LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TR-631

An Integrated Approach to Dynamic Decision Making under Uncertainty

Tze-Yun Leong

August, 1994

This document has been made available free of charge via ftp from the
MIT Laboratory for Computer Science.

545 TECHNOLOGY SQUARE; CAMBRIDGE, MASSACHUSETTS 02139 (617) 253-5851

An Integrated Approach to Dynamic Decision Making under Uncertainty

by

Tze-Yun Leong

August, 1994

© Massachusetts Institute of Technology, 1994

An Integrated Approach to Dynamic Decision Making under Uncertainty

by

Tze-Yun Leong

This report is a modified version of a thesis submitted to the
Department of Electrical Engineering and Computer Science on August 16, 1994,
in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

Decision making is often complicated by the dynamic and uncertain information involved. This work unifies and generalizes the major approaches to modeling and solving a subclass of such decision problems. The relevant problem characteristics include discrete problem parameters, separable optimality functions, and sequential decisions made in stages. The relevant approaches include semi-Markov decision processes, dynamic decision modeling, and decision-theoretic planning.

An analysis of current decision frameworks establishes a unifying task definition and a common vocabulary; the exercise also identifies the trade-off between model transparency and solution efficiency as their most significant limitation.

Insights gained from the analysis lead to a new methodology for dynamic decision making under uncertainty. The central ideas involved are *multiple perspective reasoning* and *incremental language extension*. Multiple perspective reasoning supports different information visualization formats for different aspects of dynamic decision modeling. Incremental language extension promotes the use of translators to enhance language ontology and facilitate practical development.

DynaMoL is a language design that adopts the proposed paradigm; it differentiates inferential and representational support for the modeling task from the solution or computation task. The dynamic decision grammar defines an extensible decision ontology and supports problem specification with multiple interfaces. The graphical presentation convention governs parameter visualization in multiple perspectives. The mathematical representation as semi-Markov decision process facilitates formal model analysis and admits multiple solution methods. A general translation technique is devised for the different perspectives and representations of the decision factors and constraints.

DynaMoL is evaluated on a prototype implementation, via a comprehensive case study in medicine. The case study is based on an actual treatment planning decision for a patient with atrial fibrillation. The problems addressed include both long-term discrimination and short-term optimization of different treatment strategies. The results demonstrate practical promise of the framework.

Thesis supervisor: Peter Szolovits
Title: Professor of Computer Science and Engineering

Acknowledgments

I would like to thank:

Professor Peter Szolovits, my thesis supervisor and mentor, for his guidance, encouragement, and support throughout my graduate career. I would not have made it through without his patience and belief in me.

Dr. Stephen G. Pauker, my thesis reader, for teaching me about medical decision analysis in practice, and for providing me with the opportunity to learn from the Division of Clinical Decision Making at Tufts-New England Medical Center.

Professor Alvin Drake, my other thesis reader, for his encouragement, enthusiasm, and for teaching me about precise and effective presentation of technical ideas.

Dr. Charles E. Ellis, for helping me with the case study; he has been a great teacher and domain expert.

Dr. William J. Long, my undergraduate thesis supervisor, and Professor Ramesh Patil, my undergraduate academic counsellor, for introducing me into the research field on which I shall build my professional career, and for being very supportive and encouraging over the years.

Professor Harold Abelson for his encouragement at some critical points of my graduate studies, and for inspiring me to be a computer scientist and a teacher.

Dr. Jon Doyle for providing me with financial support.

Past and present members of the Clinical Decision Making Group at the MIT Laboratory for Computer Science, for their friendships and camaraderie, and for providing me with an intellectually stimulating environment. Ira Haimowitz and Yeona Jang have been my good friends and colleagues through my undergraduate and graduate student life. Michael Wellman introduced me to automated decision analysis and has continued to be very supportive, encouraging, and enthusiastic in my work. My current office mate, Christine Tsien, has always made our office a pleasant place; she has been most understanding when I monopolized the office during the final phase of my thesis.

Members of the Division of Clinical Decision Making at Tufts-New England Medical Center, especially Dr. John Wong and Dr. Mark Eckman, for teaching me about medical decision analysis, and Dr. Brian Kan, for his friendship and many interesting discussions.

All my friends who have helped and brightened my life over the years at MIT, especially Boon Seong Ang, Boon Hwee Chew, Hui Meng Chow, Heidi Danziger,

Choon Phong Goh, Meng Yong Goh, Shail Gupta, Beng Hong Lim, Janice McMahon, Kah Kay Sung, and Wai Khum Wong.

My parents, for their unbounded love and everlasting belief in me, and for their patience during my long absence from home.

My husband, Yang Meng Tan, for sharing the joy and suffering through graduate school with me, for being a friend and a colleague, and most important of all, for his love.

This research was supported by the National Institutes of Health Grant No. 5 R01 LM04493 from the National Library of Medicine, and by the USAF Rome Laboratory and DARPA under contract F30602-91-C-0018.

Contents

Acknowledgments 5

List of Figures 11

List of Tables 13

1 Introduction 15

1.1 Background and Motivations . . . 15
1.2 The Dynamic Decision Problem . . . 16
1.3 The Application Domain . . . 18
1.4 Research Objectives and Approaches . . . 18
1.4.1 A Uniform Task Definition . . . 19
1.4.2 A Common Vocabulary . . . 19
1.4.3 A General Methodology . . . 19
1.4.4 A Prototype Implementation . . . 20
1.4.5 A Case Study . . . 20
1.5 Contributions . . . 20
1.6 Report Overview . . . 21

2 A Survey of Current Approaches 23

2.1 Historical Background . . . 23
2.1.1 Markov and Semi-Markov Decision Processes . . . 23
2.1.2 Dynamic Decision Modeling . . . 24
2.1.3 Planning in Artificial Intelligence . . . 24
2.1.4 Recent Development . . . 25
2.2 An Example . . . 25
2.3 Semi-Markov Decision Processes . . . 26
2.4 Dynamic Decision Modeling . . . 34
2.4.1 Dynamic Influence Diagrams . . . 34
2.4.2 Markov Cycle Trees . . . 37
2.4.3 Stochastic Trees . . . 41
2.5 Planning in Artificial Intelligence . . . 42
2.6 Summary . . . 44

3 A Unifying View 45

3.1 A Uniform Task Definition . . . 45
3.1.1 Problem Formulation . . . 45
3.1.2 Problem Solution . . . 47
3.1.3 Problem Analysis . . . 47
3.2 A Common Vocabulary . . . 47
3.3 An Analysis . . . 48
3.3.1 Decision Ontology and Problem Structure . . . 48
3.3.2 Solution Methods . . . 62
3.3.3 Model Quality Metrics . . . 63
3.4 Summary . . . 66

4 An Integrated Language Design 69

4.1 Desiderata of An Integrated Language . . . 69
4.1.1 Expressive Decision Ontology . . . 69
4.1.2 Formal Theoretic Basis . . . 70
4.1.3 Multiple Levels of Abstraction . . . 70
4.1.4 Multiple Perspectives of Visualization . . . 70
4.1.5 Extensibility . . . 70
4.1.6 Adaptability . . . 71
4.1.7 Practicality . . . 71
4.2 Overview of Language Design . . . 71
4.3 DynaMoL: The Language . . . 72
4.3.1 Basic Decision Ontology . . . 73
4.3.2 Dynamic Decision Grammar . . . 76
4.3.3 Graphical Presentation Convention . . . 77
4.3.4 Mathematical Representation . . . 78
4.3.5 Translation Convention . . . 79
4.4 DYNAMO: A Prototype Implementation . . . 80

5 A Case Study 83

5.1 Management of Paroxysmal Atrial Fibrillation . . . 83
5.2 Case Description and Clinical Questions . . . 84
5.3 Assumptions . . . 84
5.4 Assessment Objectives . . . 85

6 Dynamic Decision Model Formulation 87

6.1 Dynamic Decision Modeling in DynaMoL . . . 87
6.2 Model Construction . . . 88
6.2.1 Basic Problem Characterization . . . 89
6.2.2 Action Space Definition . . . 91
6.2.3 State Space Definition . . . 92
6.2.4 State Transition Specification . . . 93
6.2.5 Event Variable Space Definition . . . 94
6.2.6 Probabilistic Influence Specification . . . 95
6.2.7 Value Function Definition . . . 97
6.2.8 Constraint Management . . . 98
6.2.9 Parametric Assessment . . . 102
6.3 Model Translation . . . 106
6.3.1 Translating Event Variables and Influences . . . 106
6.3.2 Translating Declaratory Constraints . . . 110
6.3.3 Translating Strategic Constraints . . . 112

7 Dynamic Decision Model Solution and Analysis 113

7.1 Solution Methods . . . 113
7.1.1 Value Iteration . . . 113
7.1.2 Fundamental Matrix Solution . . . 116
7.1.3 Other Methods . . . 116
7.2 Sensitivity Analyses . . . 117

8 Language Enhancement and Practical Development 119

8.1 Supporting Language Extension and Adaptation . . . 119
8.1.1 Static vs. Dynamic Spaces . . . 120
8.1.2 Automatic State Augmentation . . . 120

8.1.3 Limited Memory . . . 121
8.1.4 Numeric and Ordering Constraints . . . 121
8.1.5 Canonical Models of Combination . . . 122
8.1.6 Presentation Convention . . . 122
8.2 Supporting Practical Development . . . 123
8.2.1 Editing and Specification Support . . . 124
8.2.2 Statistical Analysis . . . 124
8.2.3 Data Visualization . . . 124

9 Conclusion 125

9.1 Summary . . . 125
9.1.1 The Analysis . . . 125
9.1.2 The Unifying View . . . 126
9.1.3 The General Methodology . . . 126
9.1.4 The Prototype System . . . 127
9.1.5 The Case Study . . . 128
9.2 Related Work . . . 129
9.3 Future Work . . . 130
9.3.1 Language and System Development . . . 130
9.3.2 Large Scale Evaluation . . . 130
9.3.3 Automatic Construction of Transition Functions . . . 130
9.3.4 Supporting Knowledge Based Model Construction . . . 131

A Dynamic Decision Grammar 133

B Semi-Markov Decision Process: Definitions and Techniques 137

B.1 Components of a Semi-Markov Decision Process . . . 137
B.1.1 Time Index Set . . . 137
B.1.2 State Space . . . 137
B.1.3 Action Space . . . 138
B.1.4 One-Step Transition Functions . . . 138
B.1.5 Alternate Definitions Of One-Step Transition Functions . . . 139
B.1.6 Value Functions . . . 144
B.2 Solutions of a Semi-Markov Decision Process . . . 145

C Dynamic Decision Model for the Case Study 147

C.1 The Action Space . . . 147
C.2 The State Space . . . 147
C.3 Numerical Variables . . . 148
C.4 The Event Variable Space and Conditional Distributions . . . 151
C.5 The Transition Values . . . 164
C.6 Solution and Analysis . . . 165

D Glossary 169

References 171

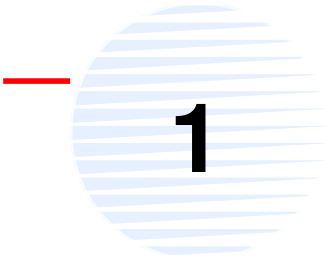
List of Figures

Figure 1.1 A dynamic decision problem. 17
Figure 2.1 State transition diagrams 26
Figure 2.2 Process trajectory diagrams 27
Figure 2.3 Information flow in a semi-Markov decision process. 27
Figure 2.4 State transition diagram of an embedded Markov chain for the example. 32
Figure 2.5 Relationships between time and stage views of decision horizons 33
Figure 2.6 An influence diagram and its embedded information. 35
Figure 2.7 A dynamic influence diagram for the example problem. 36
Figure 2.8 Use of Markov cycle trees to model dynamic effects of decision alternatives. 38
Figure 2.9 A Markov cycle tree. 39
Figure 2.10 A stochastic tree. 41
Figure 2.11 Basic manipulations in a stochastic tree. 42
Figure 2.12 A partial search space for an AI planner. 44
Figure 3.1 Interpretation by mapping. 48
Figure 3.2 Information flow in a partially observable semi-Markov decision process. 50
Figure 3.3 State transition diagram for example problem. 52
Figure 3.4 Dynamic influence diagrams representations of semi-Markov decision processes . 54
Figure 3.5 Action- and state-separable dynamic influence diagram. 56
Figure 3.6 A dynamic influence diagram with decomposable state descriptions. 56
Figure 3.7 A dynamic influence diagram with chance events constituting the state transitions and decomposed value functions. 57
Figure 3.8 Comparing a) a Markov cycle tree and b) a state transition diagram. 58
Figure 3.9 Using tunnel states to model duration dependent transitions. 59
Figure 3.10 A Markov cycle tree involving cross-over strategy with different consequences. . . 60
Figure 4.1 Transition view for an action. 78
Figure 4.2 Influence view for an action. 78
Figure 4.3 Inter-level translation by mapping. 79
Figure 4.4 Inter-perspective translation by mapping. 80
Figure 4.5 The system architecture of DYNAMO: a prototype implementation of DynaMoL. 80
Figure 4.6 The DYNAMO interface. 81
Figure 6.1 Basic problem characteristics of the case study (partial). 91
Figure 6.2 An abbreviated transition view for action Quinidine in case study. 94
Figure 6.3 Influence view for action Quinidine in case study. 96

Figure 6.4 A partial influence view in case study. 99
Figure 6.5 Constant and varying risks of bleeding for warfarin in case study. 105
Figure 6.6 Final influence view after reduction by the influence translator. 107
Figure 6.7 Indexing convention for updating conditional distribution table.. 108
Figure 6.8 An example to show conditional distribution table updating. 109
Figure 8.1 Asymmetric relations represented in an influence view. 123
Figure 8.2 Asymmetric relations represented in a tree view. 123

List of Tables

Table 6.1 State attribute variables and their outcomes for the case study 93
Table C.1 The states and corresponding value functions. 148
Table C.2 Solution to long-term discriminatory problem in case study. 166
Table C.3 Partial solution to short-term optimization problem. 167



1 *Introduction*

Decision making in our daily lives is often complicated by the complex information involved. Much of this complexity arises from the context-sensitive variations, the multiple levels of details, the uncertainty, and the dynamic nature of the underlying phenomena. Therefore, to automate decision making, we need a general and effective way to represent and manage such myriad of information.

This work is about analyzing and synthesizing techniques for supporting dynamic decision making under uncertainty. On analysis, we present a uniform way to reason about a class of dynamic decision problems; we also identify a common vocabulary for comparing and contrasting the major approaches to addressing such problems. On synthesis, we introduce a general methodology that integrates some salient features of existing techniques; the central ideas of this methodology are *multiple perspective reasoning* and *incremental language extension*. Finally, we examine how the proposed methodology can be put into practical use.

1.1 Background and Motivations

Dynamic decision making under uncertainty concerns decision problems in which time and uncertainty are explicitly considered. For example, a common medical decision is to choose an optimal course of treatment for a patient whose physical conditions may vary over time. Similarly, a common financial investment decision is to determine an optimal portfolio with respect to fluctuating market factors over time. These problems are particularly complicated if both the nature and the time of future events are uncertain.

Research in control theory, operations research, decision analysis, artificial intelligence (AI), and other disciplines has led to various techniques for formulating, solving, and analyzing dynamic decision problems. These techniques adopt

different assumptions, support different ontologies, and have different strengths and weaknesses. Consequently, insights gained and advancements achieved in one approach often do not benefit the others. Without a unifying perspective, confusing arguments about applying different techniques and wasted efforts in re-inventing existing technologies often result.

Recently, efforts to integrate techniques from different disciplines have emerged [Dean and Wellman, 1991] [DT-Planning, 1994]. Most of these works focus on adapting a particular framework to accommodate other techniques. Nevertheless, this phenomenon suggests that some properties or characteristics in each approach supplement or complement the other approaches. To facilitate integration, a uniform basis for examining and analyzing the different approaches is essential. Drawing analogy from knowledge representation research in AI, first-order predicate logic is such a common basis for analyzing the expressiveness and efficiency of deterministic representation languages.

Understanding the underlying conceptual models, instead of only the features of individual techniques, will contribute toward the effective design of a general methodology for dynamic decision making under uncertainty. On the other hand, a good methodology should be both theoretically sound and practically useful. Despite the variety of existing techniques, many of them have limited practicality. Some of the techniques are difficult to understand and apply, others impose too many restrictive assumptions. Therefore, to improve on current techniques, theoretical soundness and practical convenience should be simultaneously addressed.

1.2 The Dynamic Decision Problem

The general dynamic decision problem is to select a course of *action* that satisfies some *objectives* in an *environment*. Definitions of the actions, objectives, and environment may vary, *e.g.,* the actions can be described as continuous or discrete, the objectives as reaching some "goal states" or maximizing some effects of the actions, the environment as differential equations or discrete states. The variations distinguish the techniques applicable for the problems.

Figure 1.1 depicts the factors involved in a dynamic decision problem. The main distinguishing feature of a dynamic decision problem from a static one is the explicit reference of time. The environment description, the action or decision points, and the objective measures are specified with respect to a time horizon.

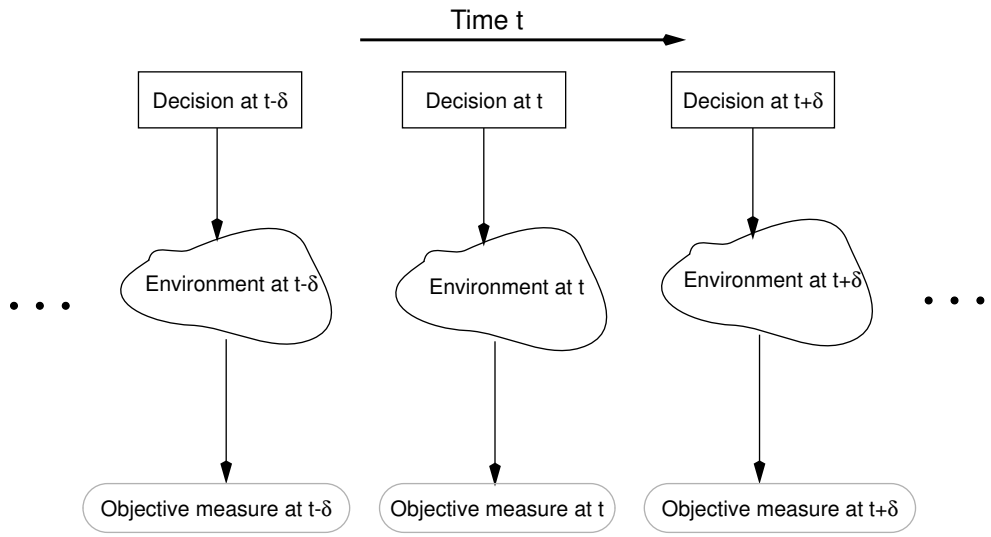


Figure 1.1 A dynamic decision problem.

This work addresses dynamic decision problems with the following properties:

First, the environment comprises a finite set of discrete, or reasonably assumed to be discrete, phenomena. A patient is either "well" or "sick" on the third day after being treated; a can that a robot is about to grasp is "full," "half-full," or "empty."

Second, there is a finite set of discrete actions. These actions are context-dependent; they have varying preconditions, usually with respect to the environment or time or both. A patient can only go through three open heart surgeries, and each surgery can only be carried out if the patient's physical conditions permit.

Third, each action has a finite set of discrete, or reasonably assumed to be discrete, effects. After a treatment, a patient who was previously "sick" is "well" again; moving forward from its current position, a robot "gets closer to the target position." The nature of the effects are often uncertain. A treatment either cures a disease or leads to some undesirable side-effects. Moreover, the time at which the effects may occur are also uncertain. A patient who is cured of peptic ulcer may have a relapse sooner than another patient.

Fourth, the effects of an action have measurable desirability. Such measure can be multiple dimensional, *e.g.,* the desirability of staying in a hospital and being well, versus staying at home and being sick, but it must be *time separable*, *i.e.,* the total desirability can be calculated by summing the desirability functions over time.

Given an initial environment, the problem is solved by choosing a course of action that optimizes the expected desirability of their potential effects The decisions are made in stages; the stages may vary in duration.

1.3 The Application Domain

While the issues we address are general, the application domain we examine is diagnostic test and therapy planning in medical decision making. Medicine is a very rich domain for dynamic decision making under uncertainty. The multitude of problems, the patient-specificity, and the uncertainty involved all contribute to the intricacy of the decisions. The large, complex, and ever-changing body of medical knowledge further complicates the process. Besides life and death decisions, maximizing the cost-effectiveness of the actions is also important. Therefore, multiple objective decision making is often involved; trade-offs are usually considered.

In this domain, the dynamic decision problems involve risks of some adverse events that continue or vary over time; the events may recur and the timing of such recurrences are important for making the decisions. The relevant actions are diagnostic tests and treatments, or combinations of them. The environment comprises the physical conditions of a patient or a class of patients. These conditions include the physiological states of the patient, or any observable or unobservable events that would lead to the states. For instance, a patient can be in a state with a stroke, and a cerebral hemorrhage may have caused the stroke. Some of these events are the effects of the actions.

1.4 Research Objectives and Approaches

Within the scope described above, this work answers the following questions:

- What are the different tasks in dynamic decision making under uncertainty?
- What is a good basis to compare and contrast the different techniques for solving the tasks?
- Can the strengths of existing techniques be integrated?
- Would such an integration improve on existing techniques?
- How can this integrated approach be practically useful?

The existing techniques relevant to this work are semi-Markov decision processes, dynamic decision modeling, and AI planning.

1.4.1 A Uniform Task Definition

We first define the different tasks in dynamic decision making under uncertainty as problem formulation, problem solution, and problem analysis. The task definition is based on analyzing the nature of the dynamic decision problems addressed, and the current approaches to solving such problems. We present a uniform way to describe the different aspects of a dynamic decision problem; we also illuminate the representational and inferential support required for these aspects.

Based on this task definition, we argue that besides involving different reasoning procedures, the different tasks also require different representational support. A vocabulary effective for supporting one task may not be effective for another. Most current techniques do not distinguish along this line.

1.4.2 A Common Vocabulary

We identify semi-Markov decision processes as a common vocabulary for analyzing current techniques. This common vocabulary is necessary because, in the different frameworks, different terms or constructs may denote the same concept, while the same term may denote different concepts.

The common vocabulary provides a uniform basis to analyze the semantic and syntactic correspondences, the strengths, and the weaknesses of the different techniques. Based on this analysis, we determine that the trade-off between model transparency and solution efficiency is the most significant limitation of current techniques.

1.4.3 A General Methodology

We introduce a general methodology for dynamic decision making under uncertainty. This new methodology motivates the design of a language design called DynaMoL, for Dynamic decision Modeling Language. It builds on the common basis of current techniques, and integrates some of their salient features.

To balance the trade-off between model transparency and solution efficiency, the DynaMoL design differentiates representational and inferential support for the modeling task from the solution or computation task. The central ideas involved are multiple perspective reasoning and incremental language extension. Multiple perspective reasoning allows us to visualize and examine the same information in different ways; it facilitates effective formulation and analysis of dynamic decision problems. Incremental language extension provides a framework that can be customized through the use of *translators*; it allows the scope of the dynamic decision problems addressed to be gradually expanded. The language design also admits various existing solution methods and supports systematic development of such methods.

1.4.4 A Prototype Implementation

We develop a prototype implementation of DynaMoL to examine the effectiveness of the proposed methodology. The prototype system, called DYNAMO, supports flexible, explicit, and concise specification and visualization of decision parameters, and incorporates several solution methods. A user can focus on the different tasks of dynamic decision making separately; the system will organize the relevant information for easy access and analysis.

1.4.5 A Case Study

We informally evaluate the effectiveness of the DynaMoL design and the DYNAMO system with a detailed case study. The case study involves an actual clinical decision situation.

Based on this case study, we demonstrate that DynaMoL is expressive enough to handle a class of real-life dynamic decision problems in medicine. We also claim that the proposed methodology is more general than most existing techniques. The exercise also illuminates some desirable features and tools for a practical system.

1.5 Contributions

The major contributions of this work are as follows:

First, we have established a unifying view of three major approaches to dynamic decision making under uncertainty. A detailed analysis based on this view highlights the capabilities and limitations of each approach. These results will facilitate choosing the correct techniques for different dynamic decision problems; they will also contribute toward effective integration and adaptation of the different techniques.

Second, we have proposed a novel language design that integrates the desirable features of current techniques. By introducing a new paradigm of multiple perspective reasoning, this design breaks the mold of single perspective reasoning supported in all existing graphical dynamic decision modeling languages. We have also established methods to systematically extend the language ontology. This is in contrast to the fixed vocabularies in most existing techniques.

Third, we have developed a prototype system that can handle a general class of dynamic decision problems. We are interested in putting the system into practical use. Towards this end, we have documented the experiences of performing a detailed case study in a complex domain; these lessons have illuminated the tools support required.

Finally, this research has provided insights into the nature of, and the difficulties and assumptions in the class of dynamic decision problems considered. These results can serve as guidelines for future research that addresses similar problems or improves current techniques.

1.6 Report Overview

This introductory chapter has briefly summarized the research motivations and objectives of this work. The remainder of the dissertation is organized as follows:

Chapter 2 introduces the current approaches to dynamic decision making under uncertainty, and briefly relates their developmental history.

Chapter 3 presents a uniform task definition of the dynamic decision problems addressed, defines a common vocabulary, and compares and contrasts the existing techniques.

Based on the analysis, Chapter 4 discusses the desiderata and the design approach for a general methodology that integrates the current techniques. The components of DynaMoL and a prototype implementation are also explained.

To facilitate description of decision making in DynaMoL, Chapter 5 introduces the domain background and decision problems for a case study. Details of the case study are included in Chapters 6 and 7.

Chapter 6 describes decision model formulation in DynaMoL; it examines in detail the syntax and the semantics of the language. The syntax prescribes how the decision parameters in a dynamic decision problem can be specified and manipulated; the semantics includes the mathematical representation of a dynamic decision problem, and a set of guidelines for interpreting the syntactic components of the language.

Chapter 7 examines the solution methods and the analyses supported by the language.

Chapter 8 discusses the possible extensions and improvements to the language features and practical capabilities. It also sketches some approaches to such extensions.

Finally, Chapter 9 summarizes the achievements and limitations of this work, compares the lessons learned with related work, and proposes some ideas for future research.

2 A Survey of Current Approaches

This chapter briefly surveys three major approaches to dynamic decision making under uncertainty: semi-Markov decision processes, dynamic decision modeling, and AI planning. This survey mainly introduces the different techniques; it serves as a basis to a more detailed analysis on the capabilities and limitations of the techniques in Chapter 3.

2.1 Historical Background

Research in all the dynamic decision making methodologies addressed began in the 1950s. Influences across disciplines were significant in the early stages, but generally became obscured, both to the researchers and the practitioners, as each approach matured and flourished. Such inter-disciplinary correspondence is again being noticed and exploited only recently.

2.1.1 Markov and Semi-Markov Decision Processes

Markov decision processes (MDPs) are mathematical models of sequential optimization problems with stochastic formulation and state structure. Research in MDPs began with the ideas of [Shapley, 1953] and [Bellman, 1957], and the formalization by [Howard, 1960] and others. [Jewell, 1963] extended these results to semi-Markov decision processes (SMDPs), which are more general than MDPs. Much progress has occurred since, both in extending the basic mathematical definitions and in improving the optimization algorithms [Puterman, 1990]. The rigorous, formal nature of the methodology, however, renders it quite formidable to formulate and analyze complex dynamic decision problems in practice. In medicine, for instance, while Markov and semi-Markov processes are often used for survival and

prognosis analyses, MDPs and SMDPs are seldom applied directly in clinical decision making [Janssen, 1986].

2.1.2 Dynamic Decision Modeling

Drawing on a set of ideas and results closely related to MDPs, decision analysis emerged in the 1960s from operations research and game theory [Raiffa, 1968]; it is a normative problem solving framework based on probability theory and utility theory. By systematically formulating, solving, and analyzing a graphical *decision model*, this approach helps in both gaining better insights into, as well as deriving optimal decisions for the problem [Howard, 1988]. In recent years, some new decision modeling formalisms have been devised to deal with dynamic decision problems. These dynamic decision models include *dynamic influence diagrams* [Tatman and Shachter, 1990], *Markov cycle trees* [Beck and Pauker, 1983] [Hollenberg, 1984], and *stochastic trees* [Hazen, 1992]; they are based on structural and semantical extensions of conventional decision models such as decision trees [Raiffa, 1968] and influence diagrams [Howard and Matheson, 1984], with the mathematical definitions of stochastic processes.

Dynamic decision modeling is used widely in real world applications. For instance, it is a common tool in clinical decision making [Kassirer et al., 1987] [Pauker and Kassirer, 1987] [Provan, 1992] [Provan and Clarke, 1993]. Nevertheless, the methodology is difficult to apply. In particular, model formulation is knowledge-intensive and labor-intensive, and the graphical structures restrict the admissible solution methods [Leong, 1993].

