been funded by the Office of Naval Research and the Ministry of Trade, Industry & Energy in the Republic of Korea. His research results at GMU include the following: (1) He developed a smart manufacturing system, called MSAW (Manufacturing Predictive Situation Awareness), which aims to analyze, predict, and optimize manufacturing factors. MSAW has been applied to a production line for a steel plate manufacturing factory. (2) He developed a process for Human-aided MEBN learning that is a framework to develop a MEBN model from a domain expert’s knowledge combined with data. With this framework, he developed a reference MEBN model which supports the design of a MEBN model for predictive situation awareness (PSAW). In addition, he developed a MEBN-RM model which is a mapping method that specifies how to convert elements of the Relational Model to elements of MEBN. He also developed a MEBN learning algorithm for conditional linear Gaussian models. (3) He developed a proof of concept system for predictive situation awareness, called HERALD, which was designed to ward off attacks against critical infrastructure by means of early detection of threatening targets, identification of the targets, estimation of the target’s activities, and prediction of virtual short-term future situations. (4) He developed an agent-based simulation for probabilistic ontologies to distributed predictive situation assessment in naval operations system (PROGNOS), designed to predict future situations in the maritime domain using social network and geospatial information. (5) He developed several JAVA programs for Bayesian Network inference algorithms (e.g., Junction Tree (JT), Direct Message Passing (DMP), and Likelihood Weighting (LW)) capable of accommodating hybrid discrete and continuous Bayesian networks. At GMU, he has published the following papers with his colleagues.