2.1.3 Planning in Artificial Intelligence

MDPs and dynamic decision modeling provide vocabularies for describing dynamic decision problems and computational models for choosing among decision alternatives. In contrast, AI planning also addresses decision knowledge organization and alternatives generation in dynamic decision problems. Motivated by the studies of human problem solving [Newell and Simon, 1972], operations research, and logical theorem proving, AI planning emerged with the works of [Newell et al., 1960] and [Fikes and Nilsson, 1971]. Early research in this approach focuses on representing and reasoning with complete and perfect information. Recent progress introduces imperfect information, extends the planning ontology, and improves the plan-generation and plan-execution algorithms [Tate et al., 1990].

Most of the planning works, however, are theoretical. Their impracticality is partly due to the complexity of the problems they address, and partly due to a trade-off between language expressiveness and solution efficiency. For instance, in a planning language with hierarchical representation of the actions, their effects,

and the relevant constraints, extra inferences are needed to derive the solutions. On the other hand, such representation support greatly facilitates problem formulation.

2.1.4 Recent Development

In the past few years, efforts to integrate techniques from the different approaches have begun to emerge. This leads to a brand new research discipline of *decision-theoretic planning*. Comparative investigations of specific aspects of the different methodologies are gaining attention [Dean and Wellman, 1991] [Haddawy and Hanks, 1990] [Haddawy and Hanks, 1992] [Wellman and Doyle, 1991] [Wellman and Doyle, 1992]. Most of the ongoing works, however, attempt to adapt a particular framework by incorporating some features of others [Dean et al., 1992] [Dean et al., 1993a] [Dean et al., 1993b] [Provan, 1992] [Provan and Clarke, 1993] [Wellman et al., 1992].

2.2 An Example

To illustrate the different methodologies, we examine a dynamic decision problem in the management of chronic ischemic heart disease (CIHD)[1]. CIHD is a disease that limits blood supply to the heart muscles, and hence impairs the normal performance of the heart as the blood circulation regulator; the most common type of CIHD is coronary artery disease (CAD). The problem, adapted and simplified from [Wong et al., 1990], is to determine the relative efficacies of different treatments for chronic stable angina, *i.e.,* chest pain, the major manifestation of CAD. The alternatives considered are medical therapy, percutaneous transluminal angioplasty (PTCA), and coronary artery bypass graft (CABG). The treatment efficacies are evaluated with respect to quality-adjusted life expectancy (QALE).

CAD is usually atherosclerotic in nature. Atherosclerosis means progressive obstruction of the arteries with the formation of plaque, which contains fatty deposits. The manifestations of CAD, therefore, are progressive. If the angina worsens after a treatment, subsequent actions will be considered. Even after successful treatment, restenosis or renewed occlusion of the arteries may occur. Hence, a sequence of decisions must be made. For ease of discussion, all three treatments are assumed to be repeatedly applicable.

As time progresses, the treatment efficacies in lowering mortality and morbidity decline, and the treatment complications worsen. These might be due to the progression of the disease, or the deteriorating status of the patient with time. A major complication for PTCA is perioperative myocardial infarction (MI), which

1. A glossary of the medical concepts can be found in Appendix D.

would render an emergency CABG necessary. Non-procedural related MI may also occur after a treatment.

In this example, therefore, the risks and benefits of the actions vary over time, some important events may recur over time, and the timing of such events are uncertain.

2.3 Semi-Markov Decision Processes

As mentioned in Section 2.1.1, an SMDP is a mathematical model of a sequential decision process. The decision problem is formulated in terms of a set of actions, a set of states with associated values, and a set of stochastic transition characteristics. The stochastic nature of the transitions are reflected in both the *destination* of the transition and the *time lapsed* before the transition. There are many subclasses of SMDPs; specializations can be along discrete or continuous time units, discrete or continuous actions, discrete or continuous states, and deterministic or stochastic inter-transition times. An MDP is an SMDP in which the inter-transition times are constant at one time unit.

Although the formal vocabulary of SMDPs does not include any graphical components, a variety of graphical devices are commonly used to depict the concepts involved. These include s*tate transition diagrams*, *process trajectory diagrams*, and *information flow diagrams*.

Figure 2.1 shows the state transition diagrams for a simple example where a patient or a class of patients can be well, sick, or dead at any point in time. States are depicted as circles and possible transitions as arcs connecting the circles; each action determines a different set of transitions.

Figure 2.1 State transition diagrams showing: a) all possible transitions; b) next transitions in time.

A process trajectory diagram illustrates a sample path or instance of a process, *e.g.,* how a single patient would behave if he is governed by the process. Figure 2.2 shows two process trajectory diagrams for the same simple example above. The first diagram assumes variable inter-transition times; the second diagram assumes constant inter-transition times, *i.e.,* a Markov process. The semi-Markov processes

depicted in the diagrams allow *virtual* transitions, which are transitions back to the same state. In contrast, *real* transitions are transitions to other states. Depending on the modeling assumptions in a semi-Markov process, sometimes only real transitions are appropriate, sometimes virtual transitions are necessary. When the state of the process represents the last brand purchased by the customer in a marketing model, for instance, a virtual transition represents a repeat purchase of a brand [Howard, 1971]. Conversion methods and correspondence between virtual and real transitions are described in more details in Appendix B.

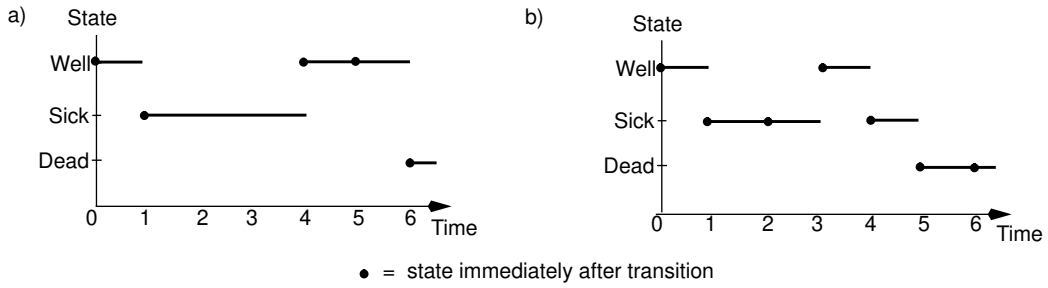


Figure 2.2 Process trajectory diagrams for: a) a semi-Markov process; b) a Markov process.

Figure 2.3 shows the information flow diagram for a simple SMDP where the states are perfectly observable. This diagram illustrates the dynamics of the sequential decision problem. Values are accumulated for each state visited.

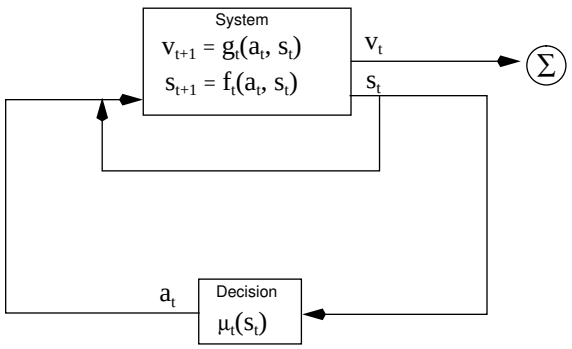


Figure 2.3 Information flow in a semi-Markov decision process. At each time t the decision maker observes the current state s_t and applies action $a_t = \mu_t(s_t)$ that depends on the state. A value v_t is accumulated in the mean time.

Formal Definitions

Formally, an SMDP addressed in this work is characterized by the following components [Howard, 1971] [Heyman and Sobel, 1984]:

- A *time index set* $\boldsymbol{T}$;
- A *decision* or *control process* denoted by a set of random variables $\{D(t);\ t \in \boldsymbol{T}\}$, where $D(t) \in \boldsymbol{A} = \{1, 2, \ldots, \alpha\}$ is the decision made at time t; and
- A *semi-Markov reward process* denoted by a set of random variables $\{S(t);\ t \in \boldsymbol{T}\}$, where $S(t) \in \boldsymbol{S} = \{0, 1, 2, \ldots\}$ is the state of the process at time t, with:

1. an embedded Markov chain denoted by a set of random variables $\{S_k;\ k \geq 0\}$; such that $S_k = S(T_k)$, where $T_1 < T_2 < T_3 < \ldots$ are the random variables denoting the successive epochs or instants of time at which the process makes transitions;
2. α sets of transition probabilities $\{P_{ij}^{(a)}(t);\ i \geq 0, j \geq 0, 1 \leq a \leq \alpha, t \in \boldsymbol{T}\}$ among the states of the embedded chain. The transition probabilities indicate the fractions of (eventual) transitions from a state *i* to another state *j*, given that the process enters state *i* at time *t*, and an action *a*.

   For any given action $a \in \boldsymbol{A}$, and states $i, j \in \boldsymbol{S}$:

$$\begin{aligned} P_{ij}^{(a)}(t) &= P\{S_{k+1} = j \mid S_k = i, D_k = a, T_k = t\} \\ &= P\{S(T_{k+1}) = j \mid S(T_k) = i, D(T_k) = a, T_k = t\} \end{aligned}$$

   which also satisfies the *Markovian property, i.e.,* it does not depend on how the process gets to the current state *i*:

$$\begin{aligned} P_{ij}^{(a)}(t) &= P\{S_{k+1} = j \mid S_k = i, D_k = a, T_k = t\} \\ &= P\{S_{k+1} = j \mid S_k = i, S_{k-1} = h, \ldots, D_k = a, T_k = t\} \end{aligned}$$

3. α sets of holding times $\{\tau_{ij}^{(a)}(t);\ i \geq 0, j \geq 0, 1 \leq a \leq \alpha, t \in \boldsymbol{T}\}$ among the states of the embedded chain, which are random numbers with corresponding probability mass functions (PMFs):

$$\{h_{ij}^{(a)}(m, t);\ i \geq 0, j \geq 0, 1 \leq a \leq \alpha, m \geq 0, t \in \boldsymbol{T}\}$$

   and cumulative distribution functions (CDFs):

$$\{H_{ij}^{(a)}(m, t);\ i \geq 0, j \geq 0, 1 \leq a \leq \alpha, m \geq 0, t \in \boldsymbol{T}\}.$$

   The holding times describe the amount of time the process spent in a particular state *i*, given that the process enters state *i* at time *t*, its next transition is to state *j*, and an action *a*.

   For any given action $a \in \boldsymbol{A}$, and states $i, j \in \boldsymbol{S}$:

$$h_{ij}^{(a)}(m,t) = P\{\tau_{ij}^{(a)}(t) = m\}$$
$$= P\{T_{k+1} - T_k = m \mid S_k = i, S_{k+1} = j, D_k = a, T_k = t\};$$
$$m = 0, 1, 2, \ldots$$

and

$$H_{ij}^{(a)}(m,t) = P\{\tau_{ij}^{(a)}(t) \le m\}$$
$$= P\{T_{k+1} - T_k \le m \mid S_k = i, S_{k+1} = j, D_k = a, T_k = t\};$$
$$m = 0, 1, 2, \ldots$$

and

4. α sets of rewards or value functions $\{v_i^{(a)}(m);\ i \ge 0, 1 \le a \le \alpha, m \ge 0\}$ associated with the states of the embedded chain, such that for any given action $a \in \boldsymbol{A}$, $v_i^{(a)}(m)$ is the value achievable in state $i \in \boldsymbol{S}$ over time duration m.

   $v_i^{(a)}(m)$ may be defined in terms of the sets of value functions associated with the possible transitions $\{v_{ij}^{(a)}(m);\ i \ge 0, j \ge 0, 1 \le a \le \alpha, m \ge 0\}$, given any action $a \in \boldsymbol{A}$ and $i, j \in \boldsymbol{S}$.

Together, the transition probabilities and the holding time distributions constitute the *one-step transition functions* with PMFs:

$$\{q_{ij}^{(a)}(m,t);\ i \ge 0, j \ge 0, 1 \le a \le \alpha, m \ge 0, t \in \boldsymbol{T}\}$$

and CDFs:

$$\{Q_{ij}^{(a)}(m,t);\ i \ge 0, j \ge 0, 1 \le a \le \alpha, m \ge 0, t \in \boldsymbol{T}\}$$

The one-step transition functions characterize the state transitions by answering the following question: Given the process enters state i at time t and an action a, how likely is it that the next transition is a transition to state j, and that the transition will occur at n time units after entering state i (for the PMF), or by n time units after entering state i (for the CDF)?

For any given action $a \in \boldsymbol{A}$, states $i, j \in \boldsymbol{S}$, and time $t \in \boldsymbol{T}$:

$$q_{ij}^{(a)}(m,t) = P_{ij}^{(a)}(t) \cdot h_{ij}^{(a)}(m,t)$$
$$= P\{S_{k+1} = j, T_{k+1} - T_k = m \mid S_k = i, D_k = a, T_k = t\};$$
$$m = 0, 1, 2, \ldots$$

and

$$Q_{ij}^{(a)}(m,t) = P_{ij}^{(a)}(t) \cdot H_{ij}^{(a)}(m,t)$$
$$= P\{S_{k+1} = j, T_{k+1} - T_k \le m \mid S_k = i, D_k = a, T_k = t\};$$
$$m = 0, 1, 2, \ldots$$

In the special case of an MDP, the holding time PMFs and CDFs are:

$$h_{ij}^{(a)}(m,t) = h_{ij}^{(a)}(m) = \delta(m-1)$$

where $\delta(m-1)$ is the unit-impulse function; and

$$H_{ij}^{(a)}(m,t) = H_{ij}^{(a)}(m) = \mathbf{1}(m-1)$$

where $\mathbf{1}(m-1)$ is the unit-step function.

Therefore, the holding times for an MDP are all exactly one time unit in length, and the corresponding transition functions are:

$$q_{ij}^{(a)}(m,t) = q_{ij}^{(a)}(m) = P_{ij}^{(a)} \cdot \delta(m-1);$$

and

$$Q_{ij}^{(a)}(m,t) = Q_{ij}^{(a)}(m) = P_{ij}^{(a)} \cdot \mathbf{1}(m-1).$$

The one-step transition functions can also be defined in terms of the *conditional transition probabilities* $\{p_{ij}^{(a)}(m,t);\ i \ge 0, j \ge 0, 1 \le a \le \alpha, m \ge 0, t \in \boldsymbol{T}\}$ and *waiting times* $\{\tau_i^{(a)}(t);\ i \ge 0, 1 \le a \le \alpha, t \in \boldsymbol{T}\}$ with PMFs:

$$\{w_i^{(a)}(m,t);\ i \ge 0, 1 \le a \le \alpha, m \ge 0, t \in \boldsymbol{T}\}$$

and CDFs:

$$\{W_i^{(a)}(m,t);\ i \ge 0, 1 \le a \le \alpha, m \ge 0, t \in \boldsymbol{T}\}.$$

The conditional transition probabilities indicate the fractions of transitions from state *i* to another state *j*, given that the process enters state *i* at time *t*, makes a transition at duration *m* after entering state *i*, and an action *a*.

$$p_{ij}^{(a)}(m,t) = P\{S_{k+1} = j \mid S_k = i, D_k = a, T_k = t, T_{k+1} - T_k = m\}$$

which also satisfies the *Markovian property*:

$$p_{ij}^{(a)}(m,t)$$
$$= P\{S_{k+1} = j \mid S_k = i, D_k = a, T_k = t, T_{k+1} - T_k = m\}$$
$$= P\{S_{k+1} = j \mid S_k = i, S_{k-1} = h, \ldots, D_k = a, T_k = t, T_{k+1} - T_k = m\}$$

The waiting time or unconditional holding times $\tau_i^{(a)}(t)$ are random variables describing the amount of time a process spent in a particular state *i*, given that the precess enters state *i* at time *t* and an action *a*.

For any given action $a \in \boldsymbol{A}$, and states $i \in \boldsymbol{S}$:

$$\begin{aligned} w_i^{(a)}(m,t) &= P\{\tau_i^{(a)}(t) = m\} = \sum_j P_{ij}^{(a)}(t)\, h_{ij}^{(a)}(m,t) \\ &= P\{T_{k+1} - T_k = m \mid S_k = i, D_k = a, T_k = t\} \end{aligned}$$

and

$$\begin{aligned} W_i^{(a)}(m,t) &= P\{\tau_i^{(a)}(t) \le m\} = \sum_j P_{ij}^{(a)}(t)\, H_{ij}^{(a)}(m,t) \\ &= P\{T_{k+1} - T_k \le m \mid S_k = i, D_k = a, T_k = t\} \end{aligned}$$

Therefore, for any given action $a \in \boldsymbol{A}$, states $i, j \in \boldsymbol{S}$, and time $t \in \boldsymbol{T}$:

$$q_{ij}^{(a)}(m,t) = p_{ij}^{(a)}(m,t)\, w_i^{(a)}(m,t);$$
$$m = 0, 1, 2, \ldots$$

and

$$Q_{ij}^{(a)}(m,t) = p_{ij}^{(a)}(m,t)\, W_i^{(a)}(m,t);$$
$$m = 0, 1, 2, \ldots$$

We shall discuss in more details the alternate definitions of one-step transition functions in Section 6.2.9. The correspondences between the two definitions of one-step transition functions, as well as different computational methods associated with SMDPs, can be found in Appendix B.

The transition functions and its components described above are *nonhomogeneous* or time-dependent; the transition characteristics depend on when the process enters a particular state. For the special case of *homogeneous* transition functions, the transition characteristics are independent of when the process enter a particular state, i.e., independent of $T_k = t;\ t \in \boldsymbol{T}$.

In the example problem, assume that the states $\boldsymbol{S}$ = {Well, Restenosis, MI, MI+Restenosis, Dead} represent the possible physical conditions or health outcomes of a patient, given any particular treatment $a \in \boldsymbol{A}$ = {MedRx, PTCA, CABG}. For ease of exposition, assume that each state $s \in \boldsymbol{S}$ is a function of a set of binary *state attribute* or health outcome variables $\Theta = \{\boldsymbol{status}, \boldsymbol{MI}, \boldsymbol{restenosis}\}$, such that Well := {***status*** = *alive,* ***MI*** = *absent,* ***restenosis*** = *absent*}, MI := {***status*** = *alive,* ***MI*** = *present,* ***restenosis*** = *absent*}, and so forth.

The semi-Markov reward process, with time index set $\boldsymbol{T} \subseteq \{0, 1, 2, \dots\}$, is defined by:

1. the embedded Markov chain as depicted in Figure 2.4;
2. three sets of transition probabilities $P_{ij}^{(a)}(.)$ among the states in $\boldsymbol{S}$, conditional on the actions in $\boldsymbol{A}$;
3. three sets of holding time PMFs $h_{ij}^{(a)}(.)$ or CDFs $H_{ij}^{(a)}(.)$ among the states in $\boldsymbol{S}$, conditional on the actions in $\boldsymbol{A}$; and
4. three sets of value functions $v_i^{(a)}(.)$, corresponding to the amount of QALE expected in each state in $\boldsymbol{S}$, conditional on the actions in $\boldsymbol{A}$.

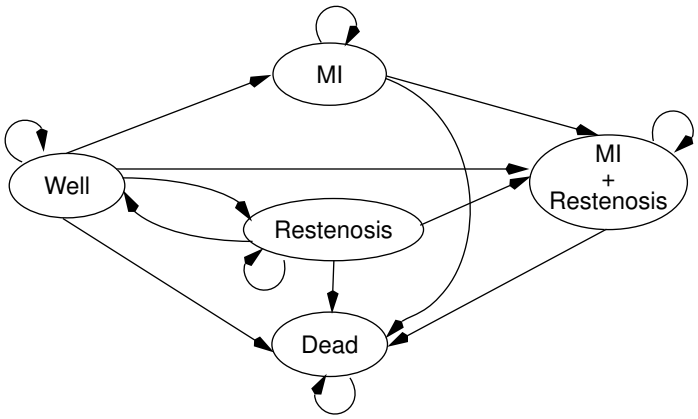


Figure 2.4 State transition diagram of an embedded Markov chain for the example. The nodes denote the states. The links represent possible transitions from one state to another, given any treatment.

Solutions

A solution to an SMDP is an *optimal policy*. An optimal policy $\pi^* = \{\mu_0^*, \mu_1^*, \mu_2^*, \mu_3^*, \dots\}$ specifies the optimal *decision rule* $\mu_t^* : \boldsymbol{S} \rightarrow \boldsymbol{A}$ for each $t \in \boldsymbol{T}$ that maximizes the expected value achievable in each state; a policy is *stationary* if it contains the same decision rule over the entire decision horizon. The optimal policies are guidelines for choosing the optimal actions over the decision horizon, for all possible evolutions of the states, that maximize the expected value or reward.

A variety of solution methods are available for SMDPs. The most common algorithms are based on the *value iteration* method of the dynamic programming or Bellman optimality equation. A dynamic decision problem can be expressed as the dynamic programming equation of an SMDP. Due to its complexity, we shall leave the details of this equation to Chapter 7. For now, it suffices to show the general form of the optimality equation for discrete-time MDPs, with an arbitrary discount factor $0 \leq \beta \leq 1$:

$$V_i^*(\beta, t) = max_a \{ \beta^t v_i^{(a)}(1) + \sum_j P_{ij}^{(a)}(t) \cdot V_j^*(\beta, t+1) \};$$
$$i, j \in \boldsymbol{S}, a \in \boldsymbol{A}, t \in \boldsymbol{T} \quad \textbf{(EQ 1)}$$

The solution to the optimality equation is an optimal policy that maximizes $V_{init}^*(0)$, the optimal expected value or reward for an initial state, *e.g.,* the well state for the example problem, at time 0.

Alternately, (EQ 1) can be reformulated as follows:

$$V_i^*(\beta, n) = max_a \{ v_i^{(a)}(1) + \beta \sum_j P_{ij}^{(a)}(N-n) \cdot V_j^*(\beta, n-1) \};$$
$$i, j \in \boldsymbol{S}, a \in \boldsymbol{A}, n = 0, 1, 2, \ldots, N \quad \textbf{(EQ 2)}$$

The solution to this alternate optimality equation is an optimal policy that maximizes $V_{init}^*(N)$, the optimal expected value or reward for an initial state over *duration N* or *N stages*.

The remaining decision stages, sometimes called decision stages "to go," indicates the remaining time in the decision horizon. The relationship between time and stage is depicted in Figure 2.5.

Time	0	1	2	...	t-1	t	Finite Horizon
Stage	n	n-1		···	1	0	
Time	0	1		...		∞	Infinite Horizon
Stage	∞			···	1	0	

Figure 2.5 Relationships between time and stage views of dynamic decision problem horizons

Note that in the Markov case, as illustrated in (EQ 1) and (EQ 2), the time variable *t* and the duration variable *n* have one-to-one direct correspondence. In other words, the time horizon is defined as $\boldsymbol{T} = \{0, 1, 2, 3, \ldots, N\}$, such that $n = N - t;\ t \in \boldsymbol{T}, n = 0, 1, 2, \ldots, N$. Therefore, (EQ 2) is equivalent to:

$$V_i^*(n, \beta, t) = max_a \{ v_i^{(a)}(1) + \beta \sum_j P_{ij}^{(a)}(t) \cdot V_j^*(n-1, \beta, t+1) \};$$
$$i, j \in \boldsymbol{S}, a \in \boldsymbol{A}, n = 0, 1, 2, \ldots, N, t \in \boldsymbol{T} \quad \textbf{(EQ 2')}$$

Other solution methods for SMDPs include policy iteration, linear programming, *etc*. Applicability of the different solution methods depends on certain problem characteristics such as constant discount factor, stationary policy, and homogeneous transition functions. Some of these methods are described in more details in Appendix B.

2.4 Dynamic Decision Modeling

A dynamic decision model is based on a graphical modeling language that depicts relevant parameters for decision analysis in a dynamic decision problem. All the graphical components have precise mathematical definitions. In general, such a model consists of the following six components, the first five of which constitute a conventional decision model such as decision tree or influence diagram:

- A set of *decision nodes* listing the alternative actions that the decision maker can take, for instance, the choices of medical therapy, PTCA, and CABG;
- A set of *chance nodes*, corresponding to a set of random variables, outlining the possible outcomes or happenings that the decision maker has no control over, for example, the physical status of the patient, the prognostic outcomes of PTCA, and the complications of CABG;
- A single or a set of *value functions*, sometimes denoted as *value nodes*, capturing the desirability, in terms of cost, life-expectancy, *etc.*, of each outcome or action;
- A set of *conditional dependencies* depicting how the outcomes of each chance node probabilistically depend on other outcomes or actions;
- A set of *informational* or *temporal dependencies* indicating the information available when the decision maker makes a decision; and
- An underlying *stochastic process* governing the evolution in time for the above five components.

The stochastic processes underlying existing dynamic decision modeling frameworks are specializations of *semi-Markov processes*. The definition of a semi-Markov process is similar to that in Section 2.3, except without the decision component.

A solution to a dynamic decision model is a course of optimal action that maximizes the value functions. The solution, derived by *evaluating* the model, usually involves interleaving chance nodes expectation and decision nodes maximization.

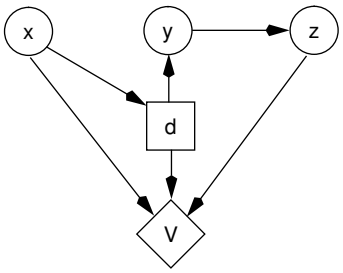
We briefly examine three types of dynamic decision models for the example problem. The embedded Markov chains for the semi-Markov processes underlying these models are identical to the one shown in Figure 2.4.

2.4.1 Dynamic Influence Diagrams

An influence diagram is a graphical stochastic model that can explicit display conditional and temporal dependencies in a decision problem. An influence diagram is a *hierarchical* representation; it has a qualitative layer and a quantitative layer. The qualitative level is a directed graph with no cycles; it summarizes the relations among the decision parameters. The nodes in the graph correspond to the parame-

ters in the model; the links represent the relations among the parameters. The quantitative level includes the embedded information in the nodes. Embedded in each chance node or value node is a list of possible outcomes or values, and a table of probabilities conditional on its probabilistic predecessors. Embedded in each decision node is a list of the alternate treatments and a list of its informational predecessors.

Figure 2.6 shows a simple influence diagram. The notation depicts rectangles as decision nodes, ovals as chance nodes, and diamonds as value nodes. The links leading into the chance and value nodes indicate conditional dependence; the links leading into the decision nodes indicate informational or temporal dependence. The absence of links between any pair of nodes indicates conditional or temporal independence.The diagram is interpreted as follows: Random variables *x* and *y* are independent; *z* is conditioned on *y*. The decision maker will know the outcome of *x* before decision *d* is made; the decision will affect the outcome of *y*. The objective is to optimize the expected value of *V*, which is conditioned on *d*, *x*, and *z*.



d:
alternatives: (A1, A2)
predecessors: (x)
x:
outcomes: (X, Not-X)
probs: [0.1 0.9]
y:
outcomes: (Y, Not-Y)
conditioning-outcomes: (A1, A2)
probs: [0.2 0.8; 0.7 0.3] (read: [Y|A1 Not-Y|A1; Y|A2 Not-Y|A2])
z:
outcomes: (Z1, Z2, Z3)
conditioning-outcomes: (Y, Not-Y)
probs: [0.1 0.3 0.6; 0.5 0.2 0.2]
V:
values: f(d, x, z)

Figure 2.6 An influence diagram and its embedded information.

Dynamic influence diagrams [Tatman and Shachter, 1990] extend influence diagrams by allowing *time-separable* value functions; each component value function measures the value achievable in a single time unit or decision stage. A dynamic influence diagram has the same kinds of graphical components as an influence diagram as shown in Figure 2.6, except that there will be several component or non-terminal value nodes, and a terminal value node. The relationship between the non-terminal value nodes v_n, where n is the time index or decision stage, and the terminal value node V has one of the following forms:

$$V = \sum_n v_n \text{ or } V = \prod_n v_n$$

Figure 2.7 shows a dynamic influence diagram for the example problem. The number at the end of each node indicates the decision stage "to go," in which the parameter is considered. Each decision stage corresponds to the time interval between two successive actions, which also indicates two successive transitions in the underlying discrete-time Markov chain; the time intervals may be in any unit and are usually constant within the model. The diagram indicates a decision horizon of three time units. At the beginning of the decision horizon, a decision is made based on whether CAD or MI or both are present, and also whether the patient is alive. At each subsequent decision stage, the decision is made based on whether restenosis or MI or both have occurred.

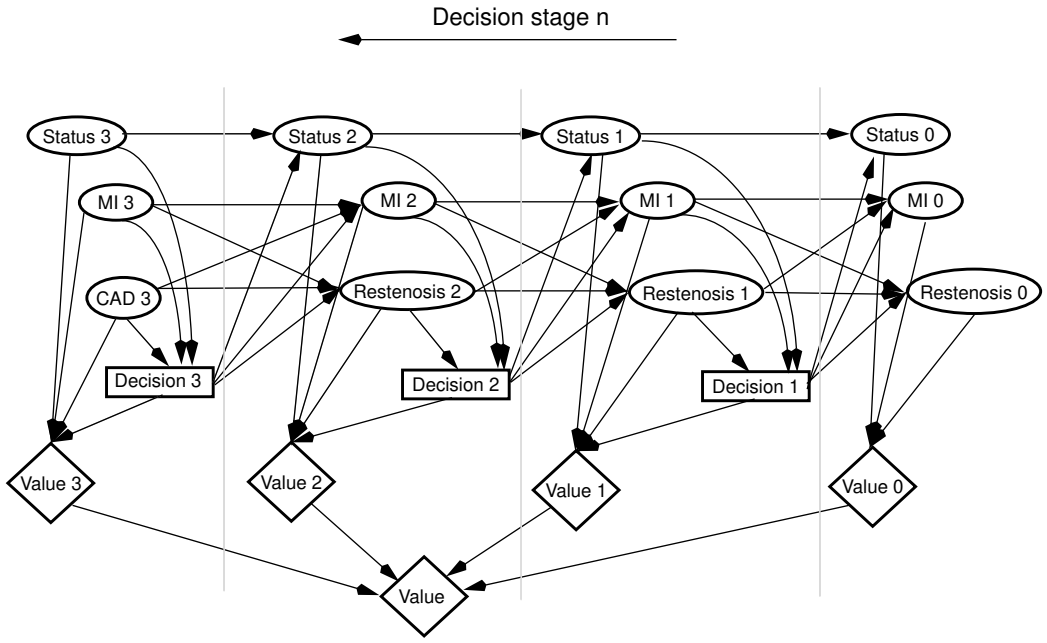


Figure 2.7 A dynamic influence diagram for the example problem.

Solutions

In a dynamic influence diagram, the decision problem is solved by graph reductions through a series of *value preserving transformations*. These reductions or transformations preserve the joint probability distribution, the optimal course of action, and the expected value of the influence diagram by removing the decision nodes, chance nodes, and non-terminal value nodes.

There are five basic transformations in solving a dynamic influence diagram [Shachter, 1986] [Tatman and Shachter, 1990]:

Barren Node Removal

A barren node is a chance node with no successors. The distribution of such a node does not affect the joint probability distribution of the diagram. Hence, a bar-

ren node can be removed from the diagram without any consequence. If the removal of a barren node causes other nodes to become barren, they can be removed as well.

Arc Reversal

Arc or link reversal between two chance nodes corresponds to applying Bayes theorem, when there is no other path between the two chance nodes. The two nodes involved inherit each others' predecessors.

Chance Node Removal

Chance node removal corresponds to conditional expectation. We first add the predecessors of the chance node to the list predecessors of its single successor. When the chance node has another chance node as its successor, we eliminate the chance node by summing its outcomes out of the joint probability distribution. When the chance node has a value node as its successor, we eliminate the chance node by calculating its expected value.

Decision Node Removal

When a decision node has a value node as its single successor, and all other conditional predecessors of that value node are informational predecessors of the decision node, the decision node can be removed by maximizing the expected value of its alternatives.

Non-terminal Value Node Removal

Finally, non-terminal value node removal corresponds to summing or multiplying the value into the terminal value node.

In addition to introducing the notion of time-separable value nodes, [Tatman and Shachter, 1990] also provide a well established algorithm for solving dynamic influence diagrams. In order to take advantage of a much larger collection of evaluation algorithms, recent efforts have tried to translate these models into Bayesian or probabilistic networks [Shachter and Peot, 1992]; all these algorithms, however, are NP-hard [Cooper, 1990] with respect to the size of the models**.**

2.4.2 Markov Cycle Trees

A Markov cycle tree models the dynamic effects of a decision alternative in the conventional decision tree. A Markov cycle tree does not represent a complete dynamic decision problem itself. Instead, it is part of a larger model which usually involves as many cycle trees as there are decision alternatives. Figure 2.8 shows the common use of Markov cycle trees in a dynamic decision model. Although the cycle trees in the same model may be different in principle, they usually have the same structure in practice.

Figure 2.8 Use of Markov cycle trees to model dynamic effects of decision alternatives.

Graphically, a Markov cycle tree is similar to a conventional decision tree. It is a single level representation; the qualitative information and the quantitative information are depicted simultaneously. The qualitative information, in terms of a set of nodes and links, summarizes the decision parameters and their relations; the quantitative information reflects the actual conditional probabilities and values. The nodes represent the decision parameters. No decision nodes are allowed in a cycle tree. The chance nodes represent random variables. The state nodes, which have associated values, represent the states in the underlying embedded discrete-time Markov chain. The links indicate the outcomes of the chance nodes, and the conditional and temporal dependencies among these outcomes and the states. Conditional and temporal independence, however, are not explicitly displayed. The conditional probabilities among the outcomes of the chance nodes, conditioned on the specific action modeled by the cycle tree, are associated with the links.

Figure 2.9 depicts a Markov cycle tree for the example problem; three cycle trees with the same structure, but different quantitative information, are attached to the end of the decision alternatives for the conventional decision tree with a single decision node. The cycle tree models the possible state transitions in the embedded Markov chain in a decision stage, conditional on the specified action. All possible combinations of the chance events that could happen at each decision stage, or between any two transitions of the underlying Markov chain, are represented in between the root and the leaves of the cycle tree. The inter-transition time intervals are constant. The cycle tree structure usually remains constant over time, only the quantitative information may change for each decision stage, *e.g.,* with time-varying transition probabilities. This prevents the size of the tree from “exploding” over time.

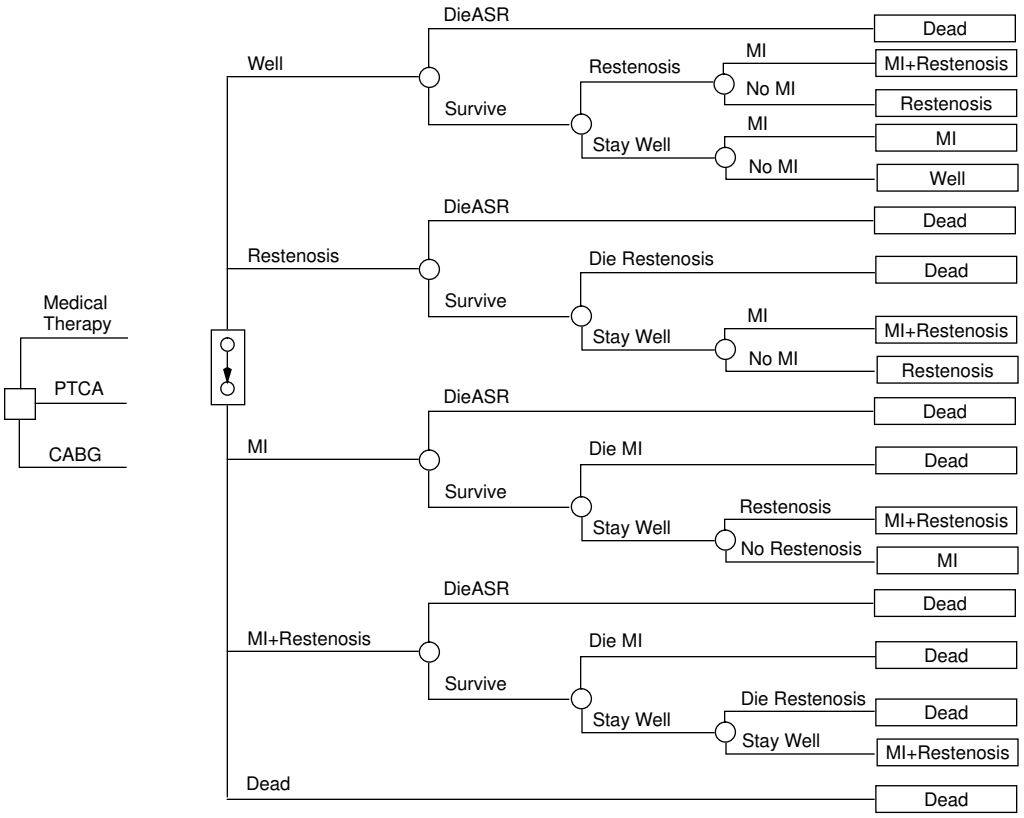


Figure 2.9 A Markov cycle tree.The symbol ⟨○→○⟩ denotes the root of the cycle tree, and each ⟨*state*⟩ a leaf, *i.e.,* the state to start in the next cycle. The label DieASR means die of all other causes, conditioned only on the age, sex, and race of the patient.

Solutions

The solution to a dynamic decision model with Markov cycle trees determines the single optimal action or decision alternative over time; this is done by calculating and comparing the values achievable over time from the respective cycle trees. Two common solution methods for Markov cycle trees are *cohort analysis* [Beck and Pauker, 1983] and *Monte Carlo simulation*; both are simulation methods, and individual simulations are conducted for each cycle tree in the model.

In cohort analysis, the simulation begins with a hypothetical cohort in some initial distribution, with respect to the root of the cycle tree, among the states. For instance, some patients who have undergone PTCA begin in the Well state, others in the MI state due to perioperative complication. In each decision stage or "cycle," the fraction of cohort in each state is partitioned among all the states according to the transition probabilities specified through the intermediate chance events. For instance, with reference to Figure 2.9, the transition probability from the Well state to the MI state, conditioned on PTCA, is:

$$p^{\text{PTCA}}(\text{Well} \rightarrow \text{MI}) = p^{\text{PTCA}}\text{Survive} \cdot p^{\text{PTCA}}\text{Staywell} \cdot p^{\text{PTCA}}\text{MI}$$

This results in a new distribution of the cohort among the states. As each individual of the cohort passes through a state, value is accumulated for the passage. At each decision stage, the expected values accumulated, called *cycle sum*, are:

$$V_n^{(a)} = \sum_s fr_{s,n}^{(a)} \cdot v_s^{(a)}(1)$$

where n is the decision stage index, $a \in \{\text{MedRx}, \text{PTCA}, \text{CABG}\}$, $s \in \{\text{Well}, \text{MI}, \text{Restenosis}, \text{MI} + \text{Restenosis}, \text{Dead}\}$, $v_s^{(a)}(1)$ is the value achievable in state s, conditioned on action a, for over a single decision stage, and $fr_{s,n}^{(a)}$ is the fraction of the cohort in state s at decision stage n, given action a. The next cycle starts with a new distribution at the root of the cycle tree. The expected values are accumulated until the process converges, or when the Dead state reaches probability 1. The final result, or the expected values achievable over time for the specified action, is:

$$V^{(a)} = \sum_{n=0}^{\infty} V_n^{(a)}$$

The evaluation may take an arbitrarily long time to terminate, or it may not terminate at all.

In Monte Carlo simulation, the simulation involves a large number of individual trials. Each individual begins in a starting state, for instance, the Well state. At the end of each decision stage, a random-number generator guided by the transition probabilities determines the next state of the individual. Values are accumulated as the individual passes through each state. The trial stops when the individual enters an *absorbing state* such as the Dead state. The process is repeated many times to derive a distribution of the values achievable; the mean of this distribution is the expected value achievable for the specified action.

Both cohort analysis and Monte Carlo simulation are *forward induction* processes; the problem is solved by starting at the initial stage or time, and moving forward one stage at a time, until all stages are included. This contrasts with the backward induction process, where the problem is solved by starting at the final stage, and moving back one stage at a time, until all stages are included. Forward induction is applicable in a Markov cycle tree because there are no decisions or choices involved, except at the very beginning. In other words, there is no future optimization involved. In general, forward induction cannot address optimization over time problems because there is no way to compare future values in the process. It can, however, be augmented with auxilliary constructs to address some history-dependent problems; this may be useful for alleviating the restrictive “memoriless” Markov assumption in such cases.

2.4.3 Stochastic Trees

A stochastic tree is the continuous time version of a Markov cycle tree; it models a continuous-time Markov chain. The inter-transition time intervals between any two states i and j, given an action a, are exponentially distributed: $H_{ij}^{(a)}(\tau) = 1 - e^{-r\tau}$, with specific rate r and time interval τ.

Figure 2.10 depicts a stochastic tree for the example problem. With reference to the embedded Markov chain in Figure 2.4, only transitions to immediate states are depicted in the stochastic tree. The bracketed state names, *e.g.,* [Well], indicate cyclic notations; in other words, when a bracketed state is reached, subsequent evolution starts from the point of the stochastic tree where that state is first located. Transition rates, instead of transition probabilities, are used to specify the probabilistic characteristics of the tree. A transition rate describes the number of occurrences of an event, per unit time, for a population of specific size; it ranges from zero to infinity.

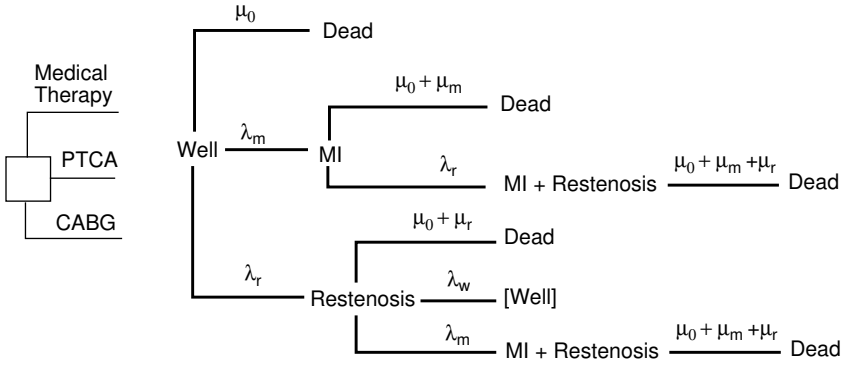


Figure 2.10 A stochastic tree. Parameters on the arcs indicate transition rates among the states.

Solutions

The solution to a dynamic decision model with stochastic trees determines the single optimal action or decision alternative over time; this is done by calculating and comparing the values achievable over time for the respective stochastic trees. The most common solution method is value iteration, analogous to that described in (EQ 1) and (EQ 2) for MDPs. In value iteration, the stochastic tree is rolled back. The expected value of each state is calculated by value expectation in the subtree with the state in concern as the root, along the branches from the leaves. The basic manipulations involved, as summarized in Figure 2.11, are based on the "memoriless" property of the exponential distributions [Taylor and Karlin, 1994]. More intuitively, a process waiting in state *s* until it makes a transition to a specific destination is equivalent to waiting in state *s* until a transition occurs and then determining the identity of the destination [Hazen, 1992].

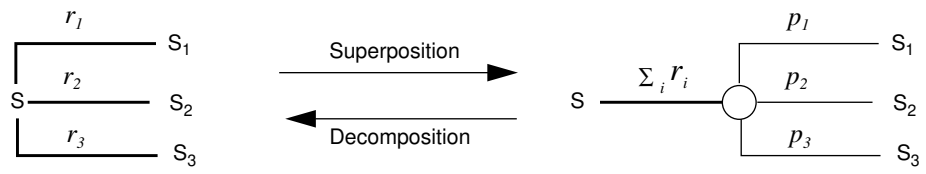


Figure 2.11 Basic manipulations in a stochastic tree. The fractions p_i's are defined by $p_i = r_i / \Sigma_i r_i$.

The expected values achievable for a specific action can be expressed in terms of the optimality equation as follows:

$$V_i^{(a)} = \frac{1}{\sum_j r_j} \cdot v_i^{(a)} + \sum_j p_j \cdot V_j^{(a)}$$

where $i, j \in \{\text{Well, MI, Restenosis, MI + Restenosis, Dead}\}$, $a \in \{\text{MedRx, PTCA, CABG}\}$, $v_s^{(a)}$ is the value achievable in state s per unit time, conditioned on action a, r_s is the transition rate from state s, and $p_s = r_s / \sum_s r_s$.

With reference to Figure 2.10, for instance:

$$V_{\text{MI+Restenosis}}^{\text{PTCA}} = \frac{1}{\mu_0 + \mu_m + \mu_r} \cdot v_{\text{MI+Restenosis}}^{\text{PTCA}} + V_{\text{Dead}}^{\text{PTCA}}$$

$$V_{\text{MI}}^{\text{PTCA}} = \frac{1}{\mu_0 + \mu_m + \mu_r} \cdot v_{\text{MI}}^{\text{PTCA}} + \frac{\lambda_r}{\mu_0 + \mu_m + \mu_r} \cdot V_{\text{MI+Restenosis}}^{\text{PTCA}} + \frac{\mu_0 + \mu_m}{\mu_0 + \mu_m + \mu_r} \cdot V_{\text{Dead}}^{\text{PTCA}}$$

If cycles appear in the state transition diagram of the embedded Markov chain, the evaluation must be done across several cycles until the numbers converge. Due to the decomposability of the exponential transition time functions, fewer calculations are required as compared to Markov cycle tree evaluation, and the evaluation terminates in finite time.

2.5 Planning in Artificial Intelligence

In classical AI planning, the dynamic decision problem consists of an initial state, a goal state, and a set of operators. The typical operators, originating from the planner STRIPS [Fikes and Nilsson, 1971], characterize actions or events (actions of nature) in terms of their preconditions and their effects. An operator ω can be described as a set of pairs $< \alpha, \beta >$, where α is a set of preconditions and β is a set of effects or postconditions. The preconditions are propositions or conditions that must hold before the operator can be applied; the postconditions are propositions or conditions that must hold after the operator is applied. Each proposition describes a subset of the states involved; the states may or may not be fully enumerated. In other words, each proposition describes an *attribute* or *dimension* of

the state definitions. A plan is a sequence of operators; it is the solution to the dynamic decision problem if the plan is applicable in the initial state, and the goal state is true after executing the plan [Tate et al., 1990].

Classical AI planning assumes deterministic action effects and absolute utility or value of the goal state. In other words, if the preconditions of an action operator are satisfied, after the operator is applied, its postconditions are guaranteed to be satisfied. Moreover, reaching a goal state or satisfying a set of objectives can either succeed or fail; there is no notion of optimality.

Recent research in decision-theoretic planning has led to a number of planning frameworks that incorporate decision theoretic methods. Many of these techniques address dynamic decision problems that can be described in terms of MDPs. These techniques, therefore, are directly relevant to this work.

In a decision-theoretic planner, an operator ω can be described as a set of triples $< \alpha, \rho, \beta >$, where α is a set of preconditions, β is a set of postconditions, and ρ is a probability indicating how likely β will be satisfied after ω is applied [Hanks, 1990] [Draper et al., 1994] [Kushmerick et al., 1994]. For instance:

Operator(PTCA) = {<{Restenosis}, 0.8, {Well}>, <{Restenosis}, 0.15, {Restenosis}>, <{Restenosis}, 0.05, {Dead}>}

Most of the research in AI planning focuses on efficiently representing and solving dynamic decision problems with large and complex state space. On representations, decomposable representation techniques of action effects or state descriptions are emphasized. On solutions, feasible ways of trading off optimality for speed, and improving such trade-offs, are addressed.

In the general SMDPs and dynamic decision modeling approaches described earlier, problem formulation and solution are in terms of explicitly and exhaustively enumerated action space and state space. In AI planning, however, problem formulation and solution involve on-demand generation of the action space and state space. AI planners, therefore, usually need to address the *ramification* problem, the *qualification* problem, and the *frame* problem. The ramification problem deals with the representation of complex action effects. The qualification problem concerns the necessary preconditions for an action to achieve specific effects.The frame problem addresses the invariant conditions after applying an action.

Solutions

Solution methods in classical AI planning are based on *heuristic search* [Korf, 1987]. Guided by a set of heuristics, plan generations often involve action and state spaces at multiple levels of abstraction. The appropriate heuristics depend on the planning ontology of the framework.

In decision-theoretic planning, many solution methods that aim to provide fast answers at the cost of optimality are based on generating partial plans and system-

atically improving on such plans. This approach is similar to the policy iteration method in MDPs [Howard, 1960].

Figure 2.12 depicts a partial search space for the example problem. The most common solution approach is to heuristically determine the size of the search space and the direction of search. The objective of the search is to find an optimal course of action. Optimality is usually defined in terms of maximizing probabilities or minimizing time. In the diagram, the oval nodes represent the states, the square nodes the actions. Links leading into the chance nodes indicate conditional dependencies, links leading into the action nodes indicate informational dependencies. Transition probabilities and other parameters such as transition times are associated with the conditional links. A decision stage spans between any two actions separated by a state. The propositions or state attributes that are relevant are labelled besides the states; unmentioned propositions are assume to be either irrelevant or unchanged from possible previous decision stage.

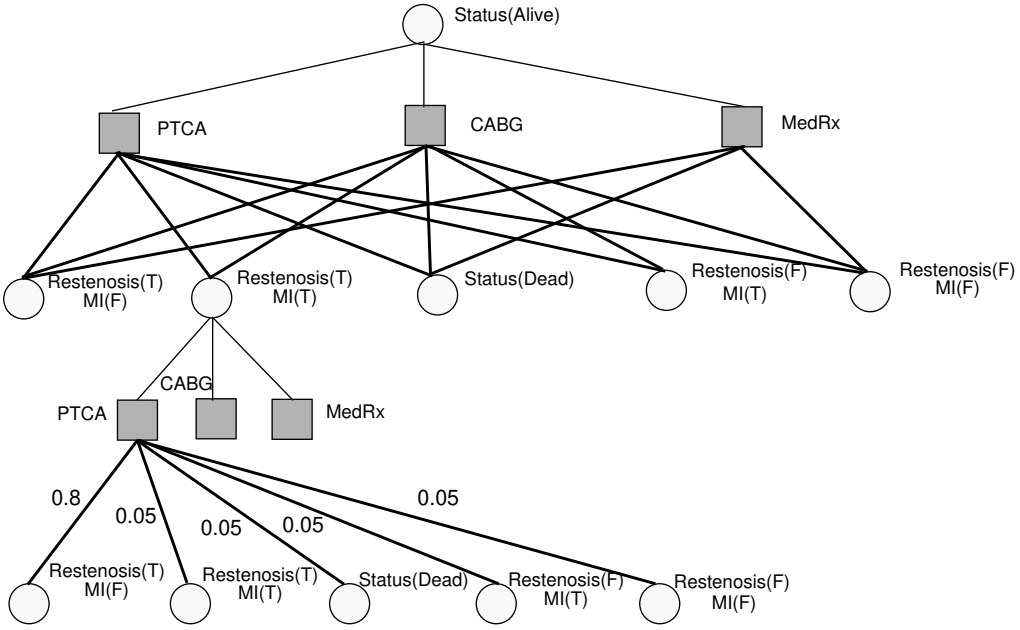


Figure 2.12 A partial search space for an AI planner.

2.6 Summary

We have so far sketched the basic framework in three approaches to dynamic decision making under uncertainty. In the next chapter, we shall examine the commonalities, the differences, and the strengths and weaknesses of the techniques in terms of their decision ontologies and problem structures, solution methods, and available metrics for assessing model quality.

3 A Unifying View

Based on the survey in Chapter 2, we present a unifying view for reasoning about the class of dynamic decision problems addressed in this work. We first define a uniform task definition of dynamic decision making under uncertainty, and then identify a common vocabulary for the existing methodologies. This unifying view establishes a basis for comparing and contrasting the different techniques. Three major arguments result from this exercise. First, there are well-defined correspondence among the different methodologies. Second, by building on their common basis, we can integrate the desirable features of these techniques to form a sound and flexible framework. Third, we can extend this integration to support a new paradigm of dynamic decision making under uncertainty; the central ideas involved multiple perspective reasoning and incremental language extension. We claim that this paradigm generalizes current approaches.

3.1 A Uniform Task Definition

The three major tasks in dynamic decision making under uncertainty are problem formulation, solution, and analysis. Problem formulation defines the decision environment or context by specifying the relevant parameters and their interrelationships. Problem solution prescribes a course of optimal action given the decision context. Problem analysis supports validation and verification of the decision parameters and solutions. The three tasks may interleave and may repeat many times for a given problem.

3.1.1 Problem Formulation

Formulating a dynamic decision problem begins with representing and specifying the decision factors and constraints involved. Examples of decision factors include

alternative actions and strategies, possible state transitions, chance events that constitute the state transitions, and relevant probability distributions with respect to time. Examples of decision constraints include applicability conditions of the actions, validity conditions of the states, and logical statements about the states and chance events. This modeling task is supported by the decision ontology or vocabulary. The defining characteristics of a decision ontology are its expressiveness, its succinctness, and its transparency.

Expressiveness of a decision ontology defines how accurately we can specify the factors and constraints in a decision situation; it determines the types and organization of the relevant parameters and their interrelationships. For example, a decision ontology may include only actions and states as basic concepts and temporal dependences as basic relations, while another may also include events and probabilistic dependences. Similarly, a decision ontology may represent actions as individual concepts, while another may support reasoning with classes of actions.

Succinctness of a decision ontology determines how easily we can specify the factors and constraints in a decision situation. For example, in SMDPs and dynamic decision models, all the states in a dynamic decision problem have to be explicitly enumerated; in AI planning, only the relevant conditions or attributes of the state descriptions are specified in the operators. Some features that would contribute to succinctness include decomposable or hierarchical descriptions, and explicit constraint declarations. An example of hierarchical description is the taxonomic organization of decision parameters such as actions, states, and chance events. An example of constraint declaration is a statement analogous to a logical and quantified sentence or well-formed formula in predicate calculus.

Transparency of a decision ontology reflects how efficiently we can draw inferences on the information involved in a decision situation, and how easily we can comprehend such information. An ontology with a simple syntax or semantics does not necessarily imply that it is transparent; unless it is succinct, it may require many terms and sentences to specify a small piece of knowledge. Some features that would contribute to transparency include constant representation and multiple perspective visualization.

Constant representation means the same piece of information is interpreted in the same way or with minimal changes at all times. In order to reduce computational costs or to promote precision, for instance, the same qualitative structure in a model may be reused with different quantitative bindings or values. While justified and preferred sometimes, such a practice often obscures the content of the information involved, especially when the changes involved are qualitative or implicit.

Multiple perspective visualization facilitates access to the same piece of information in different ways, depending on the situations or needs. This is analogous to viewing the world with different pairs of lenses, each providing a perspective most natural to a particular task.

3.1.2 Problem Solution

The solution to a well-formed dynamic decision problem is a course of action that optimize some objectives in the decision context. The solution format can be *closed-loop* or *open-loop*. A closed-loop solution solves the problem with respect to a specific initial point and a specific end point. A plan generated by a classical AI planner for a starting state and a goal state is a closed-loop solution. An open-loop solution solves the problem with respect to all possible initial points and all possible end points. A policy determined in a semi-Markov decision process (SMDP) is an open-loop solution.

The solution task involves choosing an appropriate solution method and executing the method accordingly. A solution method is a computational model that manipulates the information in the decision context. Unlike the modeling task in problem formulation, problem solution assumes a complete, explicit organization of the decision factors and constraints. In other words, given a well-formed model, the solver does not need to know how the information organization arises in the first place.

3.1.3 Problem Analysis

Both the formulation and the solution of a dynamic decision problem may require analysis of the decision factors and constraints involved. In problem formulation, analysis ensures accuracy and conciseness of the model. In problem solution, analysis supports revision and refinement of the result, via revision and refinement of the problem formulation; it also illuminates important characteristics of the problem and the solution method.

The most common analysis technique is sensitivity analysis. Sensitivity analysis can be divided into deterministic sensitivity and stochastic sensitivity. Deterministic sensitivity is reflected through adjustments of the qualitative information involved; it determines which uncertain events affect the choice and the value of the decision. Stochastic sensitivity is reflected through adjustments of the quantitative information involved; it reveals how uncertain events affect the choice and value of the decision.

3.2 A Common Vocabulary

In addition to a uniform task definition, we need a common basis to compare and contrast the existing techniques. Just as first-order predicate calculus serves as the basis for most deterministic knowledge representation research in AI, we propose that SMDPs, as described in Section 2.3, serve the same role for the class of decision making techniques relevant to this work. We substantiate this argument in the next section with a detailed analysis of the techniques described in Chapter 2. We shall interpret the language components of the other frameworks in terms of the

mathematical constructs of SMDPs. In other words, we shall define a set of interpretation or mapping functions for each technique concerned as shown in Figure 3.1.

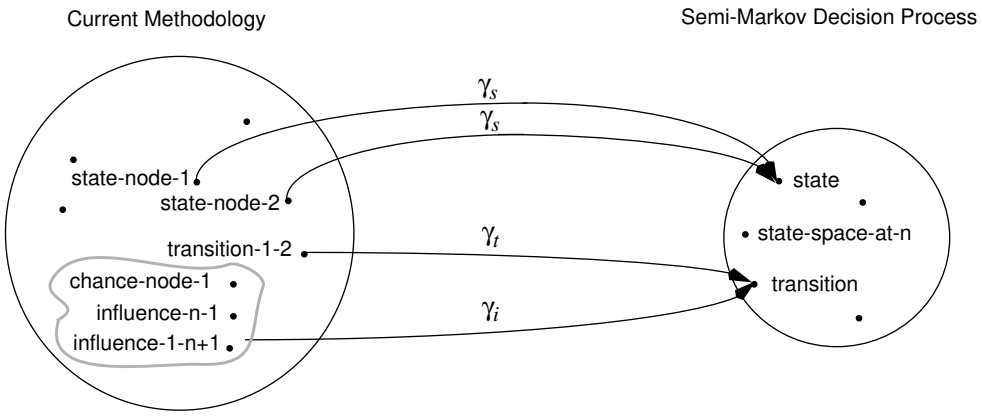


Figure 3.1 Interpretation by mapping. Each arrow type indicates a transformation function or correspondence rule.

3.3 An Analysis

Based on the uniform task definition and the common vocabulary, we examine the capabilities and limitations of the techniques outlined in Chapter 2. In particular, we compare and contrast the decision ontology and problem structure involved in, and the solution methods and model quality metrics available for the different techniques. For each technique, the first and second dimensions support problem formulation, and the third and fourth dimensions support problem solution and problem analysis.

3.3.1 Decision Ontology and Problem Structure

In this section, we analyze the decision ontology and the problem structure of the existing techniques. The decision ontology comprises the basic language components of each technique. The problem structure describes how these components can be combined together to formulate a dynamic decision problem, and the assumptions and constraints involved in the combination. These two dimensions illuminate the expressiveness, succinctness, and transparency of a language, thereby indicating its effectiveness in supporting problem formulation.

Semi-Markov Decision Processes

Recall from Section 2.3, an SMDP has the following basic components: a time index set $\boldsymbol{T}$, an action space $\boldsymbol{A}$, a state space $\boldsymbol{S}$, a set of one-step transition functions

with PMFs $q_{ij}^{(a)}(.)$ and CDFs $Q_{ij}^{(a)}(.)$ defined in terms of either 1) a set of transition probabilities $P_{ij}^{(a)}(.)$ and a set of holding times with PMFs $h_{ij}^{(a)}(.)$ and CDFs $H_{ij}^{(a)}(.)$, or 2) a set of conditional transition probabilities $p_{ij}^{(a)}(.)$ and a set of waiting times with PMFs $w_i^{(a)}(.)$ and CDFs $W_i^{(a)}(.)$, and a set of value functions $v_i^{(a)}(.)$. The stochastic processes defined on these basic components, based on the embedded Markov chain $\{S(T_k);\ T_k \in \boldsymbol{T}\}$, include a decision process $\{D(t);t \in \boldsymbol{T}\}$ and a semi-Markov reward process $\{S(t);t \in \boldsymbol{T}\}$.

The descriptions given so far summarize the basic definitions for non-homogeneous SMDPs. Although we will not address them in detail in this work, more general classes of SMDPs provide direct expression of the following concepts[1]:

- Varying action space and state space with respect to time, *e.g.,* action a is applicable only at time t and state s is valid only at time t:

$$\boldsymbol{A} = \bigcup_{t \in \boldsymbol{T}} \boldsymbol{A}_t$$

$$\boldsymbol{S} = \bigcup_{t \in \boldsymbol{T}} \boldsymbol{S}_t$$

- Varying action space with respect to state, *e.g.,* action a is applicable only in state s.

$$\boldsymbol{A} = \bigcup_{s \in \boldsymbol{S}} \boldsymbol{A}_s$$

- Varying action space with respect to both state and time, *e.g.*, action a is only applicable in state s at time t:

$$\boldsymbol{A} = \bigcup_{t \in \boldsymbol{T}} \boldsymbol{A}_{\boldsymbol{S}_t}$$

where

$$\boldsymbol{A}_{\boldsymbol{S}_t} = \bigcup_{s \in \boldsymbol{S},\, t \in \boldsymbol{T}} \boldsymbol{A}_{s,t}$$

- Partially observable states, which means that the decisions $\{D(t);\ t \in \boldsymbol{T}\}$ are made based on some observations $\{\Phi(t);t \in \boldsymbol{T}\}$, but not the states $\{S(t);\ t \in \boldsymbol{T}\}$. The states and the observations, however, are related by conditional probability distributions $P\{\Phi(t) \mid S(t)\};\ t \in \boldsymbol{T}$. Figure 3.2 shows the information flow diagram of a partially observable SMDP.

1. Some of these extended definitions are included in Appendix B.

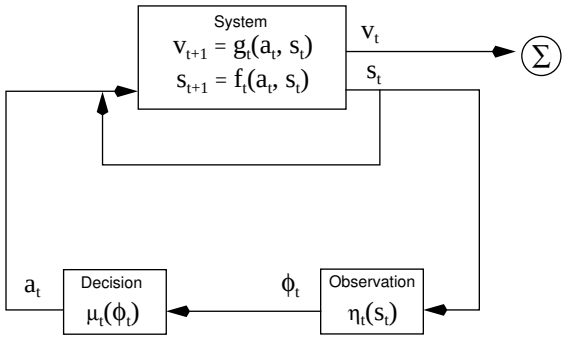


Figure 3.2 Information flow in a partially observable semi-Markov decision process. At each time t the decision maker observes the current observation ω_t, which is conditionally dependent on state s_t through $\phi_t = \eta_t(s_t)$, and applies action $a_t = \mu_t(\phi_t)$ that depends only on the observation. A value v_t is accumulated in the mean time.

The SMDPs, therefore, provide a very rich decision ontology for formulating dynamic decision problems. The general action space and state space definitions provide flexible ways to conceptualize the decision alternatives and their effects. The one-step transition functions reflect the probabilistic and temporal dependences among the decision parameters; they also allow uncertainty to be expressed in both the transition destinations and the time before transitions. The observation space adds another dimension to characterizing the decision environment by handling partial information.

Given the decision ontology, we examine how some common assumptions and constraints involved in formulating dynamic decision problems are incorporated in SMDPs. In this section, we introduce some of the issues concerned: time-indices and decision horizon, sequential separable decisions, decomposable state descriptions, strategic constraints, and granularity of transition characteristics. These issues will be addressed for other techniques as well.

In the following discussion, we assume that the *world* refers to the environment from which the decision maker gathers information before taking an action, and to which the action effects are directed. A patient's pathophysiological status is the world in an SMDP for treatment management.

Time Indices and Decision Horizon

The time indices of a dynamic decision problem can be discrete or continuous. The horizon of a dynamic decision problem can be finite or infinite. In finite horizon problems, the optimal policy is determined over a finite number of time units or decision stages *N*, *e.g.,* deciding on the best course of action for 65 year-old patients in the 5 years immediately following a PTCA. In infinite horizon problems, the decision endpoint may be at an arbitrarily distant future, *e.g.,* when a large

cohort of patients all die, using a small cycle time unit. Although we only address discrete-time dynamic decision problems in this work, for an SMDP, the time index set $\boldsymbol{T}$ can be discrete or continuous, the decision horizon can be finite or infinite.

Sequential Separable Decisions

In an SMDP, a decision is an action chosen at a single point in time *t*; uncertainty about the decision is represented by a random variable $D(t)$ over the actions at each time point *t*. Decisions are made in sequence. Based on the general definitions above, the decisions $D(t)$ may involve different sets of alternative actions for different states $\boldsymbol{A}_s$, for the same states at different time points $\boldsymbol{A}_t$, or for different states at different time points $\boldsymbol{A}_{\boldsymbol{S}_t}$.

Decomposable State Descriptions

In an SMDP, a state is a complete description of the world at a single point in time *t*; uncertainty about the world is represented by a random variable $S(t)$ over the states at each time point *t*. Although it is often useful to define a state along several distinguishable dimensions or attributes, the vocabulary does not formally include such *state attribute variables* $\{\theta_k(t); k = 1, 2, 3, \ldots, K, t \in \boldsymbol{T}\}$, where $\theta_k(t) \in \Theta_{k,t} = \{0, 1, 2, 3, \ldots\}$ is a state attribute variable at time *t*. A state attribute corresponds to a proposition or condition in the state descriptions of AI planning mentioned in Section 2.5.

The state space at time *t* can be defined as a product of the state attribute spaces at time *t*:

$$\boldsymbol{S}_t = \prod_k \Theta_{k,t} \qquad \textbf{(EQ 3)}$$

In other words, at any time *t*, a state *s* is an element of the Cartesian product of a set of state attribute variables:

$$s_t \in \Theta_{1,t} \times \Theta_{2,t} \times \Theta_{3,t} \times \ldots \times \Theta_{K,t}$$

The above definition, however, is sometimes too general. Domain-specific knowledge often imposes further restrictions on how the state space can be defined in terms of the state attribute variables. Consider the state of a patient that is defined in terms of whether he is alive or dead, whether he has MI or not, and whether he has restenosis or not. If the patient's ***status*** is *dead*, it does not matter whether he has either or both of the diseases concerned. The corresponding state space will be smaller than the Cartesian product space of the three state attribute variables involved. Hence, an alternate form of (EQ 3) is as follows:

$$\boldsymbol{S}_t = g(\Theta_{1,t}, \Theta_{2,t}, \Theta_{3,t}, \ldots, \Theta_{K,t}) \qquad \textbf{(EQ 3')}$$

where *g* is a function conforming to the imposed domain-specific restrictions.

Strategic Constraints

In the example problem introduced in Section 2.2, all the alternate treatments are assumed to be applicable at every decision stage; the decisions are assumed to be independent. The independent decision assumption may not be true in general. The efficacy of PTCA may be lower if the patient had gone through another PTCA before. A patient could only go through three CABG procedures. A CABG could only be followed by a PTCA or medical therapy. These considerations are examples of strategic constraints in dynamic decision problems.

As mentioned earlier, SMDPs allow definitions of action space that varies with time or state or both, specifying the applicable actions with respect to the time or the state of the system or both. Hence, most of the strategic constraints can be expressed in terms of extra states in the embedded Markov chain. For instance, if CABG can only be applied three times, extra states are needed to keep track of the number of CABG administered. In general, a constraint on the applicability of the action needs to be encoded as a state attribute variable. Figure 3.3 shows a state transition diagram for an extension of the example problem in which CABG is applicable up to two times. The state attribute *sick* is defined as either having MI or restenosis or both. CABG is not applicable in the shaded states; the other two actions, PTCA and MedRx, are applicable in all the states.

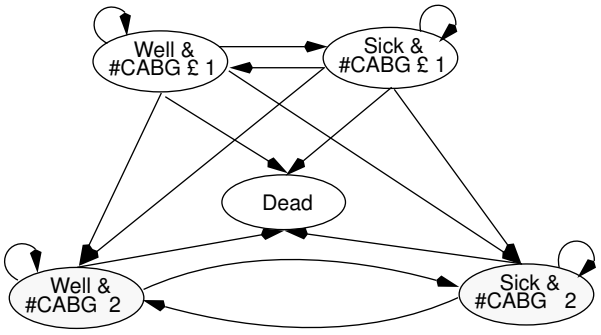


Figure 3.3 State transition diagram for example problem. The action CAGB is not applicable in shaded states shown. All other actions are assumed to be applicable in every state.

Some solution methods such as those employing forward induction, where the decision problem is solved from the initial stage and moving forward in time, can make use of external devices to keep track of the strategic constraints. An example is a counter for the number of each action applied so far. This would alleviate the problem of "state explosion" if many strategic constraints are involved. The general problem of incorporating constraints as new dimensions, however, is inherent in all Markov processes due to the "memoriless" nature of their state space.

Granularity of Transition Characteristics

All the possible effects of an action are captured in the transition functions and the next reachable states. As mentioned, the states and the sets of transition functions conditional on each action are theoretically independent of those conditional on other actions. In other words, the effects of the actions can be modeled separately. The SMDP ontology, however, does not support decomposing the transitions into constituent events. The transition functions may be difficult to assess directly. For instance, without explicitly considering if a patient has MI or restenosis or both, it can be confusing to keep track of all the ways a patient can become *sick*.

Dynamic Decision Modeling

SMDPs are the common theoretical basis for the three types of dynamic decision models addressed [Leong, 1993]. Consider again the example problem in Section 2.2 and its dynamic decision models described in Section 2.4. With reference to the definition of an SMDP in Section 2.3, in each model:

- The set of actions $\boldsymbol{A} = \{\text{MedRx, PTCA, CABG}\}$.
- The semi-Markov reward process, with time index set $\boldsymbol{T} \subseteq \{0, 1, 2, \ldots\}$ for the dynamic influence diagram and the Markov cycle tree, and $\boldsymbol{T} \subseteq [0, +\infty]$ for the stochastic tree, is defined by:
  1. An embedded Markov chain, as illustrated in Figure 2.4, with state space $\boldsymbol{S} = \{\text{Well, MI, Restenosis, MI + Restenosis, Dead}\}$;
  2. Three sets of transition probabilities $P_{ij}^{(a)}(t)$ among the states in $\boldsymbol{S}$, conditional on the actions in $\boldsymbol{A}$;
  3. Constant holding times with cumulative distribution functions (CDFs) $H_{ij}^{(a)}(m, t) = \mathbf{1}(m-1)$, where $\mathbf{1}(m-1)$ is a step function at duration $m = 1$ (in any unit) for the dynamic influence diagram and the Markov cycle tree, and exponential holding times with CDFs $H_{ij}^{(a)}(m, t) = 1 - e^{-\mu_i m}$, where μ_i are the state-dependent transition rates for the stochastic tree; and
  4. Three sets of value functions $v_i^{(a)}(m)$, corresponding to the amount of QALE expected in each state in $\boldsymbol{S}$, conditional on the actions in $\boldsymbol{A}$.

While sharing the general theoretical basis, the three dynamic decision modeling techniques capture and reflect the underlying mathematical information in different ways; different assumptions are also adopted.

One major advantage of dynamic decision modeling is the graphical modeling languages involved. Although different types of decision models explicitly display different types of information, all the graphical tools provide much insights into the decision context.

Dynamic influence diagrams

A dynamic influence diagram has the following basic components: a set of decision nodes, a set of chance nodes, a set of value nodes, a set of probabilistic influence links, a set of informational influence links, a set of conditional probabilistic distributions associated with the chance nodes, and a set of value functions associated with the value nodes.

As shown in Figure 3.4, a dynamic influence diagram depicts a sequential view of the interactions among the action space, the state space, and the value functions in an SMDP. The decision node represents the action space at a particular time t or decision stage n in an SMDP. The notion of states is not explicitly included in the decision ontology, but the states in the embedded Markov chain can be represented as the outcomes of a set of specially designated chance node, called *state variable nodes*. The probabilistic influence links between each pair of state variable nodes indicate, but not explicitly illustrate as in a state transition diagram, transitions among the states in two successive time points or decision stages. The conditional probability distribution associated with each state variable node details the set of PMFs $q_{ij}^{(a)}(.)$ of the one-step transition functions involved. A set of extra chance nodes can be incorporated to represent a partially observable SMDP.

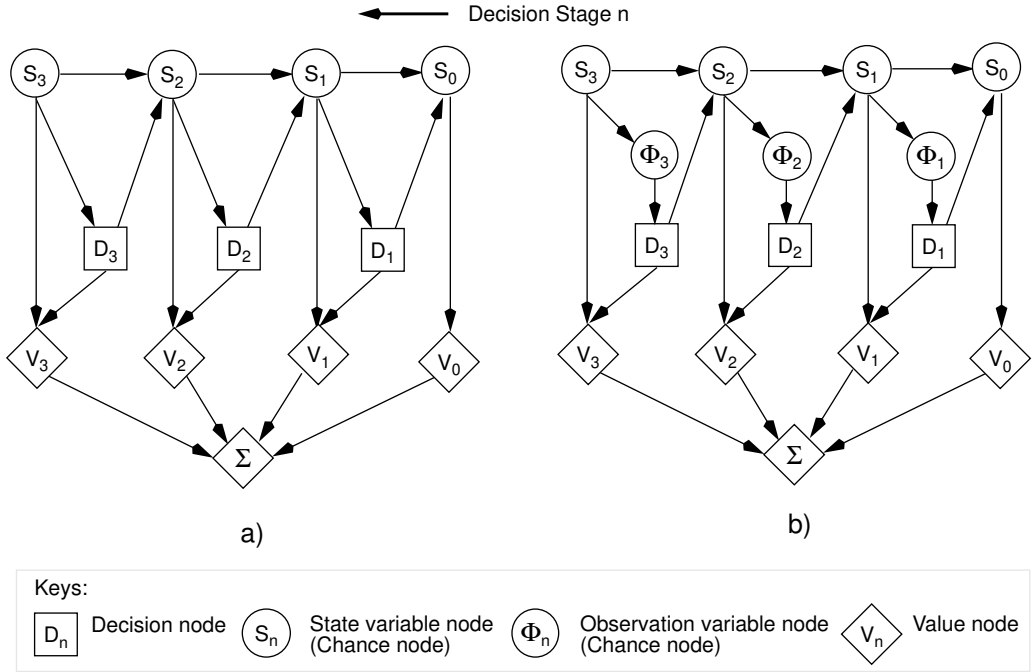


Figure 3.4 Dynamic influence diagrams representations of semi-Markov decision processes. a) Structure of semi-Markov decision process. b) Structure of partially observable semi-Markov decision process.

Time Indices and Decision Horizon

The time indices are discrete in dynamic influence diagrams. The inter-transition intervals are usually assumed to be constant, although varying intervals can also be incorporated. In other words, the underlying stochastic processes are usually Markov, instead of semi-Markov in nature. Since the distinct decision stages are explicitly depicted, the decision parameters and transition functions are separately specified for the conditional distributions in each stage. Therefore, dynamic influence diagrams are usually feasible only for finite horizon problems.

Sequential Separable Decisions

As shown in Figure 2.7 and Figure 3.4, dynamic influence diagrams explicitly display the sequential nature of the decisions in dynamic decision problems. In these diagrams, the alternative actions embedded in each decision node constitute the entire action space $\boldsymbol{A}$ applicable at that decision stage. From these perspectives, however, applicability of the actions with respect to the different states are implicitly encoded in the conditional distribution of the corresponding state variable node. For instance, if action a is not applicable in state s, state s is an absorbing state conditional on action a. In other words, the conditional probabilities from state s at any decision stage n to the same state s at decision stage n-1 are unity, those from all other states to state s are zero, and state s conditional on action a does not accrue any value.

An alternate way to represent varying action space, be it dependent only on the states, the time, or both, is shown in Figure 3.5. This diagram incorporates the same amount of information as the one in Figure 3.4 a), but depicts more clearly the varying nature of the action space. The same organization is also suitable for formulating dynamic decision problems with separable state space, *i.e.,* there are subsets of states that are inaccessible from the others. Note that an optimal decision in this formulation involves choosing an action from among all the decision nodes within a decision stage; the expected values for the alternatives of these decision nodes are calculated separately.

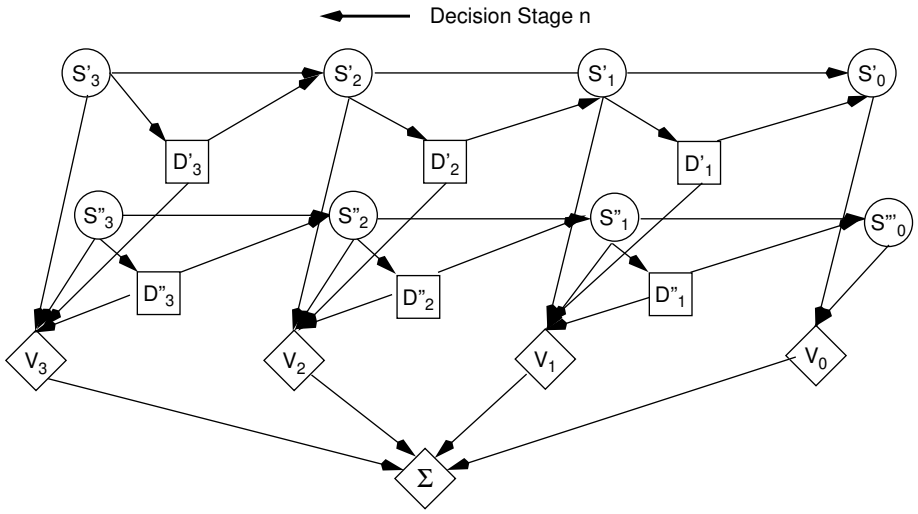


Figure 3.5 Action- and state-separable dynamic influence diagram. The state variable nodes S'_n and S''_n partition the state space, and the decision nodes D'_n and D''_n partition the action space at decision stage n.

Decomposable State Descriptions

In a dynamic influence diagram, independent state attribute variables can be dealt with separately. For instance, the model for the example problem in Figure 2.7 explicitly shows that the state variables can be decomposed into their state attribute variables ***status***, ***MI***, and ***restenosis***; the corresponding transition characteristics can thus be specified separately before being combined together. The general structure of such decompositions is shown in Figure 3.6.

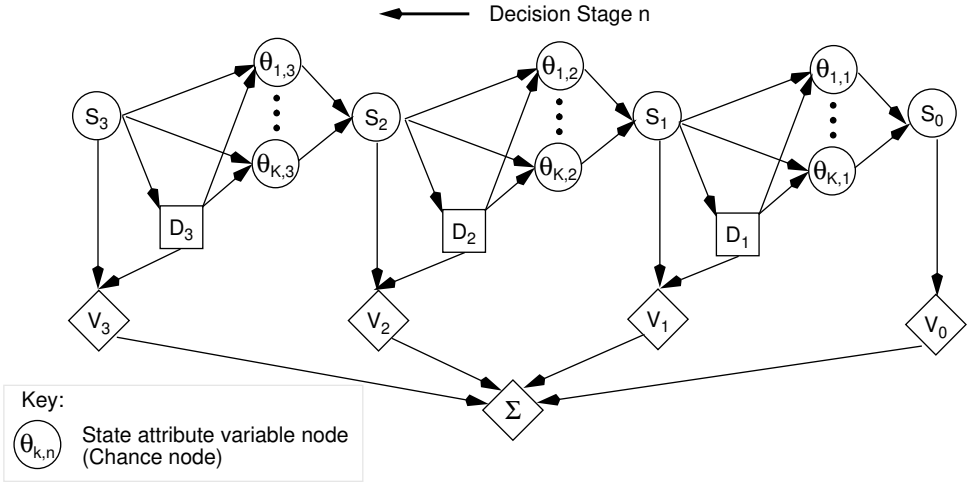


Figure 3.6 A dynamic influence diagram with decomposable state descriptions.

Strategic Constraints

In general, the capabilities of a dynamic influence diagram for handling strategic constraints are similar to those of an SMDP. For instance, since the conventional solution methods for dynamic influence diagrams usually employ backward induction, dependent efficacies of and numeric restrictions on specific actions can be handled only with extra state attribute variables.

Granularity of Transition Characteristics

The qualitative structure of a dynamic influence diagram is at a higher level or abstraction than a state transition diagram. Unless the action-separable organization as shown in Figure 3.5 is used, a dynamic influence diagram does not usually explicate the distinct effects of different actions at a particular decision stage; such information is revealed only through the corresponding conditional distributions associated with the state variables.

With chance nodes as part of its decision ontology, however, a dynamic influence diagram can represent the transitions in more details. In addition to explicitly modeling the state attribute variables as shown in Figure 3.6, other possible chance events that might influence them can also be incorporated. The structure of the value functions can also be explicitly depicted in these models. For a particular decision stage, the general structure of such decompositions is shown in Figure 3.7. The ability of separately displaying these decomposed parameters allow the underlying conditional distributions to be independently assessed.

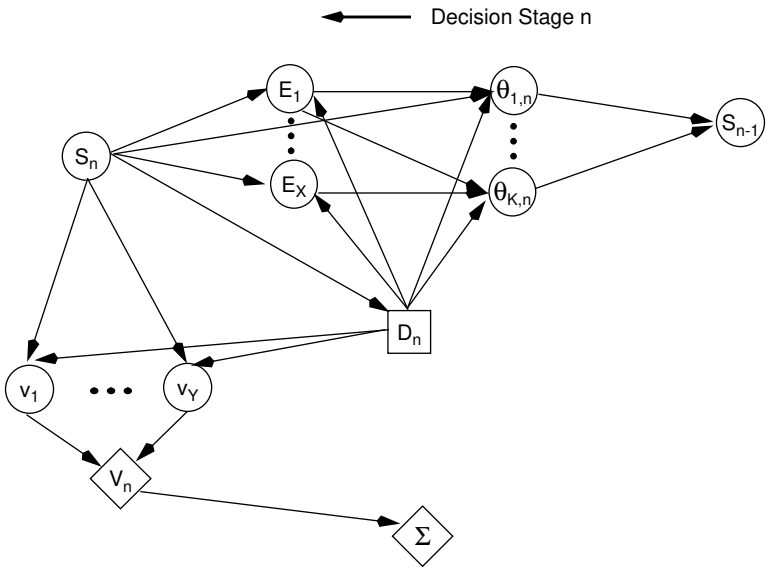


Figure 3.7 A dynamic influence diagram with chance events constituting the state transitions and decomposed value functions. All the circles are chance nodes.

Markov cycle trees and Stochastic trees

Since stochastic trees are the continuous-time analogs of Markov cycle trees, we discuss these two techniques together. A Markov cycle tree or a stochastic tree has the following basic components: a set of decision nodes, a set of state nodes, a set of chance nodes, a set of conditional probability distributions associated with the chance nodes, and a set of value functions associated with the state nodes. The conditional probability distributions in a stochastic tree is in the form of constant transition rates.

A Markov cycle tree or stochastic tree models the consequences of a specific action by depicting a more detailed view of the state transition diagram of the embedded Markov chain. The tree explicitly displays the possible effects of the action, in terms of a set of chance nodes, which are implicitly captured in the transition arrows of the embedded Markov chain. This correspondence is more obvious if we remove all the intermediate chance nodes by conditional expectations as shown in Figure 3.8.

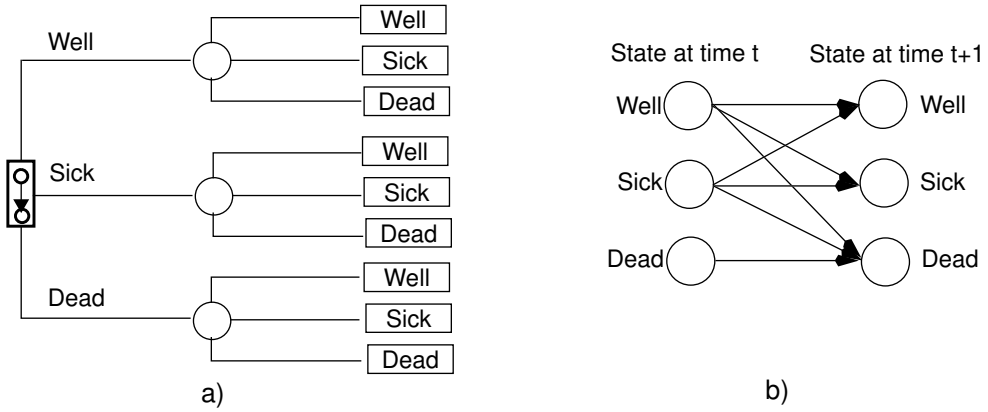


Figure 3.8 Comparing a) a Markov cycle tree and b) a state transition diagram.

Time Indices and Decision Horizon

The time indices are discrete in Markov cycle trees, and continuous in stochastic trees. The constant holding times in the former and the constant transition rates in the latter are very strong assumptions; they cannot express situations where the transition functions depend on the time duration since entering a state. After a PTCA, for instance, if the patient is well, it is three times more likely that he will develop restenosis in the next 6 months than later. This cannot be easily captured in the models; extensions such as *tunnel states* in the cycle trees are necessary [Sonnenberg and Beck, 1993]. Each tunnel state is a transient state that is accessible in a fixed sequence; it can represent a single unit of time duration spent in the original state. Figure 3.9 shows a (partial) state transition diagram for the tunnel

state organization in the above example. The tunnel states are represented as shaded circles. Assuming the cycle length or inter-transition interval is three months, the first tunnel state represents the first three months after PTCA, the second the next there months, and the third any time, at three-month intervals, six months after the treatment.

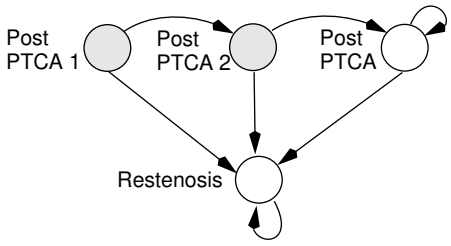


Figure 3.9 Using tunnel states to model duration dependent transitions.

Markov cycle trees can model both finite and infinite horizon problems. In stochastic trees, state transitions can occur at any time corresponding to some exponential distributions, hence they are valid only for infinite horizon problems.

Sequential Separable Decisions

In the tree-based dynamic decision models, all the states are assumed to be valid at all times; the action associated with each Markov cycle tree or stochastic tree is assumed to be applicable in all states. Since the decision ontology does not include decision nodes, the sequential nature of the decisions cannot be concisely represented. For instance, the Markov cycle tree and the stochastic tree for the example problem as shown in Figure 2.9 and Figure 2.10 do not include sequential decisions; they do not consider how backup or alternate treatments may affect the patient's prognosis. Such "cross-over" information can only be incorporated as additional state attributes. The combinations of all subsequent actions and their possible consequences between any two successive transitions would have to be modeled as separate states in the cycle tree. Figure 3.10 shows such an extended Markov cycle tree for the example problem. CABG can be performed as back-up treatment for other actions; the implications for staying well after a CABG is different from staying well after other treatments.

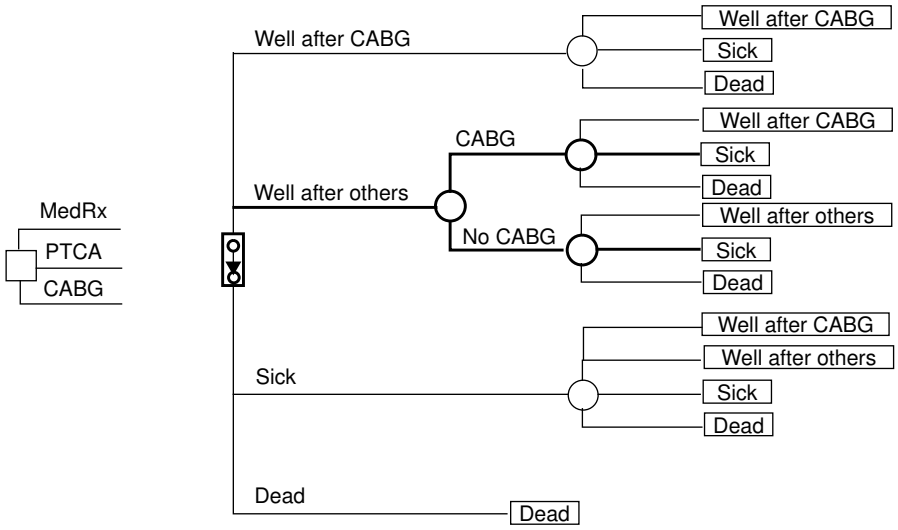


Figure 3.10 A Markov cycle tree involving cross-over strategy with different consequences.

Decomposable State Descriptions

In a Markov cycle tree or a stochastic tree, state attribute variables are not graphically depicted as chance nodes. In a Markov cycle tree, state attributes handling is closely related to the forward induction based solution methods. The state attributes are defined in a set of logical or algebraic expressions as *bindings* associated with some of the outcomes of the chance nodes. The outcomes in question have bindings that act like counters. For instance, in Figure 2.9, the outcome MI of the chance node indicating the presence or absence of MI has an associated binding MI := 1. When that particular branch is traversed, the binding is set on; the states represented as leaf nodes in a Markov cycle tree are determined by logical or algebraic expressions in terms of the bindings such as:

$$\text{MI+Restenosis} := (\text{MI} = 1) \wedge (\text{Restenosis} = 1) .$$

As in an SMDP, state attribute variables are not handled formally by the decision ontology. The backward induction based solution methods employed with stochastic trees also do not incorporate external devices such as bindings.

Strategic Constraints

In a Markov cycle tree or a stochastic tree, the same action, which may include one or more diagnostic tests or treatments in the medical domain, is assumed at every decision point. In other words, dynamic decision models employing these techniques can only model problems that compare the effectiveness of fixed strategies, but not those that determine an optimal sequence of possibly varying decisions.

Numeric restrictions on the actions, however, can be incorporated into a Markov cycle tree as bindings as described earlier. This avoids the extra states needed to incorporate such constraints. As mentioned, the bindings mechanism is specific to forward induction based solution methods, and cannot be generalized.

Granularity of Transition Characteristics

As mentioned earlier, a Markov cycle tree or stochastic tree depicts the transitions in more details. In addition to the chance events constituting the transitions, the different paths or combinations of the corresponding outcomes that constitute such transitions are also displayed. As an example, consider the thickened lines shown in Figure 3.10. Starting at any state *i* from the root node, the expectation of probabilities associated with all the paths leading to the same state node *j* constitutes the one-step transition function $q_{ij}^{(a)}(.)$ from *i* to *j*.

Comparing to the influence diagram perspective, the tree structure is also good for modeling modular, asymmetric pieces of knowledge. Again with reference to Figure 3.10, the different subtree structures for the well states indicate that CABG is not considered as a backup strategy if the original treatment is CABG itself. In a dynamic influence diagram, such asymmetry will be encoded in the conditional distributions, but not revealed at the structural level.

AI Planning

The basic decision ontology of an AI planner includes a set of operators, a set of propositions or conditions, and a set of states. An operator corresponds to an action in an SMDP, together with its applicability constraints and effects. The preconditions of an operator specify the subset of states in which the action is applicable; the postconditions of an operator specify the subset of states that are reachable after taking the action. Together the preconditions and the postconditions determine the set of transition functions associated with the action. In classical AI planning, the transition functions are deterministic; in decision-theoretic planning, the transition functions are stochastic. A proposition corresponds to a state attribute in an SMDP. A state is a complete description of the world at a single point in time; a collection of states that describe the world at different levels of abstraction at the same time may be defined simultaneously [Mansell, 1993]. The goal state corresponds to a value function in an SMDP. There is usually no explicit notion of time except for sequencing order of the operators.

Time Indices and Decision Horizon

For AI planning, the time indices are usually discrete and the planning horizon is often finite.

Sequential Separable Decisions

Most AI planning research deals with complex world states and actions; it may be very difficult or even impossible to enumerate all the states involved in a planning problem. The rich and complex operator representations simplify the specifi-

cation of action applicability conditions and effects. Without additional aids such as dependency diagrams, however, the sequential and separable nature of the actions are difficult to visualize.

Decomposable State Descriptions

In an AI planning problem, the states are usually defined in terms of a set of propositions. Problem formulation may or may not involve a complete enumeration of the state space.

Strategic Constraints

Besides the basic components described earlier, AI planning frameworks vary in their ontologies. Strategic constraints, however, are usually explicitly represented. Most of the constraints involved in AI planning are either captured in the operators or represented with external modules. *e.g.,* temporal databases. While the constraints are explicitly expressed, additional inferences may be necessary to reason about them.

Granularity of Transition Characteristics

In AI planning, the transition functions are defined in terms of the state attributes affected by the actions. There is no further breakdown of the transitions into chance events that would in turn influence the state attributes.

3.3.2 Solution Methods

Solving a dynamic decision problem assumes that the problem is completely and properly formulated in the respective framework. If more information is needed during the process, the solution method will determine and direct the derivation of such information.

Semi-Markov Decision Processes

A variety of solution methods are available for SMDPs. The most common algorithms are based on the *value iteration* method of the dynamic programming or Bellman optimality equation. Other methods include policy iteration, linear programming, *etc*. Applicability of the different solution methods depends on certain characteristics of the problem, *e.g.,* constant discount factor, stationary policy, and homogeneous transition functions. A more detailed discussion on these techniques can be found in Chapter 7 and Appendix B.

Dynamic Decision Modeling

Solving a dynamic decision model for the optimal policy is also called evaluating the model. Most of the evaluation algorithms are based on the *value iteration* method of the dynamic programming or Bellman optimality equation of semi-Markov decision processes, the discrete time Markov version of which is shown in (EQ 2') in Section 2.3. This method is based on the notion of *backward induction*.

All possible state evolutions over time have to be explicitly enumerated in the models, and hence considered by the evaluation algorithms. There are, however, calculations on optimal substructures that can be reused, thereby reducing the overall amount of calculations involved.

Even then, the power of dynamic programming, *i.e.,* considering only the optimal decisions obtained so far from the future by backward induction, is actually not fully exploited in dynamic decision modeling. Although dynamic influence diagrams correspond to exact formulations of the dynamic programming equation [Tatman and Shachter, 1990], all the tree-based solution methods that are based on forward induction do not make use of the optimal substructure. In these latter methods, all the possible decision consequences have to be simulated or traversed forward over time during evaluation. No reuse of optimal substructures is involved.

AI Planning

Solution methods in AI planning are based on *heuristic search* [Korf, 1987]. The solution algorithms are often not only concerned with finding the solution paths, but also how to cut down the search space in finding such paths. Guided by a set of heuristics, plan generations often involve action and state spaces at multiple levels of abstraction. The appropriate heuristics depend on the planning ontology of the framework.

3.3.3 Model Quality Metrics

To perform analysis on the model and the solution of a dynamic decision problem, we need a set of metrics to assess the quality of the model and the solution. The quality of the solution is usually determined, with respect to domain-specific knowledge, using standard statistical techniques such as one-way and two-way sensitivity analysis. This can be done on solutions generated by all the techniques addressed. The quality of the model can be measured in terms of its accuracy, conciseness, and clarity. A model is accurate if both its structure and parameters reflect the decision situation to a “satisfactory” extent. A model is concise if it contains only the relevant information pertaining to the decision situation. A model is clear if it contains information that can be easily accessed by an inference process to produce “meaningful” answers. Therefore, the quality measures are not only theoretically meaningful, but also indicate how easily the model can be “debugged” during problem formulation and problem analysis in practice.

In this section, we briefly compare the model quality metrics for the current techniques. A detailed explanation of the metrics is beyond the scope of this analysis; references for these metrics, however, are provided. We focus on highlighting the availability or absence of, and the similarities and differences in the metrics for the different techniques.

Semi-Markov Decision Processes

Accuracy

The fidelity of the probabilistic parameters involved in an SMDP can be measured with standard statistical techniques. The accuracy of the SMDP can be assessed in terms of the mathematical properties that can be proved about it, *e.g.,* existence and convergence of optimal policies. Some methods that facilitate such proofs are the certainty equivalence principle, the controllability and observability conditions, and the monotonicity and contraction mapping principle [Bertsekas, 1987]. The certainty equivalence principle asserts that an optimal policy is unaffected by the presence of zero-mean disturbances. By analyzing the ranks of the system matrices, controllability indicates if a dynamical system can be controlled; observability determines if its variables can be inferred. If the value functions manifest monotonicity or contraction mapping properties, convergence of the corresponding solution method, and hence the existence of optimal policies, is guaranteed.

Due to its precise mathematical definitions, however, many assumptions and transformations of the decision factors and constraints must be made to fit into the SMDP ontology. Therefore, while an SMDP supports formal analysis, such techniques may be difficult to apply and to understand.

Conciseness

The main difficulty in formulating a SMDP is the combinatorially increasing dimension of the state space with each relevant state attribute variable. One way to ensure conciseness of the state space description is to explicitly manipulate the combinations of state attributes with either formal *canonical forms* or domain-specific heuristics. Another way is to develop quantities called *sufficient statistics*, which ideally would be of smaller dimension than the original state space, and yet summarize all the essential information for problem solution and analysis [Bertsekas, 1987]. We do not know of any effective guidelines for developing such techniques in general.

Clarity

The clarity of an SMDP can be assessed in terms of its inferential efficiency and information explicitness. No formal metrics exist for this quality. In general, applicable inferences can be easily determined in a complete model with precise mathematical properties. Informational explicitness, however, cannot be easily verified due to the assumptions and transformations involved in problem formulation.

Dynamic Decision Modeling

Accuracy

Various metrics exist for assessing the accuracy of a dynamic decision model. Examples of these metrics include the equivalence decision class analysis for determining relevant chance events [Provan and Poole, 1991], the value of infor-

mation, and the "confidence" measure of the recommended decision, which compares the fidelity of subjective probabilities with objective relative frequencies in the model [Willard and Critchfield, 1986] [Neapolitan, 1993]. While most of these metrics are applicable to any dynamic decision model type, actual implementations of the metrics, and hence their ease of use, vary with different model types. This is particularly true for structural accuracy metrics. For instance, equivalence decision class analysis is usually employed in updating dynamic influence diagrams; the structural complexity of the tree-based decision models renders such updating much more difficult. In addition, some metrics are designed with specific model types in mind, *e.g.,* structural controllability and observability in influence diagrams [Chan and Shachter, 1992].

Conciseness

The structural differences also complicate the measure of conciseness among the dynamic decision models. The network structure of an influence diagram, for instance, is much simpler than a branching decision tree; by allowing asymmetric outcomes, however, a decision tree actually depicts more specific information than an influence diagram.

Without an intuitive, uniform definition of conciseness, we usually resort to determining the *adequacy* of a dynamic decision model. The information in a dynamic decision model is adequate when all elements of the decision problem are clearly defined. This usually involves refining the conceptualizations of parameters associated with the situation to a sufficient level so that a decision can be made [Clemen, 1991]. The *clarity test* is a technique for measuring such adequacy [Howard, 1988].

Clarity

The clarity of a dynamic decision model can also be assessed with respect to its inferential efficiency and informational explicitness. Again, the structural differences of the decision model types render these capabilities difficult to be characterized.

Although a dynamic influence diagram displays all the chance variables and their conditional dependencies in a simple network structure, the exact relations among the outcomes of the variables, which may be asymmetric, are encoded in the associated conditional distributions. Careful examination of the distributions are needed to derive the specific relations; this may hinder or complicate model debugging in practice.

On the other hand, a Markov cycle tree or a stochastic tree explicitly displays all chance variable outcomes. While the specific information is easily visualizable this way, the details may lead to a very complex tree structure. In practice, the resulting complexity is countered by adopting a few constant tree structures to model the consequences of all alternative actions, and using devices such as bindings to model the variations in information [Sonnenberg and Beck, 1993]. Unfor-

tunately, this violates the constant representation requirement for transparency as described in Section 3.1.1; it makes a lot of the information implicit, and hence renders model debugging difficult.

AI Planning

Accuracy

Several formal criteria for checking the correctness of a planning model have been proposed [Chapman, 1987] [Lifschitz, 1986] [Rosenschein, 1981]. These criteria are restricted to models with some basic operator representations. Currently, there are no general techniques for ensuring the accuracy of a planning model. AI planning languages, however, usually incorporate rich and variable representations for the concepts involved. For example, actions and goals represented in multiple levels of abstraction, capabilities for handling time constraints and resource requirements, *etc.* Such expressiveness may improve model accuracy by allowing more intuitive and more direct specification of the decision factors and constraints.

Conciseness

Given an expressive planning ontology, a planning model is usually concisely specified. For example, instead of enumerating all the states that an action or operator is applicable in, only the relevant state attributes are specified in the preconditions and the effects of the operator.

Clarity

There are no formal metrics for inferential efficiency and informational explicitness in AI planning. Since most of the decision factors and constraints are explicitly specified in a planning model, informational explicitness can be informally assessed. Due to the expressiveness of the planning ontology and the conciseness of the resulting model, however, more inferences may be necessary to derive useful information.

3.4 Summary

An analysis of the SMDPs, dynamic decision modeling, and AI planning approaches to dynamic decision making under uncertainty illuminates the following issues:

First, there is a common basis, in terms of both the task definition and the decision ontology, among the three techniques. All techniques support problem formulation, solution, and analysis. Their decision ontologies and problem structures for addressing the dynamic decision problems concerned in this work can also be interpreted in terms of the mathematical definitions of SMDPs. Each framework, however, has additional strengths and weaknesses relative to the others.

Second, there is a trade-off between model transparency and solution efficiency. A formal framework such as SMDP supports efficient and varied solution

methods, but problem formulation may require many assumptions and transformations of the information involved. On the other hand, a well-structured knowledge organization framework in AI planning facilitates problem formulation, but problem solution may involve complex inferences on such information.

Third, we have noticed that the graphical capabilities of dynamic decision models add an invaluable dimension to support problem formulation and analysis. Different dynamic decision models, however, graphically depict the relevant information in different perspectives, and each perspective has its strengths and limitations.

We draw from these results the following conclusions:

First, a sound and flexible framework for dynamic decision making under uncertainty could result from integrating the desirable features of existing techniques; such an integration should build on the common basis identified.

Second, the modeling task and the solution task are not only different in nature, but they also require different representational and inferential support. A vocabulary effective for supporting one task may not be adequate for the other. The weaknesses of most existing techniques arise from striving toward such a compromise.

Third, based on what we have learned in this analysis, we believe a new paradigm with multiple perspective reasoning and incremental language extension would supplant current approaches. Building on the basis of SMDPs, this new paradigm would allow different representational and inferential capabilities for problem formulation, solution, and analysis, and support visualization of the information involved in different perspectives. We introduce such a framework in the next chapter.

4 An Integrated Language Design

To address the issues concluded from the analysis in Chapter 3, we propose a new language, called DynaMoL (for Dynamic decision Modeling Language), for dynamic decision making under uncertainty. This language design is motivated by a set of specific desiderata for dynamic decision modeling and influenced by some general design principles in computer science. It distinguishes the representational and inferential support for formulating, solving, and analyzing dynamic decision problems.

4.1 Desiderata of An Integrated Language

An effective language for dynamic decision making under uncertainty should provide representational and inferential support for all the different tasks involved. In particular, it should balance the trade-off between model transparency and solution efficiency. Such a language should have an expressive decision ontology and a formal basis, it should support multiple levels of abstraction and multiple perspectives of visualization, and it should be extensible, adaptable, and practical.

4.1.1 Expressive Decision Ontology

The decision ontology of the new language should include decision factors and constraints common to most current frameworks; it should address, therefore, a wide range of issues in typical dynamic decision problems.

The vocabulary should facilitate problem formulation. In other words, expressiveness, succinctness, and transparency are its most important qualities. These qualities are determined not only by the types and varieties of the ontological components, but also by the representation and organization of the components.

4.1.2 Formal Theoretic Basis

The language should have a mathematical basis to support problem solution and problem analysis. This theoretical framework should reflect the expressiveness of the decision ontology, but it should have simple and rigorous syntax and semantics. These qualities would facilitate examining and analyzing the formal properties of the problem and its solutions.

4.1.3 Multiple Levels of Abstraction

During problem formulation, a user should be able to deal mainly with the relevant ontological concepts, instead of the specific mathematical definitions. This indicates the need for a decision ontology that is flexible, “high level”, and intuitive. The language, therefore, must support reasoning at multiple levels of abstraction.

Two general types of abstraction are involved. The first type refers to the ontological abstraction within the high level decision ontology. Some examples are action classes, and state and transition descriptions at different levels of details. The second type refers to the abstraction of the high level ontology from its formal representation. To ensure coherent integration of a high level decision ontology and an underlying theoretical framework, proper translations should be established for each type of abstraction.

4.1.4 Multiple Perspectives of Visualization

The graphical capabilities of dynamic decision modeling should be preserved in the new language. These capabilities facilitate modeling and analysis; they should be extended to visualize the decision factors and constraints involved in multiple perspectives, and across different levels of abstraction.

Visualization in multiple perspectives reflects a common pattern in human decision making. For instance, at one stage of problem formulation, it might be essential to consider only the possible state transitions; at another stage, it might be illuminating to estimate the uncertain effects of an action between state transitions. Again correspondences must be established among the graphical entities and relations in different perspectives.

4.1.5 Extensibility

The language should be incrementally extensible. Extensions involve incorporating additional language constructs for new types of decision factors and constraints. Both the high level decision ontology and the theoretical framework should be extensible. Modularity should be preserved through the translation across multiple levels of abstraction and multiple perspectives of visualization.

Given an expressive and stable theoretical framework, most of the extensions should be on the high level decision ontology. When additions are made to the theoretical framework, the mathematical properties should carry over, or generalize in a straightforward manner.

4.1.6 Adaptability

The language should be systematically adaptable. Adaptation involves changes in the organization of the language constructs; it does not necessarily affect the expressiveness of the language. For instance, instead of incorporating asymmetric relations in terms of the probabilities in a conditional distribution table, explicit statements can be made about the relations, thereby making them more obvious. Again both the high level decision ontology and the theoretical framework should be adaptable. Modularity should also be preserved through the translation across multiple levels of abstraction and multiple perspectives of visualization.

4.1.7 Practicality

The language should support implementations that can be put into practical use. A practical dynamic decision modeling language is modular, comprehensive, and explicit. The components of a modular language can be implemented separately and then integrated directly. Implementation is thus easier and less prone to errors. The number of the implemented language features can also be incrementally increased. A comprehensive language is applicable for real problems. The information expressed in terms of an explicit language can be easily accessible and examined. This would facilitate model debugging and support tools development.

4.2 Overview of Language Design

The DynaMoL design attempts to satisfy the above desiderata by integrating the expressiveness and transparency of AI planning languages, the graphical capabilities of dynamic decision models, and the concise properties and varied solutions of SMDPs. It also incorporates the general idea of programming language design: translation of a higher level language for modeling into an object language for execution.

Based on a basic decision ontology, the language has four major components: a *dynamic decision grammar,* a *graphical presentation convention,* a *formal mathematical representation,* and a *translation convention*. The decision grammar supports problem formulation with multiple interfaces. The presentation convention, in the tradition of graphical decision models, governs parameter visualization in multiple perspectives. The mathematical representation provides a concise formulation of the decision problem; it admits various solution methods, depending on the different properties of the formal model. The translation convention guides

development of correspondence rules among the different perspectives and representations of the decision factors and constraints. This in turn supports modular extensions to the scope of admissible decision problems and systematic improvements to the efficiency of their solutions.

A dynamic decision model formulated in DynaMoL has the following parts:

- the *time-horizon*, denoting the time frame for the decision problem;
- a set of *value functions*, denoting the evaluation criteria of the decision problem;
- a set of *states*, denoting the possible conditions that would affect the value functions;
- a set of *actions*, denoting the alternative choices at each state;
- a set of *events*, denoting occurrences in the environment that might affect the evolution of the states;
- a set of *transition parameters*, denoting the probabilistic and temporal characteristics, conditional on the actions, among the states;
- a set of *influence parameters*, denoting the probabilistic and temporal characteristics, possibly conditional on the actions, among the events and the states;
- a set of *declaratory constraints*, such as the valid conditions of and logical relationships among the states, events, transition parameters, and influence parameters; and
- a set of *strategic constraints*, such as the valid durations for certain actions, the number of times an action can be applied, the valid ordering of a set of actions, and the required duration between successive actions.

The initial DynaMoL design consists of a basic set of model components. In particular, the constraint definitions are minimal. The ontology is extended by incrementally incorporating individual translators or correspondence rules.

4.3 DynaMoL: The Language

The syntax of the language is defined by the dynamic decision grammar and the graphical presentation convention. The semantics is defined by the mathematical representation of an SMDP and the translation that bridges the grammar, the presentation, and the mathematical representation of the decision factors and constraints. This section summarizes the decision ontology and the features of the language components; Chapters 6 and 7 discuss the definitions in more details.

4.3.1 Basic Decision Ontology

We adopt a knowledge formalization approach similar to first-order predicate calculus (FOPC), as presented in [Genesereth and Nilsson, 1987]. In DynaMoL, a *conceptualization* of a decision situation includes the relevant decision parameters and the interrelationships among them. Formally, a conceptualization consists of a *universe of discourse*, a *functional basis set*, and a *relational basis set*. The universe of discourse is the set of decision parameters about which knowledge is being expressed. Various functions and relations can be defined among these parameters. The functional basis set is the set of relevant functions in the universe of discourse. The relational basis set is the set of relevant relations in the universe of discourse. In the example problem introduced in Section 2.2, the universe of discourse includes the actions PTCA, CABG, and MedRx, and their consequences in terms of status, MI, and restenosis. The functional basis set includes the probability functions and value functions defined. The relational basis set includes the probabilistic and temporal dependences.

Based on the conceptualization, we define the expressions in DynaMoL in terms of the following components: *basic concepts*, *variable concepts*, *functions, relations*, and *constraints*. These concepts are analogous to the constant symbols, variable symbols, function symbols, relation symbols, and sentences or well-formed formulas in FOPC. Many of the DynaMoL concepts, however, also have probabilistic interpretations. All of the variable concepts, for instance, also correspond to the random variables in probability theory. Moreover, both deterministic and probabilistic relations are involved.

Basic Concepts

A *basic concept* is a description of the world, within the universe of discourse, at a single point in time. There are three types of basic concepts: *events*, *actions*, and *states*.

Event

An *event* is a partial description of the world at a single point in time. It corresponds to a proposition or atomic sentence in FOPC. In particular, it describes an occurrence or a lack of occurrence of a phenomenon. Both "presence of MI" and "absence of MI" are events. In a clinical decision situation, an event represents either an occurrence or a lack of a physical or physiological condition of one or more patients, *e.g.,* "restenosis occurred" and "restenosis did not occur."

Action

An *action* is a special type of events which involves an actor. The actor can be the decision maker concerned, in which the action is *controlled*; the actor can also be unknown and the action is subject to chance, in which case the action is *embedded*. In a clinical decision situation, relevant actions include different tests and treatments, and their combinations, *e.g.,* "revascularization performed" and "no

revascularization performed." The effects of an action are expressed in terms of events. A complication of the action PTCA is the event "presence of MI."

State

A *state* is a complete description of the world at a single point in time. The state description comprises a set of events or proposition as defined above. For instance, the state well in the example problem is defined in terms of the events: {***status*** = *alive,* ***MI*** = *absence,* ***restenosis*** = *absence*}. Every state has an associated *value*, indicating the desirability of being in that state.

Variable Concepts

A *variable concept* corresponds to a random variable in probability theory. It represents the uncertainty about the world, as described by the basic concepts, at a single time point. We only consider discrete variables in this work.

a) Chance Variable

A *chance variable* corresponds to a chance node in a decision model. It has several possible outcomes or values; the outcomes can be events or states. Each possible outcome occurs only by chance and cannot be explicitly chosen. There are three types of chance variables: event variables, state attribute variables, and state variables.

Event Variable:

The outcomes of an event variable are events. An event variable usually represents a possible consequence of an action. In particular, it can capture an observable or unobservable phenomenon that directly or indirectly describe the states.

State Attribute Variable:

A state attribute variable is a special kind of event variables; it represents a characteristic or property directly relevant to describing the states. The outcomes of a state attribute variable are events. Given a set of such variables, a state is usually defined in terms of the Cartesian product of their outcomes. For instance, in the example problem, all the event variables involved: ***status***, ***MI***, and ***restenosis***, are state attribute variables. The state attribute variable ***status*** has outcomes *alive* and *dead*, ***MI*** has outcomes *presence of MI* and *absence of MI*, and ***restenosis*** has outcomes *presence of restenosis* and *absence of restenosis*.

State Variable

A *state variable* represents the uncertainty about the actual state that the world is in at a single point in time. It corresponds to a *state space* in an SMDP and a state variable node in a dynamic influence diagram at that time point or decision stage. All of the outcomes of a state variable are states.

b) Action Variable

An *action variable* denotes a decision or choice point at a single point in time; it corresponds to part or all of the *action space* in an SMDP or a decision node in a dynamic influence diagram at that time point or decision stage. An action variable has several possible *alternatives*, which are actions. Any alternative action can be explicitly controlled or chosen as the decision.

c) Function Variable

A *function variable* denotes the value function or desirability measure structure at a single time point. It corresponds to a value node in a dynamic influence diagram at that time point or decision stage. Each outcome of the function variable is a number, a table, or a function representing the value or desirability of a state given an action. In other words, the outcomes of such a variable are determined by the Cartesian product or other functional combination of the outcomes of a state variable and those of an action variable.

Relations

The basic *relations* among the concepts are probabilistic dependences and temporal dependences.

Probabilistic Dependences

A *probabilistic dependence* relation between two concepts corresponds to the conditional dependence notion in probability theory. Such a relation indicates a concept is conditionally dependent on another. The direction of a probabilistic dependence relation reflects the definition of the underlying conditional dependence; it has no temporal implication. The special class of probabilistic dependence among the states are called *transitions*.

Temporal Dependences

A *temporal dependence* relation between two concepts indicates that one temporally precedes another; it has no causal implication.

Functions

The basic *functions* defined over the concepts and the relations are probability functions and value functions.

Probability Function

A probability function is either a PMF or a CDF.

Value Function

A *value function* measures the desirability of a state. Such a measure may have different dimensions, e.g., monetary cost and life expectancy, and are usually conditional on an action.

Constraints

The *constraints* are meta-level descriptive or prescriptive conditions imposed on the concepts, the relations, and the functions as defined above. They correspond to the logical sentences and quantification sentences in FOPC. Some examples are the applicability of actions with respect to the states or time or both, and the validity of states and events with respect to each other or time or both. For instance, given a state space $\boldsymbol{S}$, the following constraint specifies that the action PTCA is applicable in all states:

$$\forall s \in \boldsymbol{S}, Applicable\,(\text{PTCA}, s)$$

The relation *Applicable*, of course, will be properly interpreted by the model constructor and solution algorithm to impose the constraint.

4.3.2 Dynamic Decision Grammar

The dynamic decision grammar for DynaMoL is an *abstract grammar*. Following the convention in [Meyer, 1990], this grammar contains the following components:

- A finite set of names of *constructs*;
- A finite set of *productions*, each associated with a construct.

  An example of a production that defines the construct "model" is as follows:

$$\begin{aligned} \textit{Model} \quad \rightarrow \quad & \textit{name: Identifier;} \\ & \textit{contexts: Context-list;} \\ & \textit{definitions: Definition-list;} \\ & \textit{constraints: Constraint-list;} \\ & \textit{solution: Optimality-policy} \end{aligned}$$

Each construct describes the structure of a set of objects, called the *specimens* of the construct. The construct is the (syntactic) *type* of its specimens. In the above example, an object with the type Model has five parts: name (with type Identifier), contexts (with type Context-list),... *etc.* The constructs/types appearing on the right-hand side of the above definition are similarly defined by different productions. A set of primitive constructs/types are assumed, *e.g,* String, Cumulative Distribution Function, *etc*.

There are some fixed ways in which the structure of a construct can be specified. The above example shows an "aggregate" production, *i.e.,* the construct has specimens comprising a fixed number of components. Other types of productions include:

"Choice" productions, *e.g.*, the time duration in the decision horizon can be finite or infinite:

$$\textit{Time-duration} \quad \rightarrow \quad \Im^{+} \cup \{0\} \mid \infty$$

"List" productions, *e.g.*, the state-space of the decision problem consists of one or more states:

$$\textit{State-space} \quad \rightarrow \quad \textit{State}+$$

The DynaMoL grammar defines the structure of a dynamic decision model in terms of its components; the structures of these components are recursively defined in a similar manner. The grammar specifies the information required to build a model. On the other hand, there can be many ways to manipulate such information. In other words, the abstract grammar can support different interface implementations. For example, an object of type State-space may be specified in 3 different ways:

- Text command interface: Type "state-space state-1 state-2 state-3" to command-prompt.
- Graphical interface: Draw three state-nodes labeled "state-1," "state-2," "state-3" in the display window.
- Routine or code interface: Type "(define-state-space 'state-1 'state-2 'state-3)" to the Common Lisp prompt.

The complete grammar for the current version of DynaMoL is given in Appendix A.

4.3.3 Graphical Presentation Convention

The graphical presentation convention in DynaMoL prescribes how the decision factors and constraints expressible in the grammar are displayed; it determines how the same information can be visualized in multiple perspectives. Two general *perspectives* or *views* are defined: *transition view* and *influence view.* The current presentation definitions adhere to the established graphical representations such as Markov state transition diagrams and dynamic influence diagrams. These definitions, however, can be changed easily as long as the graphical components consistently correspond to the dynamic decision grammar constructs. Additional perspectives such as partial decision tree view, information flow view, and two-dimensional Cartesian plots, can also be incorporated when appropriate.

Transition View

The transition view corresponds directly to the Markov state transition diagram. Given an action, the transition view depicts the possible state transitions, as shown in Figure 4.1.

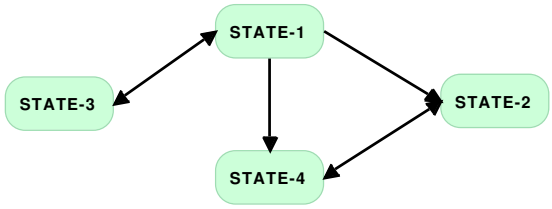


Figure 4.1 Transition view for an action. The nodes denote the states; the links denote the possible transitions from one state to another.

Influence View

Given an action, the influence view depicts the possible event variables that affect the transitions from one state to another, as shown in Figure 4.2.

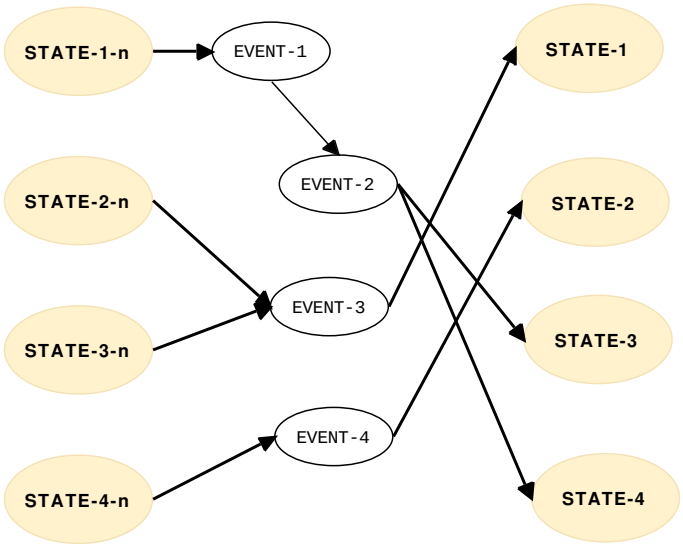


Figure 4.2 Influence view for an action. The index "n" indicates the relevant decision stage. The shaded nodes depict the states, the clear nodes the possible event variables that affect the state transitions, and the links the probabilistic dependences.

4.3.4 Mathematical Representation

The mathematical representation of a dynamic decision model formulated in DynaMoL is an SMDP as described in Section 2.3. All the solution methods for SMDPs can be used to derive an optimal policy for the dynamic decision model formulated in DynaMoL. The mathematical representation will also support formal analysis on the model and its solutions.

4.3.5 Translation Convention

Two general types of translation are involved: *inter-level translation* and *inter-perspective translation*. In inter-level translation, a dynamic decision model specified in the dynamic decision grammar in DynaMoL is automatically translated into an SMDP. The model specification may involve more constructs than those defining an SMDP. A set of translators or correspondence rules are employed to bridge the constructs of the dynamic decision grammar and the graphical presentation convention with the definitions of the SMDP. The general idea, as illustrated in Figure 4.3, is to map a construct or a set of constructs in the grammar to an entity or relation in the mathematical representation.

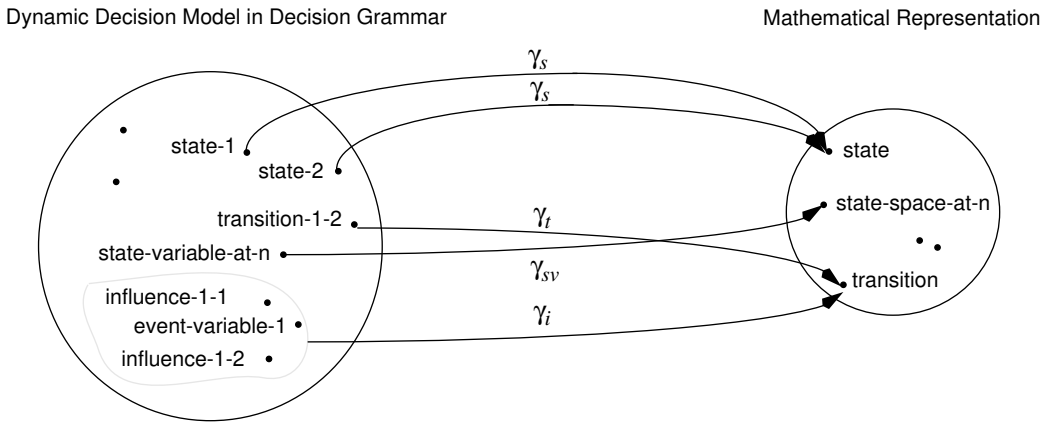


Figure 4.3 Inter-level translation by mapping. Each arrow type indicates a transformation function or correspondence rule.

Problem formulation and analysis in DynaMoL involve specifying and examining the decision factors and constraints in multiple perspectives. The different perspectives may facilitate expressing the same knowledge in different ways. For instance, a state transition can be specified either directly in terms of the PMF or CDF of a one-step transition function, or indirectly as several constituent event variables and the associated conditional probability distributions. Inter-perspective translation, therefore, establishes proper correspondence among the different representation or organization formats. Since the same information is involved, inter-level translation can then be defined in terms of any covering set of organization formats among the different perspectives. Figure 4.4 illustrates the idea of inter-perspective translation.

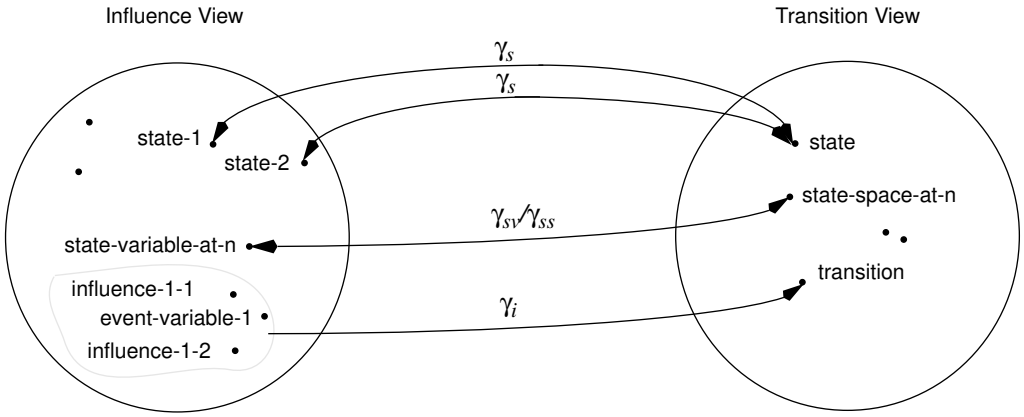


Figure 4.4 Inter-perspective translation by mapping. Each arrow type indicates a transformation function or correspondence rule. Directions of the arrows indicate directions of translation.

4.4 DYNAMO: A Prototype Implementation

Figure 4.5 shows the system architecture of a prototype implementation of DynaMoL. The system, called DYNAMO[1], is implemented in Lucid Common Lisp on a Sun SparcStation, with the GARNET graphics package [Myers et al., 1990]. It includes a graphical user interface that allows interactive model specification. The specification can be strictly text-based, or aided by the available graphical presentations of the decision factors and constraints. Figure 4.6 shows the interface of the current version of DYNAMO. Only a subset of the dynamic decision grammar is included in this implementation; the solution method supported is value iteration.

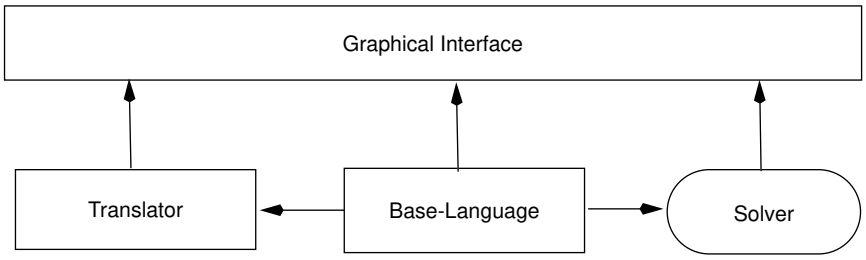


Figure 4.5 The system architecture of DYNAMO: a prototype implementation of DynaMoL. The blocks indicate system components; the arrows indicate information inflows.

1. This system has no association with the DYNAMO simulation language [Pugh, 1970].

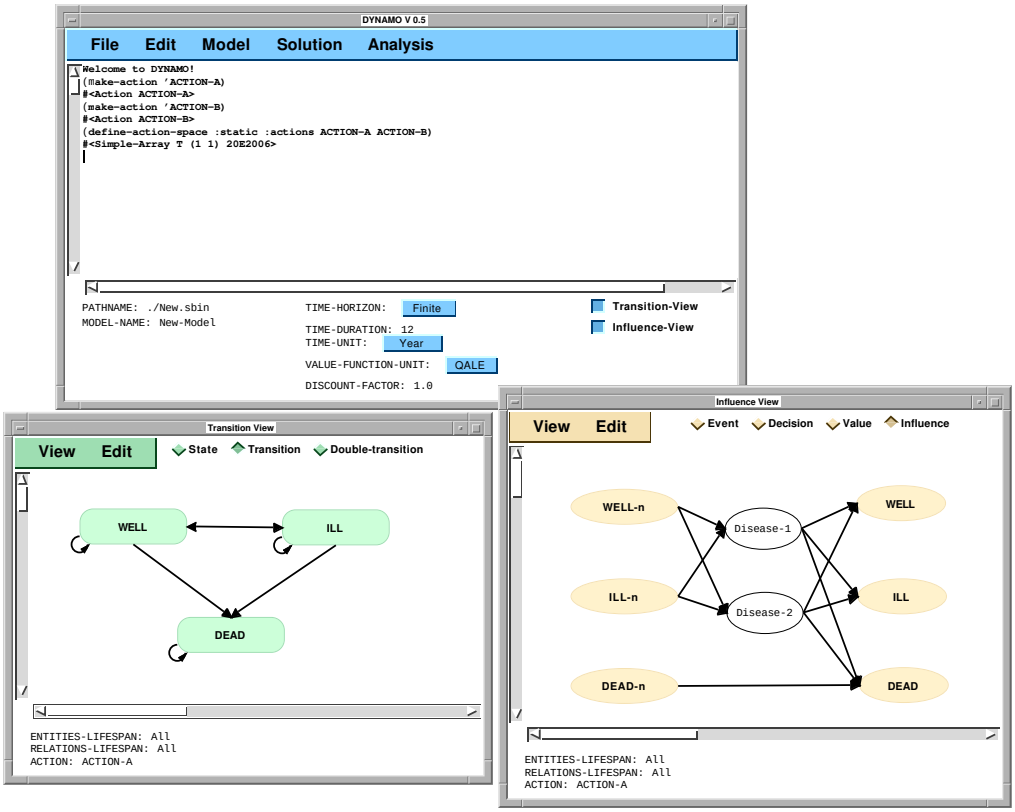


Figure 4.6 The DYNAMO interface.

5 A Case Study

To facilitate discussion and evaluation of decision making in DynaMoL, we have conducted a comprehensive case study based on an actual decision situation in medicine[1]. This chapter introduces the background, the problems and the assumptions involved. Details of the case study are discussed in Chapters 6 and 7. A complete description of the model data and solution results is given in Appendix C.

5.1 Management of Paroxysmal Atrial Fibrillation

Atrial fibrillation (AF) is a kind of cardiac arrhythmia or abnormal heartbeat. It means irregular and rapid randomized contractions of the atria working independently of the ventricles. It has two undesirable side-effects: 1) it causes hemodynamic deterioration at very slow or very rapid heart rates and a loss of atrial mechanical contraction; and 2) blood clots can collect in the fibrillating atrium that may subsequently embolize, *i.e.,* become material masses present in the blood stream [Fishman et al., 1985]. Embolization is dangerous because it blocks blood flow to the target organs and may lead to strokes, renal failure, gangrene of the legs, *etc*.

AF can occur in paroxysms, *i.e.,* sudden, periodic episodes. Both the frequency and the length of the AF episodes usually increase with time; constant fibrillation often develops eventually.

1. This case study is based on a consult case for the Division of Clinical Decision Making at the Tufts-New England Medical Center. Dr. Charles A. Ellis is in-charge of this consult; he has kindly provided the background medical knowledge and shared the dynamic decision modeling expertise involved in the case study. The original consult was done in a Markov cycle tree framework.

Management of AF involves antiarrhythmic agents to control the heart rates and to restore normal sinus rhythm. Because of the risk of embolization, anticoagulants such as warfarin and aspirin are often used to prevent blood clot formation in the atria. The treatments, however, have undesirable side-effects of their own. In particular, a commonly used antiarrhythmic agent, Quinidine, increases the risk of sudden death; an anticoagulant may cause excessive bleeding, which in turn may lead to strokes or death.

5.2 Case Description and Clinical Questions

The patient is a 52 year old white male with a history of paroxysmal AF. He has been on warfarin, an anticoagulant, for about 10 years, and on Quinidine, an antiarrhythmic agent, for a shorter period. His arrhythmic attacks are asymptomatic, and of unknown frequency and duration. Since mid-1990, several episodes of AF have been detected, and the level of quinidine prescribed has been steadily raised accordingly. Each prescription adjustment has been able to bring him back to normal sinus rhythm (NSR), *i.e.,* out of arrhythmia. The patient is recently diagnosed as being diabetic and put on diabetic medication. He also has mild hypertension and a slightly diminished ejection fraction. All these three factors are associated with increased risk of embolization.

The clinical questions to be addressed is as follows:

Problem 1: Quinidine decreases the proportion of time that the patient spends in AF. *Does this decrease his risk of embolic complications enough to justify the increased risk of sudden death?*

Problem 2: What is the optimal course of treatment in the next five years, taking into account the varying relative risk of bleeding for warfarin with respect to the duration of treatment?

5.3 Assumptions

AF is the only relevant condition targeted in the dynamic decision problems. The patient's diabetes, hypertension, and diminished ejection fraction are not affected by the AF therapy; these conditions, however, markedly influence the likelihoods of various therapeutic consequences.

The actions involved are Quinidine and warfarin. Quinidine is strictly an externally controlled action. Warfarin, on the other hand, can be initiated or prohibited in accordance with certain observable events. The effects of the actions may recur and also vary in duration.

Effects of Quinidine

Quinidine decreases the likelihood of AF. Its most undesirable side-effect is to increase the risk of sudden cardiac death.

Effects of Warfarin

Warfarin decreases the risk of thromboembolism or embolization of blood clots. A thromboembolic event may be transient or result in a stroke. The undesirable side-effects of warfarin include cerebral hemorrhage, *i.e.,* bleeding in the brain, and gastrointestinal bleeding, *i.e.,* bleeding in the stomach and the intestines. All these consequences may be fatal.

The relative risk ratio compares the rate at which an event occurs given and not given an action [Wong et al., 1990]. The relative risk of bleeding for warfarin, with respect to the duration of treatment, is assumed to be constant in the first problem, and varying in the second problem.

Initiation and Prohibition of Warfarin

Occurrence of a thromboembolic event will mandate initiation of warfarin if the patient is in AF, and if warfarin is not prohibited. Once warfarin is initiated, it will not be stopped unless there are contra-indications.

Occurrence of a bleeding event, either cerebral or gastrointestinal, will mandate discontinuation of warfarin. This will also prohibit warfarin from being prescribed in the future.

5.4 Assessment Objectives

The case study aims to illuminate the theoretical and practical capabilities and limitations of the DynaMoL framework. To show that DynaMoL can handle an actual dynamic decision problem in medicine, we adhere closely to the original clinical consult in addressing the first clinical question. To show that DynaMoL offers additional flexibility to existing frameworks, we focus on illustrating the ontological and computational features involved in addressing the second clinical question; the clinical significance of the data and assumptions adopted in this latter problem are less important.

We informally assess the effectiveness of modeling and solving the two dynamic decision problems with respect to three criteria. First, we demonstrate how the relevant decision factors and constraints can be expressed in the DynaMoL ontology. Second, we compare the solutions to the results of the actual clinical consult or the clinical judgment of domain experts or both. Third, we examine sensitivity analysis results on the solutions to ensure reasonable behavior of the parameters involved.

For ease of exposition, we formulate the two dynamic decision problems in terms of a single dynamic decision model with two sets of slightly different numerical parameters. Chapter 6 details how we construct the dynamic decision model and translate it into its mathematical representation. Chapter 7 describes the solution methods involved and the sensitivity analyses supported.

6 Dynamic Decision Model Formulation

As mentioned in Chapter 3, dynamic decision making under uncertainty is an iterative process; it involves interleaving and repeating steps of problem formulation, solution, and analysis. Problem formulation is mainly a modeling process; problem solution is a computational process. Analyses on the model and its solutions bridge and improve the modeling and the computational results.

In this chapter, we examine problem formulation in DynaMoL. With the aid of the DYNAMO system interface, we describe in detail the language syntax and semantics in terms of the modeling tasks they support. To highlight the language features, we also explain how such tasks are handled in other dynamic decision modeling frameworks when appropriate.

In the following discussion, details of the case study introduced in Chapter 5 serve mainly as examples to illustrate the different modeling tasks. The choice of the modeling assumptions are less essential.

6.1 Dynamic Decision Modeling in DynaMoL

Problem formulation in DynaMoL involves constructing a dynamic decision model and translating it into its mathematical representation. The model captures the relevant decision factors and constraints for solving a dynamic decision problem. In the programming language analogy, a dynamic decision model is a program written in DynaMoL that can be compiled or interpreted into its mathematical representation. Such a program can also be viewed as an *object* representing the dynamic decision problem in the object-oriented programming paradigm. Computational *methods* are then applied to the model and its mathematical representation to produce solutions for and answer queries about the problem.

A dynamic decision model in DynaMoL is a well-formed or complete model when it corresponds to a well-formed SMDP with an *optimality criterion*. Some model constructs directly correspond to the underlying mathematical definitions, others need to be translated. The default optimality criterion is finite horizon discounted total expected value.

Definition 1 (Dynamic Decision Model) *A dynamic decision model M in* DynaMoL *is an 8-tuple* $\langle \boldsymbol{T}, \boldsymbol{A}, \boldsymbol{S}, \boldsymbol{E}, \Omega_{\boldsymbol{E}}, \Xi, \Psi, \varsigma, \mathrm{K}, \Gamma \rangle$ *together with an optimality criterion, where:*

$\boldsymbol{T}$ *is the time-index set or decision horizon;*

$\boldsymbol{A}$ *is the action space;*

$\boldsymbol{S}$ *is the state space;*

$\boldsymbol{E}$ *is the event-variable space;*

$\Omega_{\boldsymbol{E}}$ *is the set of events or outcomes of the event variables;*

Ξ *is the set of transitions;*

Ψ *is the set of influences;*

ς *is the set of value functions;*

K *is the set of constraints; and*

G is the set of translators.

6.2 Model Construction

Dynamic decision model construction in DynaMoL typically comprises the following tasks:

1. Specify a dynamic decision problem type, its duration, and its optimality and evaluation criteria.
2. Define the alternative actions and the states involved.
3. Conditional on each action, specify the possible transitions among the states and the values achievable in the states.
4. When such transitions and values are unobservable or difficult to be directly assessed, specify the possible event variables and their relations that constitute the transitions.
5. Impose relevant constraints among the decision factors when appropriate.

Besides a natural way to begin model construction, there is no strict order on the tasks involved. The specifications and definitions may and often change in the modeling process. A major challenge for the modeling language, therefore, is to provide adequate and direct support for the changes. In DynaMoL, such support is provided by the dynamic decision grammar, the graphical presentation convention, and their respective interfaces.

6.2.1 Basic Problem Characterization

The basic characteristics of a dynamic decision problem are determined by the *problem type*, the *decision horizon*, and the *evaluation criterion*. The problem type indicates the nature of the decisions involved. The decision horizon determines the time frame in which the decisions are relevant. The evaluation criterion specifies the objective or goal of the decisions; multiple criteria may sometimes be involved.

Problem Type

Two types of dynamic decision problems are supported by DynaMoL: *optimization* problems and *discrimination* problems with respect to some objective or evaluation criterion. An optimization problem is solved by constructing an optimal policy:

$$\pi^* = \{\mu_0^*, \mu_1^*, \mu_2^*, \mu_3^*, \ldots\}$$

where $\mu_t^* : \boldsymbol{S} \rightarrow \boldsymbol{A}$; $t \in \boldsymbol{T}$ is an optimizing function from the state space $\boldsymbol{S}$ to the action space $\boldsymbol{A}$ at time t with respect to the time index set $\boldsymbol{T}$.

An optimization problem compares the independent effects of actions; the problem solution answers the question: At *each* decision stage, what is the best alternative action to take?

Contrarily, a discrimination problem is solved by choosing the best policy:

$$\pi^* = \pi^{a*}$$

from a predetermined set of single-action stationary policies $\{\pi^a\}$, *i.e.,* policies which involve a single action over the entire decision horizon, such that

$$\pi^{a*} = \{\mu^{a*}, \mu^{a*}, \mu^{a*}, \mu^{a*}, \ldots\}$$

where $\mu^{a*} : \boldsymbol{S} \rightarrow a*$; $a* \in \boldsymbol{A}$, is a function from the state space $\boldsymbol{S}$ to an optimizing action $a*$.

There are two types of discrimination problems. The first type also compares the independent effects of actions; the problem solution answers the question: For *all* decision stages, what is the best alternative action to take? The second type delineates *strategies* or combinations of actions with dependent effects. It allows all the possible consequences of some actions to be considered, instead of only the effects of the optimal action. This formulation designates a single action as externally *controlled* in a strategy, and incorporates the application pattern of all the other actions, called the *embedded* actions, in the state descriptions. To solve the problem, an explicit enumeration of the policies associating each controlled action and all possible combinations of the embedded actions is necessary. The problem solution answers the question: For *all* decision stages, taking into account the

effects of the controlled and embedded actions, what is the best alternative strategy to take?

In clinical decision making, determining the best course of diagnostic tests and treatments are optimization problems, while comparing the efficacies of diagnostic tests and treatments are discrimination problems. The first dynamic decision problem in the case study is formulated as a discrimination problem.The problem is to determine the net effects of the antiarrhymic agent Quinidine on a patient who is on the anticoagulant warfarin; Quinidine is a controlled action, warfarin an embedded action. As mentioned in Section 5.3, warfarin is never stopped once it is administered, unless there are contra-indications for it. Similarly, warfarin is never administered again once it is stopped. Four strategies are to be discriminated:

1. Strategy "*None*": Start without any drug, with warfarin administered and stopped as necessary;
2. Strategy "*Warfarin*": Start with warfarin, which can be stopped when necessary;
3. Strategy "*Quinidine*": Start with Quinidine, with warfarin administered and stopped as necessary; and
4. Strategy "*Both*": Start with both Quinidine and warfarin, with warfarin stopped when necessary.

The second dynamic decision problem in the case study is formulated as an optimization problem. The problem is to determine a course of optimal initial action, *e.g.,* start with or without warfarin or Quinidine or both, with strategic implications as defined above.

Other dynamic decision modeling frameworks support formulation of optimization and discrimination problems to different extents. Dynamic influence diagrams support direct formulation of both optimization and discrimination problems. Without augmenting computational structures such as bindings or flags, and assumptions on the solution methods, both Markov cycle trees and stochastic trees only support direct formulation of discrimination problems. The tree-based techniques do not support direct formulation of optimization problems because they do not allow decisions to be made in stages; a decision is made only at the final stage by comparing the objective values for the individual decision alternatives over the entire decision horizon.

Decision Horizon

The decision horizon $\boldsymbol{T} = \{0, 1, 2, 3, \ldots\}$ can be finite or infinite. A finite horizon model with a long duration and small time units may be approximated as an infinite horizon model. We shall discuss more about the conditions under which such an approximation is feasible in Chapter 7.

Clinical decision problems usually involve finite decision horizons because the patient's life-expectancy is finite. Long-term decision horizons project the deci-

sions over an approximation of the patient's lifetime; short-term decision horizons consider the best course of action in the immediate future, *e.g.,* in the next five years. In the case study, we assume a long-term decision horizon of 50 years or 600 months for the first dynamic decision problem, and a short-term decision horizon of 5 years or 60 months for the second.

Evaluation Criterion

The evaluation criterion can be in any unit of interest, *e.g.,* life expectancy, monetary cost, and cost-benefit ratio; it is defined by one or more sets of value functions. More than one evaluation criterion may be involved. We may be interested to determine the quality-adjusted life expectancy of a patient, as well as the monetary cost incurred by his illness in the next five years. As in all other dynamic decision modeling frameworks, the value functions are assumed to be linear and additive. We will formally define the value functions in Section 6.2.7. The evaluation criterion for the case study problem is quality-adjusted life expectancy.

Figure 6.1 shows part of the DYNAMO interface that specifies some of the basic problem characteristics for the case study.

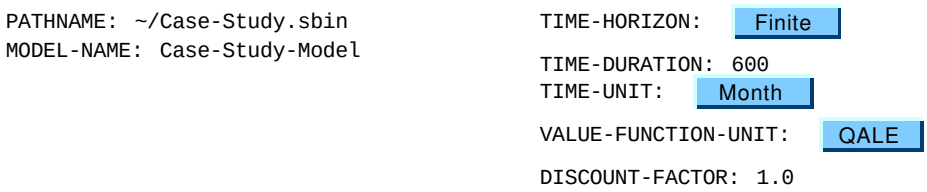


Figure 6.1 Basic problem characteristics of the case study (partial).

6.2.2 Action Space Definition

The *action space* $\boldsymbol{A} = \{a_1, a_2, \ldots, a_{|\boldsymbol{A}|}\}$ denotes the set of actions to be considered in the dynamic decision problem. DynaMoL currently assumes a *static* action space, *i.e.,* the action space remains unchanged over the entire decision horizon. We explain how this assumption can be relaxed in Chapter 8.

The action space of an optimization problem model consists of the individual actions. The action space of a discrimination problem model consists of the controlled actions of the relevant strategies; the most basic strategies involve only the controlled actions with no additional embedded actions. Note that the word action indicates a single unit of behavior. An action may actually involve more than one activity, *e.g.,* administer two drugs and perform two diagnostic tests; the constituent activities in an action are assumed to be simultaneous and constant at every decision stage.

In the case study, the action space consists of the two controlled actions for the strategies involved: $\boldsymbol{A} = \{\text{NoDrug}, \text{Quinidine}\}$. The effects of the embedded

action warfarin with or without Quinidine are expressed in the state descriptions, as described in the next subsection, and can be read off directly.

6.2.3 State Space Definition

The *state space* $\boldsymbol{S} = \{s_1, s_2, \ldots, s_{|\boldsymbol{S}|}\}$ denotes the set of states in which the controlled actions can take place. DynaMoL currently also assumes a *static* state space, *i.e.,* the state space remains unchanged over the entire decision horizon. We explain how this assumption can be relaxed in Chapter 8.

The states are defined in terms of a set of *state attribute variables*; these variables denote the different characteristics that constitute and distinguish the states. There are three types of state attribute variables. The first type affects the value functions, *e.g.,* quality-adjusted life expectancy of a patient depends on whether he has normal sinus rhythm or atrial fibrillation. The second type indicates the eligibility of one or more actions, *e.g.,* warfarin is applicable only in a state with no previous contra-indication for anticoagulation. The third type reflects the status of one or more embedded actions, *e.g.,* whether warfarin is administered in a state when Quinidine is considered.

The states are usually defined as the Cartesian product of the outcomes of the state attribute variables. This definition, however, leads to many redundant or invalid state descriptions in practice. For instance, the most common state attribute involved in clinical decision making is whether the patient is alive or dead; if the patient is dead, all other state attributes are irrelevant, and the state space size can be cut in half. There are many ways in which the state space can be reduced in size. Most of them, however, involve domain specific heuristics. In general, such heuristics can be expressed in terms of the declaratory constraints to be described in Section 6.2.8.

The relevant state attribute variables and their possible outcomes in the case study are shown in Table 6.1. ***Sinus rhythm*** indicates heartbeat pattern. ***Cerebral vascular accident*** indicates obstruction of blood flow to the brain, which may result in a stroke or temporary weakness. These two state attributes affect the value functions. ***Warfarin eligibility*** affects the applicability of the action warfarin. ***Warfarin status*** reflects the status of the embedded action warfarin.

The states are defined in terms of the Cartesian product of the outcomes of the above four state attribute variables. A complete listing of the states can be found in Appendix C. We employ a simple heuristic to reduce the size of the state space: Since warfarin will not be administered when it is not eligible, we eliminate that combination in the resulting states.

Table 6.1 State attribute variables and their outcomes for the case study

State Attribute Variable	Outcomes
Status	*Alive (Alive)* *Dead (Dead)*
Sinus rhythm	*Normal sinus rhythm (NSR)* *Atrial Fibrillation (AF)*
Warfarin eligibility	*Warfarin-OK (War-OK)* *Warfarin-Not-OK (War-No)*
Warfarin status	*Warfarin-On (War-On)* *Warfarin-Off (War-Off)*
Cerebral vascular accident	*None (None)* *Transient ischemic attack (TIA)* *Stroke (Stroke)*

In dynamic influence diagrams, the state attribute variables are represented as chance nodes directly influencing the value nodes; the implicit state descriptions, therefore, are strict Cartesian product of the outcomes of these chance nodes. In Markov cycle trees and stochastic trees, state attribute variables are not captured in the bindings; state space definition in these frameworks often involves logical combinations of the relevant bindings.

6.2.4 State Transition Specification

Conditional on each controlled action, a set of possible *transitions* $\Xi = \{\xi(a, i, j, t) \mid i, j \in \boldsymbol{S},\ a \in \boldsymbol{A},\ t \in \boldsymbol{T}\}$ is defined among the states.

Definition 2 (Transition) *A transition relation* $\xi \subseteq \boldsymbol{A} \times \boldsymbol{S} \times \boldsymbol{S} \times \boldsymbol{T}$ *between any two states* $i, j \in \boldsymbol{S}$*, given an action* $a \in \boldsymbol{A}$ *at time* $t \in \boldsymbol{T}$*, denotes the accessibility of j from i given a at t. The nature of this accessibility is characterized by a one-step transition function with* PMF $q_{ij}^{(a)}(.)$ *and* CDF $Q_{ij}^{(a)}(.)$ *as defined for an* SMDP *in Section 2.3.*

Recall that a transition function is a joint probability distribution governing the two dimensions of a transition: destination of the transition and time remaining or duration before the transition; temporal precedence is imposed in determining the destination. As described in Section 2.3, one way to calculate the transition functions is by multiplying the eventual transition probability $P_{ij}^{(a)}(.)$ and the holding-time PMF $h_{ij}^{(a)}(.)$ or CDF $H_{ij}^{(a)}(.)$. Another way to calculate the transition function is by multiplying the conditional transition probability $p_{ij}^{(a)}(.)$ and the waiting-time PMF $w_i^{(a)}(.)$ or CDF $W_i^{(a)}(.)$. States with no out transitions are called *absorbing states*.

The possible transitions may be estimated before the transition functions are exactly defined. For instance, there is a transition from every state to the dead state. Similarly, once a patient is ineligible for warfarin treatment, he should not be able to make a transition to a state where warfarin is on. In DynaMoL, the transition view supports graphical presentation of the Markov state transition diagram. Transition functions can be specified directly on the links representing the transitions.

A transition view of the states and the possible transitions for the action Quinidine is shown in Figure 6.2. For ease of exposition, it is an abstract or abbreviated view in that only two state attributes that affect the transition patterns: ***warfarin eligibility*** and ***warfarin status*** are included in the state descriptions. The actual state transition diagram has all the states with the displayed state attributes connected to each other in a similar manner.

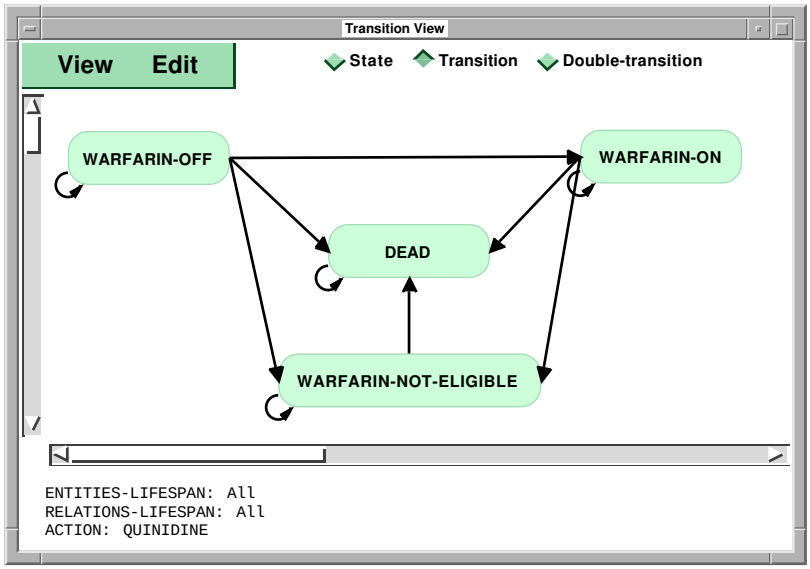


Figure 6.2 An abbreviated transition view for action Quinidine in case study.

All current dynamic decision modeling frameworks do not support visualization of the state transitions in the transition view manner.

6.2.5 Event Variable Space Definition

At times it is difficult to specify the transition functions directly, especially when the action effects are partially observable or unobservable. Such effects are modeled in DynaMoL as event variables and their associated relations among the states. For each controlled action $a \in \boldsymbol{A}$, its *action-specific event variable space* $\boldsymbol{E}_a = \{e_1, e_2, \ldots, e_{|\boldsymbol{E}_a|}\}$ denotes the set of event variables that constitute its effects. The *event variable space* $\boldsymbol{E} = \bigcup_{a \in \boldsymbol{A}} \boldsymbol{E}_a$ denotes the set of all event vari-

ables for all the actions. Each event variable e has an associated *event* or *outcome space* $\Omega_e = \{\omega_1, \omega_2, \ldots, \omega_k\}$; $k \geq 1$, which denotes the set of possible outcomes for e. The *action-specific event space* $\Omega_{\boldsymbol{E}_a}$ and the *overall event space* $\Omega_{\boldsymbol{E}}$ are similarly defined.

In DynaMoL, the action-specific event variable space can be different; the event variable space however, is currently assumed to be static over time. In other words, the event variables associated with different actions may be different, but the same set of event variables are associated with each action over all decision stages. In the case study, the action-specific event variable spaces for the controlled actions NoDrug and Quinidine are assumed to be the same; the relevant event variables indicate the presence or absence of ***thromboembolization***, ***cerebral hemorrhage***, ***gastro-intestinal bleeding***, whether the patient survives the events should they occur, and the set of state attribute variables mentioned earlier.

An event variable corresponds to a random variable in probability theory; it also corresponds to a chance node in existing dynamic decision model frameworks. A conditional probability distribution on the outcomes, which may be conditional on an action, the outcomes of other event variables, and states, is associated with each event variable. The distributions may be non-homogeneous or time-dependent. Specification of the distributions usually follows the specification of a set of probabilistic influences among the event variables, as described below. The state space $\boldsymbol{S}$ at each time point t or decision stage n can be represented as a special event variable, called the state variable $\boldsymbol{s}_t$ or $\boldsymbol{s}_n$. The outcomes of the state variable are all the states in the corresponding state space.

6.2.6 Probabilistic Influence Specification

Conditional on each controlled action, a set of *probabilistic influences* $\Psi = \{\psi(a, \boldsymbol{x}, \boldsymbol{y}, t) \mid a \in \boldsymbol{A},\ \boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{E}_a,\ t \in \boldsymbol{T}\}$ is defined among the event variables.

Definition 3 (Probabilistic Influence) *A probabilistic influence relation* $\psi \subseteq \boldsymbol{A} \times \boldsymbol{E}_a \times \boldsymbol{E}_a \times \boldsymbol{T}$ *between any two event variables* $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{E}_a$*, given an action* $a \in \boldsymbol{A}$ *at time* $t \in \boldsymbol{T}$*, reflects the probabilistic dependences, conditional on the effects of an action, among the outcomes of the two event variables at time t. The absence of a direct probabilistic influence between two event variables means that they are conditionally independent at time t, given the action.*

When two event variables are related by a probabilistic influence $\psi(a, \boldsymbol{x}, \boldsymbol{y}, t)$, read $\boldsymbol{x}$ influences $\boldsymbol{y}$ given a at t, the actual conditional probabilities involved are specified in a table or matrix associated with $\boldsymbol{y}$. The conditional distribution for the outcomes of $\boldsymbol{y}$, conditioned on the action a at time t, is specified for all possible combinations of the outcomes of its *probabilistic predecessors*, including $\boldsymbol{x}$, as indicated by the set of probabilistic influences leading into y. In other words, let $\boldsymbol{u}$

and $\boldsymbol{w}$ be the other two event variables influencing $\mathbf{y}$ through $\psi(a, \boldsymbol{u}, \boldsymbol{y}, t)$ and $\psi(a, \boldsymbol{w}, \boldsymbol{y}, t)$. The conditional distribution for the outcomes y_k of $\mathbf{y}$, conditioned on the outcomes x_p of $\boldsymbol{x}$, u_q of $\boldsymbol{u}$, w_r of $\boldsymbol{w}$, given action a at time t, is:

$$P\{y_k | x_p, u_q, w_r, a, t\}$$

The conditional distribution between two state variables $\boldsymbol{s}_n$ and $\boldsymbol{s}_{n-1}$, therefore, represents the set of all transition functions among the states, given action a in decision stage n.

In DynaMoL, the influence view supports graphical presentation of the event variables and probabilistic influences for an action at a particular decision stage. Conditional probabilities can be defined directly on the nodes representing the event variables.

An influence view for the action Quinidine in the case study is shown in Figure 6.3. We assume the same action-specific event variable space for both controlled actions NoDrug and Quinidine; only the probabilistic influences vary. The node labeled ***state-n*** is the state variable representing the state space $\boldsymbol{S}$ at decision stage n; the node labeled ***state*** is the *next-state variable* representing the states that are directly accessible from the state space at decision stage n.

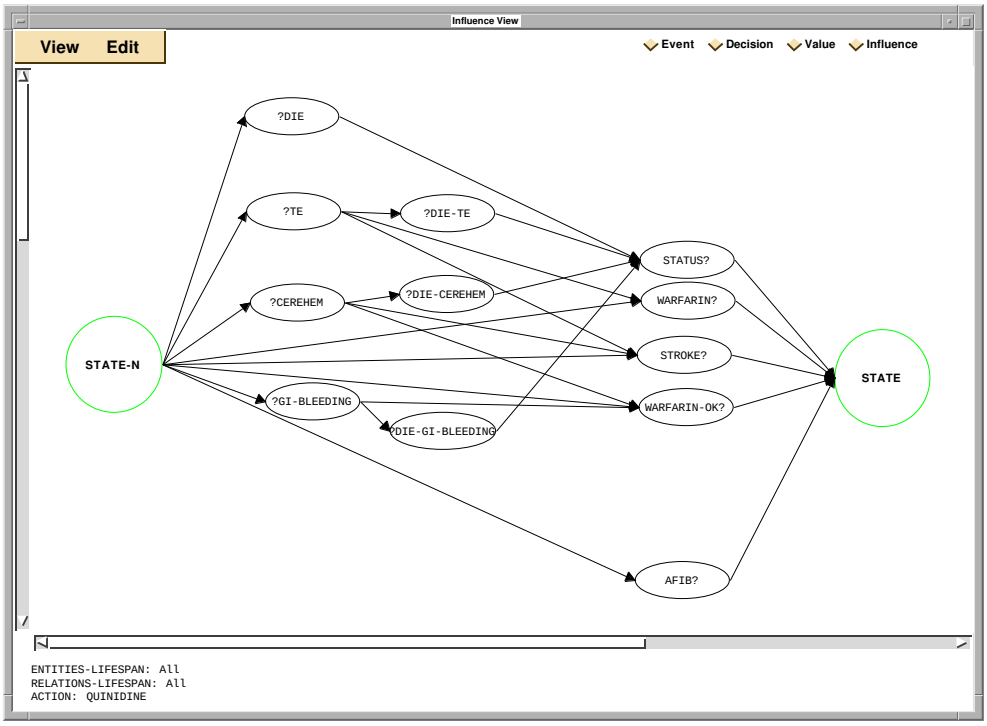


Figure 6.3 Influence view for action Quinidine in case study.

The perspective depicted in Figure 6.3, with the states represented as the outcomes of a state variable, is similar to dynamic influence diagrams in that conditional independence among the states and the event variables are explicitly shown. It is, however, a more detailed depiction since it corresponds to the effects of a single action. As mentioned, different actions may have different influence views. An alternate perspective, as shown in Figure 4.2, is similar to Markov cycle trees and stochastic trees in that the influence relationships and the individual states are clearly displayed. Conditional independence is not graphically shown in Markov cycle trees nor in stochastic trees; asymmetric and varying patterns of relevant intermediate event variables between individual states are not graphically displayed in dynamic influence diagrams.

6.2.7 Value Function Definition

Conditional on each controlled action, and with respect to the evaluation criterion, a set of *value functions* $\varsigma = \{ v_i^{(a)}(m) \mid i \in \boldsymbol{S};\ a \in \boldsymbol{A}, m = 0, 1, 2, \ldots \}$ are defined for the states.

Definition 4 (Value Function) *A value function* $v : \boldsymbol{A} \times \boldsymbol{S} \times \aleph \rightarrow \Re$ *determines the objective value achievable in state* $i \in \boldsymbol{S}$ *over time duration* $m \in \aleph$*, conditional on action* $a \in \boldsymbol{A}$*. It is defined in terms of the associated transition value functions* $v_{ij}^{(a)}(m)$ *for each possible transition* $\xi(a, i, j, .)$ *from state i to state j after staying in state i for duration n, conditional on the action a.*

Definition 5 (Transition Value Function) *A transition value function* $v_\xi : \boldsymbol{A} \times \boldsymbol{S} \times \boldsymbol{S} \times \aleph \rightarrow \Re$ *determines the objective value achievable in a transition from state* $i \in \boldsymbol{S}$ *to state* $j \in \boldsymbol{S}$ *over time duration* $m \in \aleph$*, conditional on action* $a \in \boldsymbol{A}$*. It has two components: a yield rate* $y_i^{(a)}(l)$ *and a bonus* $b_{ij}^{(a)}$*. The yield rate indicates the value achievable at time l after entering state i over time interval (l, l+1); the bonus is a one-time value achievable when the transition takes place from state i* [Howard, 1971]*. The value functions are assumed to be independent of the absolute time t.*

Formally:

$$v_i^{(a)}(m) = \sum_j q_{ij}^{(a)}(.)\, v_{ij}^{(a)}(m) = \sum_j q_{ij}^{(a)}(.) \left(\left[\sum_{l=0}^{n-1} y_i^{(a)}(l) \right] + b_{ij}^{(a)} \right); \qquad \text{(EQ 4)}$$

$$a \in \boldsymbol{A},\ \ i, j \in \boldsymbol{S},\ \ l, m \geq 0$$

In the case study, we adopt a special case for the value function: the yield rates and the bonus are constant for all states and transitions, and for all actions. In other words, $v_i^{(a)}(m) = v_{ij}(m) = my_i + b_{ij}$, for all $a \in \boldsymbol{A}$, $i, j \in \boldsymbol{S}$, $m \geq 0$. The yields are specified for the individual states, those for the absorbing states being

zero. Bonuses are negative in this case, indicating short-term morbidities; they are associated only with transitions in which some adverse events may occur, *e.g.,* the presence of stroke, cerebral hemorrhage, or gastro-intestinal bleeding.

In DynaMoL, the bonuses are currently specified with respect to the transitions, instead of their constituent event variables. Both dynamic influence diagrams and Markov cycle trees allows more detailed bonus specifications. In dynamic influence diagrams, the value functions usually contain only the yield rates for fixed time intervals; the bonuses can be incorporated by adding influence links from the chance nodes that are not state variables to the value nodes. This latter formulation, however, would complicate the structure of the model and tax the solution algorithm.

Similarly, the "toll" structures in Markov cycle trees allows bonuses to be associated with the chance nodes. A toll is a binding to an outcome of a chance node, indicating a positive or negative value associated with that outcome, *e.g.,* -0.25 quality-adjusted life months for a stroke. Tolls can be added or subtracted from the total expected value accumulated as the branches of the tree structure are traversed. This feature, however, is feasible only with forward induction based algorithms.

6.2.8 Constraint Management

So far we have described the basic components of a dynamic decision model formulated in DynaMoL. Definitions of these components can be subjected to a set of general or domain-specific constraints $\mathrm{K} = \{\kappa \mid \kappa \in \mathrm{K}_D \text{ or } \kappa \in \mathrm{K}_S\}$, where K_D is the set of *declaratory constraints* and K_S is the set of *strategic constraints*.

In general, a constraint κ corresponds to a logical or quantification sentence or *well-formed formula* (wff) in FOPC.

Declaratory Constraints

The set of declaratory constraints K_D concern the definitions of the states, event variables, and their associated relations.

Definition 6 (Declaratory Constraint) *A declaratory constraint is a relation* $\kappa_D \subseteq \{\boldsymbol{S} \cup \boldsymbol{\Omega_E} \cup \Xi \cup \Psi \cup \varsigma\}^n$; $n \geq 2$, *where* $\boldsymbol{S}$ *is the state space,* $\boldsymbol{\Omega_E}$ *is the event space representing the set of all outcomes of the event-variables in the event variable space* $\boldsymbol{E}$*, and n is the arity.*

Many of these constraints are inherent in the definitions supported by DynaMoL. For instance, conditional on an action $a \in \boldsymbol{A}$, an absorbing state $i \in \boldsymbol{S}$ has a zero value function and has no outgoing transitions to other states $j \in \boldsymbol{S}$:

$$\forall a \forall i \; \textit{absorbing}\,(a, i) \Rightarrow (v_i^{(a)}(.) = 0) \wedge \forall j \; (i \neq j) \wedge [\xi(a, i, j, .) \notin \Xi]$$

A similar example is the relevance of a particular set of event variables in describing a state transition conditional on an action. These constraints are imposed during model construction; their effects are explicitly encoded in the definitions of the structural or numerical parameters in the dynamic decision model. While translators can be devised in future to simplify the syntax for specifying such constraints, there is currently limited support for such translations in DynaMoL. For instance, an absorbing state is explicit defined, as illustrated by the Common Lisp code below:

```
(MAKE-STATE 'DEAD :ABSORBING-P T :VALUE-FUNCTION 0.0)
(MAKE-TRANSITION NODRUG DEAD DEAD :TRANS-FUCNTION 1.0)
(MAKE-TRANSITION QUINIDINE DEAD DEAD :TRANS-FUNCTION 1.0)
```

The Disjunctive Definition Constraint

There are some constraints whose effects need to be explicitly interpreted during inter-level or inter-perspective translations. An example of such constraints in DynaMoL is *disjunctive definition*, or *"partial OR-gate*," for event combinations. The partial OR-gate is analogous to a *canonical model* for specifying Bayesian networks.

Canonical models are default strategies for specifying the conditional distributions of the chance variables in a Bayesian network; they are used when detailed interactions among the variables are unavailable, too numerous to elicit, or too complex to be determined precisely [Pearl, 1988]. A common canonical form used to reduce the number of conditional probabilities that need to be assessed is *disjunctive interaction*, or "*noisy-OR-gate*."

The partial OR-gate is devised to facilitate conditional distribution specifications of event variables. Consider a partial set of event variables in the influence view for the action Quinidine in the case study, as shown in Figure 6.4. The variable ***die*** indicates death from age, sex, or race related causes, ***die-TE*** indicates death from thromboembolism, ***die-cerehem*** indicates death from cerebral hemorrhage, and ***die-GI-bleeding*** indicates death from gastro-intestinal bleeding.

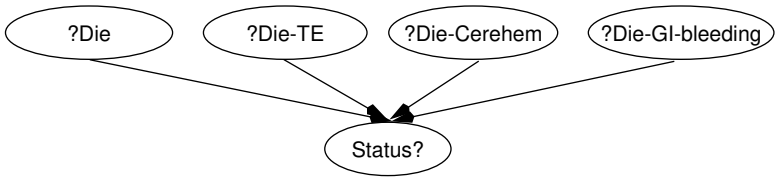


Figure 6.4 A partial influence view in case study. The nodes represent event-variables, the links probabilistic influences.

The probability distribution of the outcomes of the state attribute variable ***status*** in Figure 6.4 is conditioned on the Cartesian or cross product of the outcomes of its probabilistic predecessors:

```
(distribution status?/Quinidine)
#<Distribution-Table #:G725
    Type: DISCRETE
    Matrix: #:G725
         "Pr{c|r}"                                ALIVE-A          DEAD-A
(DIE DIE-GIB DIE-CHM DIE-TE)                      0.00000          1.00000
(DIE DIE-GIB DIE-CHM NOT-DIE-TE)                  0.00000          1.00000
(DIE DIE-GIB NOT-DIE-CHM DIE-TE)                  0.00000          1.00000
(DIE DIE-GIB NOT-DIE-CHM NOT-DIE-TE)              0.00000          1.00000
(DIE NOT-DIE-GIB DIE-CHM DIE-TE)                  0.00000          1.00000
(DIE NOT-DIE-GIB DIE-CHM NOT-DIE-TE)              0.00000          1.00000
(DIE NOT-DIE-GIB NOT-DIE-CHM DIE-TE)              0.00000          1.00000
(DIE NOT-DIE-GIB NOT-DIE-CHM NOT-DIE-TE)          0.00000          1.00000
(NOT-DIE DIE-GIB DIE-CHM DIE-TE)                  0.00000          1.00000
(NOT-DIE DIE-GIB DIE-CHM NOT-DIE-TE)              0.00000          1.00000
(NOT-DIE DIE-GIB NOT-DIE-CHM DIE-TE)              0.00000          1.00000
(NOT-DIE DIE-GIB NOT-DIE-CHM NOT-DIE-TE)          0.00000          1.00000
(NOT-DIE NOT-DIE-GIB DIE-CHM DIE-TE)              0.00000          1.00000
(NOT-DIE NOT-DIE-GIB DIE-CHM NOT-DIE-TE)          0.00000          1.00000
(NOT-DIE NOT-DIE-GIB NOT-DIE-CHM DIE-TE)          0.00000          1.00000
(NOT-DIE NOT-DIE-GIB NOT-DIE-CHM NOT-DIE-TE)      1.00000          0.00000
```

In the conditional distribution table shown above, each matrix entry is read as: "the probability of <column index> conditional on <row index>." The suffixes "-A" for the column indices simply indicate that they are the outcomes of a state attribute variable. The indices are conjunctions of events. This specification, as we realize, is not only cumbersome, but unrealistic with respect to the domain knowledge. We want to express: a patient is dead if he dies from natural causes or dies from a stroke or dies from cerebral hemorrhage or dies from gastro-intestinal bleeding. In other words, we want to specify some indices of the conditional distribution table as a disjunction of events. By incorporating the partial OR-gate, we can specify the conditional distribution table for the state attribute variable ***status*** above as follows, where the square-bracketed index is a disjunction:

```
(distribution status?/Quinidine)
#<Distribution-Table #:G725
    Type: DISCRETE
    Matrix: #:G725
         "Pr{c|r}"                                ALIVE-A          DEAD-A
[DIE DIE-GIB DIE-CHM DIE-TE]                      0.00000          1.00000
(NOT-DIE NOT-DIE-GIB NOT-DIE-CHM NOT-DIE-TE)      1.00000          0.00000
```

With respect to the generalized noisy OR model developed by [Srinivas, 1992], of which the binary noisy OR-gate as described in [Pearl, 1988] is a special case, the partial OR-gate is different in the following manners. First, in the DynaMoL vocabulary, the former is a function on the event variable space $\boldsymbol{E}$, while the latter is a function on the event space $\Omega_{\boldsymbol{E}}$. Second, in the former, the combination constraint is imposed on an event variable and all its predecessors, while in the latter, the combination constraint can be imposed on an event variable and any subset of its predecessors, hence the name "partial."

Formally, the disjunctive definition constraint has the general form:

$$e_1 \vee e_2 \vee \ldots \vee e_k \Rightarrow e_c$$

where e_i, e_2, …, e_k are the *constraining events* and e_c is the *consequent event.* The constraint is read as: "If e_i or e_2 or ... or e_k, then e_c." The constraint is imposed on the specification and interpretation of the conditional distribution table of an event variable $\boldsymbol{x}$. The constraining events are outcomes of the probabilistic predecessors of $\boldsymbol{x}$; the consequent event is an outcome of $\boldsymbol{x}$.

In DYNAMO, a user can explicitly specify the partial OR-gate constraints for the different event variables in terms of a set of logical statements as shown above. The conditional distribution tables with the correct dimensions and labels are automatically created. The numerical parameters involved can then be specified accordingly.

For simplicity, DYNAMO currently allows only one partial OR-gate constraint to be imposed on each event variable. In other words, only one outcome of an event variable can be a consequent event. There are no theoretical impediments to relaxing this assumption.

The partial OR-gate is rather useful in the case study. In addition to the example shown in Figure 6.4, another example is the definition of the ***cerebral vascular accident*** state attribute variable: a patient will be in a state with the attribute *stroke* if he has a permanent stroke, a cerebral hemorrhage, or had a stroke before.

Other Declaratory Constraints

Other declaratory constraints such as numeric restrictions on the events are expressed in terms of additional state attribute variables. For instance, if a patient is assumed to be dead after having two strokes, we need to define the number of strokes as a state attribute variable to incorporate the assumption. In general, a constraint can usually be expressed in terms of a state attribute variable. But doing so will lead to an explosion of the state space size. We discuss in more detail how this can be avoided in Chapter 8.

Strategic Constraints

The set of strategic constraints K_S concern the definitions of the actions.

Definition 7 (Strategic Constraint) *A strategic constraint is a relation* $\kappa_S \subseteq \{\boldsymbol{S} \cup \Omega_{\boldsymbol{E}} \cup \Xi \cup \Psi \cup \varsigma\}^n$; $n \geq 2$, *where* $\boldsymbol{S}$ *is the state space,* $\Omega_{\boldsymbol{E}}$ *is the event space representing the set of all outcomes of the event-variables in the event variable space* $\boldsymbol{E}$*, and n is the arity.*

Again most of these constraints are inherent in the definitions supported by DynaMoL. For instance, all the controlled actions $a \in \boldsymbol{A}$ are assumed to be applicable in all the states $s \in \boldsymbol{S}$ at all times $t \in \boldsymbol{T}$:

$$\forall a \forall s \forall t \;\; applicable\ (a, s, t)$$

Extension to include state-dependent or state- and time-dependent action space as described in Section 3.3.1 is equivalent to designating the states in which an action

is not applicable as absorbing states with no out transitions and with zero value functions.

In the case study, all the controlled actions are assumed to be applicable in all the states in the state space. Since warfarin is modeled as an embedded action, its effects are not taken into account in the states where warfarin is ineligible.

Numeric restriction and order restriction on the actions are currently expressed directly in terms of extra state attribute variables. Chapter 8 discusses how additional grammar constructs and their corresponding translators may be incorporated to manage such constraints in DynaMoL.

6.2.9 Parametric Assessment

The numbers, functions, and probability distributions associated with the variables and relations are usually assessed after the structure of the model is in place. As in all other dynamic decision modeling frameworks, DynaMoL does not currently provide explicit guidance or specification on how such parameters should be assessed, except for checking the types of the parameters.

A set of definitional guidelines and techniques for directly specifying one-step transition functions in an SMDP, corresponding to the transition view in DynaMoL, is documented in Appendix B. When direct estimation of the one-step transition functions is difficult, the same set of numeric parameters associated with the transitions can be specified in the influence view, in terms of their constituent event variables instead.

Specification for the conditional distributions in the influence view is straightforward for the Markov case. In this case, each entry $C[x, y]$ in a conditional distribution table or matrix C is simply:

$$P\{y|\, x, a, t\} \qquad \textbf{(EQ 5)}$$

subject to

$$\sum_{y} P\{y|\, x, a, t\} = 1$$

where x is a row index representing a combination of outcome events of all the probabilistic predecessors for an event variable $\boldsymbol{y}$, $y \in \Omega_{\boldsymbol{y}}$ is a column index representing an outcome of $\boldsymbol{y}$, $a \in \boldsymbol{A}$, and $t \in \boldsymbol{T}$. The entry, which is usually a function $f(t)$ of time, indicates the probability of event y in the next decision stage one time unit later, given that event x occurs and action a taken at the decision stage at time t.

For the semi-Markov case, there are a few different ways to assess the one-step transition functions in terms of the constituent event variables. Consider specifying the PMFs of one-step transition functions, $q_{ij}^{(a)}(.)$. Recall that the PMFs of one-

step transition functions can be defined either in terms of the transition probabilities $P_{ij}^{(a)}(.)$ and holding time PMFs $h_{ij}^{(a)}(.)$, such that

$$q_{ij}^{(a)}(m,t) = P_{ij}^{(a)}(t)\, h_{ij}^{(a)}(m,t);$$

$$a \in \boldsymbol{A}, i,j \in \boldsymbol{S}, t \in \boldsymbol{T}, m \geq 0$$

or the conditional transition probabilities $p_{ij}^{(a)}(.)$ and waiting time PMFs $w_i^{(a)}(.)$, such that

$$q_{ij}^{(a)}(m,t) = p_{ij}^{(a)}(m,t)\, w_i^{(a)}(m,t);$$

$$a \in \boldsymbol{A}, i,j \in \boldsymbol{S}, t \in \boldsymbol{T}, m \geq 0$$

The first approach is to assess the probabilities of the transition destinations first, then decide on the holding times with respect to those transitions. In this case, we interpret the probability given in (EQ 5), which is a function $f(t)$ of time, as an eventual transition probability, *i.e.,* the probability of event y over those of other outcomes of $\boldsymbol{y}$, given that event x occurs and action a is taken at the decision stage at time t. The collection of conditional distributions in the influence view will then constitute the eventual transition probabilities $P_{ij}^{(a)}(t)$, where $i,j \in \boldsymbol{S}, a \in \boldsymbol{A}, t \in \boldsymbol{T}$ for the underlying SMDP. We then estimate the holding time PMFs directly for the corresponding transitions. In the clinical context, this approach first determines the transition destination of a patient, and then depending on this destination, estimates the time duration spent in the current state before making the transition. For instance, there is a probability of 0.3 that a patient who has undergone surgery A will develop complication B; the probability for such a patient to develop the complication is 0.2 in the first month after surgery, 0.5 in the second month, and 0.3 in the third month.

The second approach is to decide on the waiting times first, then assess the probabilities of transition destinations with respect to those waiting times. In this case, we directly estimate the waiting time PMFs for each state $s \in \boldsymbol{S}$. We then interpret the probability given in (EQ 5), which is now a function $f(m,t)$ of both duration and time, as a conditional transition probability, *i.e.,* the probability of event y over those of other outcomes of $\boldsymbol{y}$, given that event x occurs, action a is taken, and the waiting time is m at the decision stage at time t. The collection of conditional distributions in the influence view will then constitute the conditional transition probabilities $p_{ij}^{(a)}(m,t)$, where $i,j \in \boldsymbol{S}, a \in \boldsymbol{A}, t \in \boldsymbol{T}, m \geq 0$ for the underlying SMDP. In the clinical context, this approach first estimates the time duration a patient spends in the current state before making a transition, and then depending on the this duration, determines the transition destination. For instance, the probability that a patient who has undergone surgery A will remain well is 0.2 for a month after surgery, 0.5 for two months and 0.3 for three months; if the patient is well, the probability for him to develop complication B is 0.6 in the first month after surgery, 0.5 in the second month, and 0.3 in the third month.

Validity, adequacy, and relevance of the numeric parameters should be determined by employing external techniques when appropriate. In future, such techniques could be incorporated into the DynaMoL ontology as additional constraints.

In the case study, the distributions involved are estimated from the medical literature and derived from statistical tables such as life tables. Since the relevant data for the occurrence of an event are usually reported in terms of yearly or monthly rates instead of probabilities, the following equation is used to convert a constant rate *r* into a probability:

$$P = 1 - e^{-r\tau} \qquad \textbf{(EQ 6)}$$

where τ is the unit of the rate with respect to the time unit of the decision horizon, *e.g.,* to derive a monthly probability, $\tau = 1$ for a monthly rate, and $\tau = 1/12$ for a yearly rate.

The first dynamic decision problem in the case study is formulated as a Markov decision problem. An example conditional distribution table for the variable ***die-TE*** conditional on the action NoDrug is shown below:

```
> (distribution ?die-te/NoDrug)
#<Distribution-Table #:G721
    Type:               DISCRETE
    Matrix: #:G721
         "Pr{c|r}"                DIE-TE          NOT-DIE-TE
          PERM-STR                PDIESTROKE      (- 1 PDIESTROKE)
         TRANS-STR                PDIESTROKE      (- 1 PDIESTROKE)
               TIA                0.00000         1.00000
            NOT-TE                0.00000         1.00000
```

The second dynamic decision problem in the case study is formulated as a semi-Markov decision problem. Most of the numeric parameters are identical to those for the first problem. We assume that the relative rate of bleeding for warfarin varies with the duration *m* since entering a state *s* according to a function of the form:

$$A + Be^{-Cm} \qquad \textbf{(EQ 7)}$$

where $A, B, C \in \Re^+$. Figure 6.5 compares the constant and varying relative risks of bleeding for warfarin assumed for the first and second dynamic decision problems respectively.

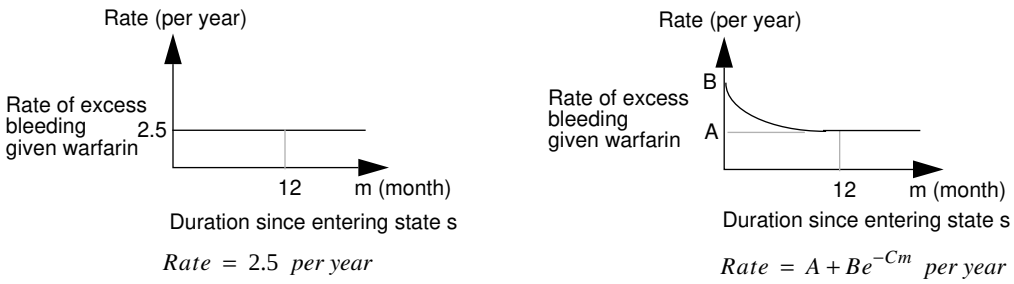


Figure 6.5 Constant and varying risks of bleeding for warfarin in case study.

With respect to the varying risk as described above, we assess the corresponding conditional distributions of the event variables as the conditional transition probabilities. An example conditional distribution table for the variable ***GI-bleeding*** conditional on the action NoDrug is shown below:

```
> (distribution ?gi-bleeding/nodrug)
#<Distribution-Table #:G727
    Type:               DISCRETE
    Matrix: #:G727
         "Pr{c|r}"                GIB              NOT-GIB
      NSR-N-WOFF-N                PGIB             (- 1 PGIB)
       NSR-N-WON-N                PGIB-WARF        (- 1 PGIB-WARF)
       NSR-N-WNE-N                PGIB             (- 1 PGIB)
    NSR-TIA-WOFF-N                PGIB             (- 1 PGIB)
     NSR-TIA-WON-N                PGIB-WARF        (- 1 PGIB-WARF)
     NSR-TIA-WNE-N                PGIB             (- 1 PGIB)
    NSR-STR-WOFF-N                PGIB             (- 1 PGIB)
     NSR-STR-WON-N                PGIB-WARF        (- 1 PGIB-WARF)
     NSR-STR-WNE-N                PGIB             (- 1 PGIB)
       AF-N-WOFF-N                PGIB             (- 1 PGIB)
        AF-N-WON-N                PGIB-WARF        (- 1 PGIB-WARF)
        AF-N-WNE-N                PGIB             (- 1 PGIB)
     AF-TIA-WOFF-N                PGIB             (- 1 PGIB)
      AF-TIA-WON-N                PGIB-WARF        (- 1 PGIB-WARF)
      AF-TIA-WNE-N                PGIB             (- 1 PGIB)
     AF-STR-WOFF-N                PGIB             (- 1 PGIB)
      AF-STR-WON-N                PGIB-WARF        (- 1 PGIB-WARF)
      AF-STR-WNE-N                PGIB             (- 1 PGIB)
```

where *pgib-warf* is a function indicating the probability of developing gastro-intestinal bleeding given warfarin after entering the state for duration *m*.

We assume the waiting time PMFs for the states affected by the varying relative risk of bleeding for warfarin are functions of duration *m* with the same form as indicated in (EQ 7). This means that a patient on warfarin is more likely to make a transition out of that state sooner than later. Due to a lack of actual clinical data, this assumption is made to help evaluate the parameter specification and computation processes in DynaMoL more than to reflect the clinical significance.

The complete list of the numeric parameters in the case study can be found in Appendix C.

6.3 Model Translation

Most of the constructs in a dynamic decision model formulated in DynaMoL have direct correspondence in its mathematical representation as an SMDP. Constructs without direct correspondence are translated by a set of translators $\Gamma = \{\gamma_\chi | \chi \in C \subseteq \boldsymbol{A} \cup \boldsymbol{S} \cup \boldsymbol{E} \cup \Omega_{\boldsymbol{E}} \cup \Xi \cup \Psi \cup \mathrm{K}\}$. The translators can be inter-level or inter-perspective.

Definition 8 (Translator) *A translator is a function* $\gamma_\chi : C \rightarrow \boldsymbol{A} \cup \boldsymbol{S} \cup \boldsymbol{E} \cup \Omega_{\boldsymbol{E}} \cup \Xi \cup \Psi \cup \varsigma \cup \mathrm{K}$, *where C is a language construct such that* $C \subseteq \{\boldsymbol{A} \cup \boldsymbol{S} \cup \boldsymbol{E} \cup \Omega_{\boldsymbol{E}} \cup \Xi \cup \Psi \cup \varsigma \cup \mathrm{K}\}^n$, *where* $\boldsymbol{A}$ *is the action space,* $\boldsymbol{S}$ *is the state space,* $\boldsymbol{E}$ *is the event variable space,* $\Omega_{\boldsymbol{E}}$ *is the event space,* Ξ *is the set of transitions,* Ψ *is the set of influences,* K *is the set of constraints, and n is the arity.*

6.3.1 Translating Event Variables and Influences

The mathematical formulation of SMDP does not involve event variables and influences. The probabilities involved in the event variables and the corresponding influences between any two states, therefore, should be translated into the one-step transition functions characterizing the transitions between the two states.

A translator called *influence view translator* is devised in DynaMoL to handle this task. The algorithm is based on the expectation of conditional probabilities; it is analogous to the chance node reduction algorithm in influence diagrams [Shachter, 1986].

Given random variables $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$, conditional expectation of $\boldsymbol{z}$ given $\boldsymbol{x}$ with respect to $\boldsymbol{y}$ means, for $x \in \Omega_{\boldsymbol{x}}$, $y \in \Omega_{\boldsymbol{y}}$, $z \in \Omega_{\boldsymbol{z}}$:

$$P\{z|x\} = \sum_i P\{z|y_i\} \cdot P\{y_i|x\} \qquad \textbf{(EQ 8)}$$

With reference to Figure 6.3, the overall idea of this translation algorithm is to reduce the intermediate event variable nodes between the two state variable nodes, so that the final diagram contains only a direct influence between the two state variables as shown in Figure 6.6. Assuming static action space and static state space, the conditional distribution table associated with the state variable on the right contains the set of PMFs or CDFs for the one-step transition functions, eventual transition probabilities, or conditional transition probabilities for the transitions, conditional on an action.

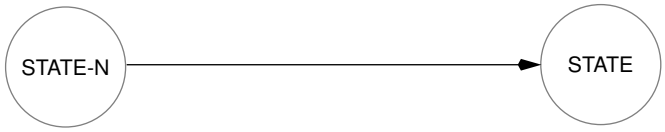


Figure 6.6 Final influence view after reduction by the influence translator.

In essence, the influence view translator iteratively identifies an event variable node to be reduced, updates the conditional distributions of other event variables affected by it, and removes it. The main algorithm is as follows:

INFLUENCE-VIEW-TRANSLATOR (ID)

◊ ID is a sequence of event-variable-nodes
x ← FIND-REMOVABLE-EVENT-VARIABLE (ID)
while x
  do ID ← ABSORB-EVENT-VARIABLE (ID, x)
    x ← FIND-REMOVABLE-EVENT-VARIABLE (ID)
return ID

An event variable node is removable only if it has a single probabilistic successor. A simple heuristic is employed to remove event variable nodes with smaller conditional distribution dimensions first. This heuristic helps keep the size of the conditional distribution tables to be updated as small as possible.

FIND-REMOVABLE-EVENT-VARIABLE (ID)

◊ ID is a sequence of event-variable-nodes
E-list ← ∅
for each x ∈ ID
  unless x = state-variable or x = next-state-variable
    do if length[successors[x]] = 1
      then E-list ← E-list ∪ {x}
◊ Sort elements of E-list according to the increasing number of predecessors nodes and the increasing size of (lone) successor.
return first[E-list]

To remove an event variable node ***x***, its lone successor ***y*** must inherit its predecessors, and the conditional distribution of ***y*** must be updated accordingly. Similarly, ***x*** will no longer be a successor to all its predecessors.

ABSORB-EVENT-VARIABLE (ID, x)

◊ ID is a sequence of event-variable-nodes, x is an event-variable-node.
D_x ← distribution-table[x]
y ← first[successors[x]] ◊ Assume x has only one successor.

$D_y \leftarrow$ distribution-table[y]
predecessors[y] $\leftarrow$ predecessors[x] $\cup$ predecessors[y]
distribution[y] $\leftarrow$ new-distribution ()
UPDATE-DISTRIBUTION-TABLE (distribution-table[x], D_x, D_y)
for each p $\in$ predecessors[x]
  do successors[p] $\leftarrow$ successors[p] $\cup$ {y} \ x
return ID \ x

Updating the conditional distribution table of the successor event variable begins with the proper combination of conditioning events with respect to its new predecessors. Recall that the default row and column indices in the conditional distribution table are conjunctions of the conditioning events; each event in the sequence constituting an index is an outcome of a predecessor event variable. The conditioning events from the event variable to be removed must be filtered from each combination appropriately.

Consider the conditional distribution table *X* for the event variable ***x*** to be removed, the conditional distribution table *Y* for the lone successor ***y*** of ***x***, and the new conditional distribution table *Z* for ***y***. The event variables ***x*** and ***y*** may have common predecessors. Figure 6.7 shows a general organization of the event variable nodes involved and the indexing convention of their conditional distribution tables. The indices shown are sets of conjunctive events. An index notation $index_1 index_2$ indicates a union of the two sets. The indices can be interpreted as follows:

- *k* is an outcome of ***x***, which is the event variable to be removed.
- i_1 is a cross product of the outcomes of the predecessors of ***x***.
- *j* is an outcome of ***y***, which is the successor event variable of ***x***.
- i_2 is a cross product of the outcomes of the predecessors, minus ***x***, of ***y***.

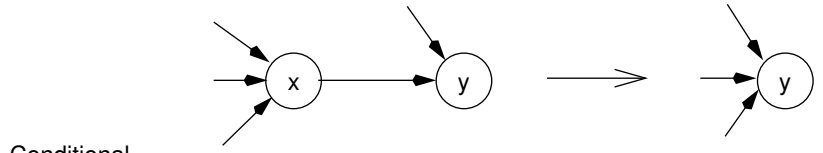
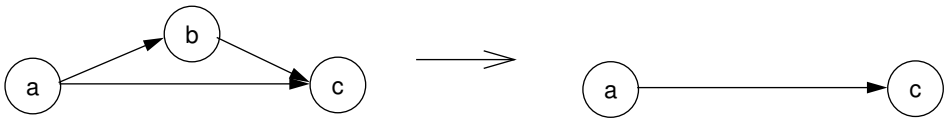


Figure 6.7 Indexing convention for updating conditional distribution table.

The entries of the updated conditional distribution table *Z* is related to those of tables *X* and *Y* by the following equation:

$$Z[i,j] \;=\; Z[i_1 i_2, j] \;=\; \sum_k X[i_1, k] \cdot Y[i_2 k, j] \qquad \textbf{(EQ 9)}$$

Figure 6.8 shows a simple example of the calculations as described in (EQ 9).



Conditional distribution table for b:

"P(c|r)"	B1	B2	B3
A	0.70	0.20	0.10
NOT-A	0.30	0.50	0.20

Conditional distribution table for c:

"P(c|r)"	C	NOT-C
(A B1)	0.70	0.30
(NOT-A B1)	0.60	0.40
(A B2)	0.45	0.55
(NOT-A B2)	0.10	0.90
(A B3)	0.20	0.80
(NOT-A B3)	0.30	0.70

Updated conditional distribution table for c, after b is removed:

"P(c|r)"	C	NOT-C
A	0.60	0.40
NOT-A	0.29	0.71

Figure 6.8 An example to show conditional distribution table updating.

The actual updating algorithm is as follows:

UPDATE-DISTRIBUTION-TABLE (Z, X, Y)

◊ Z, X, and Y are (conditional) distribution tables. Z is the new table; X is the table of the removed event-variable; Y is the table of the (lone) successor event variable.
I ← row-indices[Z]
◊ Indices are sets of labels indicating conjunctions of events.
◊ Row-indices are the conditioning events; column-indices are the outcome events.
J ← column-indices[Z]
K ← column-indices[X]
X-conditioning-events ← ∪ elements[row-indices[X]] ◊ "Flatten" the set.
Y-conditioning-events ← ∪ elements[row-indices[Y]]
for each i ∈ I
  do i_1 ← i ∩ X-conditioning-events
    i_2 ← i ∩ Y-conditioning-events
    for each j ∈ J
      do Z[i, j] ← 0
        for each k ∈ K
          do Z[i, j] ← Z[i, j] + X[i_1, k] * Y[i_2k, j]

Assuming that each event variable involved has m outcomes and n predecessors on average, the number of conditional distribution table entries to be updated is $O(m^{n+1})$.

For the influence view of the case study as shown in Figure 6.3, the state variable and the next-state variable have 18 and 19 outcomes each, and the other event variables have an average number of 3 outcomes. Applying the influence view translator algorithm as described above, the most common table updating involves the next-state variable, with about 3 other event variables and the state variable as its predecessors. To reduce an event variable from an influence view, the number of table entries that need to be calculated is of the order of 8700; the addition of one more predecessor would make it to the order of 26,000! This is the main motivation for introducing the partial OR-gate, which drastically cuts down the sizes of the intermediate conditional distribution tables in practice.

6.3.2 Translating Declaratory Constraints

Most declaratory constraints have direct correspondence in the SMDP representation. Each state variable, for instance, corresponds to the state-space at a particular time or decision stage in the SMDP.

The only explicit translator currently defined in DynaMoL is a translator for the partial OR-gate constraint. Recall that this constraint can be expressed in terms of an implication statement. The antecedent contains the constraining events in disjunctive form; the consequent represents the target definition for the constraining events. The corresponding translator is encoded as a special version of the update distribution table algorithm described in Section 6.3.1. This new algorithm is used when the event variable $\boldsymbol{x}$ to be removed, or its lone successor $\boldsymbol{y}$, or both contain conditional distribution tables with the partial OR-gate constraints. In other words, some of the indices of these tables are sequences or sets of labels representing disjunctions of events. Recall also that currently only one partial OR-gate can be imposed on each event variable.

Let $X[i_1, k]$ be the table entries for $\boldsymbol{x}$, $Y[i_2 k, j]$ be the table entries for $\boldsymbol{y}$, and $Z[i, j]$ be the table entries for the updated table of $\boldsymbol{y}$. A constraining row index is a row index in disjunctive form, representing all of the constraining events in the partial OR-gate. A constraining *partial* row index is a row index in disjunctive form, representing *some* of the constraining events in the partial OR-gate. For instance, `[DIE DIE-GIB DIE-CHM DIE-TE]` is a constraining row index, `[DIE-GIB DIE-TE]` is a constraining partial row index for the partial OR-gate defined with respect to Figure 6.4. Similar, a constrained column index represents the consequent event in a partial OR-gate. For instance, `DEAD-A` is a constrained column index for the above example.

The translator needs to consider three cases:

Case 1: Constraint imposed on the event variable $\boldsymbol{x}$

- For constraining row index i_1 in X, the entry to be summed into $Z[i, j]$ is:

$$X[i_{1C}, k] \cdot Y[i_2 k, j]$$

  where i_{1C} is the constraining row index in *X*.

- For normal row indices i_1 in X, the entries to be summed into $Z[i, j]$ are:

$$X[i_1, k] \cdot Y[i_2 k, j]$$

Case 2: Constraint imposed on the successor event variable $\boldsymbol{y}$

- For constraining partial row indices k in Y, the entries to be summed into $Z[i, j]$ are:

$$X[i_1, k] \cdot Y[i_{2C}, j]$$

  where i_{2C} is the constraining row index in *Y.* The partial row indices i_2 do not matter in these calculations because they will be filtered out appropriately.

- For "normal" or non-constraining partial row indices k in Y, the entries to be summed into $Z[i, j]$ are:

$$X[i_1, k] \cdot Y[i_2 k, j]$$

Case 3: Constraint imposed on both $\boldsymbol{x}$ *and* $\boldsymbol{y}$

- For constraining row index i_1 in X and constraining partial row indices k in Y, the entries to be summed into $Z[i, j]$ are:

$$X[i_{1C}, k] \cdot Y[i_{2C}, j]$$

  where i_{1C} is the constraining row index in *X*, and i_{2C} is the constraining row index in *Y.*

- For normal row indices i_1 in X and constraining partial row indices k in Y, the entries to be summed into $Z[i, j]$ are:

$$X[i_1, k] \cdot Y[i_{2C}, j]$$

- For constraining row index i_1 in X, and normal partial indices k fin Y, the entries to be summed into $Z[i, j]$ are:

$$X[i_{1C}, k] \cdot Y[i_2 k, j]$$

- For normal row indices i_1 in X, and normal partial row indices k in Y, the entries to be summed into $Z[i, j]$ are:

$$X[i_1, k] \cdot Y[i_2 k, j]$$

  The algorithm is as follows:

UPDATE-DISTRIBUTION-TABLE-WITH-CONSTRAINTS (Z X, Y)

◊ Z, X, and Y are (conditional) distribution tables. Z is the new table; X is the table of the removed event-variable; Y is the table of its (lone) successor.
I ← row-indices[Z]
◊ Indices are sequences or sets of labels indicating conjunctions or disjunctions of events.
◊ Row-indices are the conditioning events; column-indices are the outcome events.
J ← column-indices[Z]
K ← column-indices[X]
i_{1C} ← constraining-row-index[X]
i_{2C} ← constraining-row-index[Y]
X-conditioning-events ← ∪ elements[row-indices[X]] ◊ "Flatten" the set.
Y-conditioning-events ← ∪ elements[row-indices[Y]]
for each i ∈ I
do i_1 ← i ∩ X-conditioning-events
i_2 ← i ∩ Y-conditioning-events
for each j ∈ J
do Z[i, j] ← 0
for each k ∈ K
do if constraining? (i_1)
then if constraining? (k)
then Z[i, j] ← Z[i, j] + X[i_{1C}, k] * Y[i_{2C}, j]
else Z[i, j] ← Z[i, j] + X[i_{1C}, k] * Y[i_2k, j]
else if constraining? (k)
then Z[i, j] ← Z[i, j] + X[i_1, k] * Y[i_{2C}, j]
else Z[i, j] ← Z[i, j] + X[i_1, k] * Y[i_2k, j]

The number of table entries to be updated is of about the same order as before: $O((m-1)^{n+1})$ where m is the average number of outcomes and n is the average number of predecessors of an event variable. The smaller dimensions of the conditional distribution tables rendered by the partial OR-gate, however, makes a big difference in computational costs in practice.

Explicit translations for other declaratory constraints such as numeric restriction on the events are not currently handled in DynaMoL.

6.3.3 Translating Strategic Constraints

The only strategic constraints supported by the current version of DynaMoL ontology are on the applicability of the actions in the states. Given the state space, the states in which an action are not applicable are specified as absorbing states. No explicit translators are devised for such strategic constraints.

7 Dynamic Decision Model Solution and Analysis

Once the dynamic decision model is formulated and translated into an SMDP, the model can be solved by any solution methods for SMDPs. Analysis can be carried out on both the model and its corresponding SMDP. This chapter discusses some applicable solution methods and the types of relevant analysis.

7.1 Solution Methods

A variety of solution methods are available for SMDPs. We examine in detail the two solution methods implemented in the current version of DYNAMO*: value-iteration* and *fundamental matrix solution.* We also briefly mention other available methods. Admitting various solution methods is a major feature of the DynaMoL architecture. Based on the characteristics of the dynamic decision problem concerned, we can choose the most efficient solution method available.

7.1.1 Value Iteration

A dynamic decision problem can be expressed as the dynamic programming equation, or Bellman optimality equation, of an SMDP. For a decision horizon of n stages, with a discount factor $0 \le \beta < 1$, the optimal value achievable in any state $i \in \boldsymbol{S}$, V_i^*, given an initial value $V_i(0)$ and current time $t \in \boldsymbol{T}$, is [Howard, 1971]:

$$V_i^*(n, \beta, t)$$

$$= max_a \{ \sum_j \sum_{m=n+1}^{\infty} q_{ij}^{(a)}(m, t) \left[\sum_{l=0}^{n-1} \beta^l y_i^{(a)}(l) + \beta^n V_i(0) \right]$$

$$+ \sum_j \sum_{m=1}^{n} q_{ij}^{(a)}(m, t) \left[\sum_{l=0}^{m-1} \beta^l y_i^{(a)}(l) + \beta^m b_{ij}^{(a)} + \beta^m V_j^*(n-m, \beta, t+m) \right] \};$$

$$n \geq 0, \quad i, j \in \boldsymbol{S}, \quad a \in \boldsymbol{A}, \quad t \in \boldsymbol{T}$$

(EQ 10)

where

$$q_{ij}^{(a)}(m, t) = P_{ij}^{(a)}(t) \cdot h_{ij}^{(a)}(m, t)$$

is the one-step transition function from state i to j, conditional on action a, for duration m after entering a state i at time t; and

$$v_{ij}^{(a)}(m) = [\sum_{l=0}^{m-1} y_i^{(a)}(l)] + b_{ij}^{(a)}$$

is the transition value function with yield rate $y_i^{(a)}(.)$ and bonus $b_{ij}^{(a)}$ for transition $\xi(a, i, j, t)$ such that

$$v_i^{(a)}(m) = \sum_j q_{ij}^{(a)}(.)\, v_{ij}^{(a)}(m)$$

denotes the objective value achievable in each state $i \in \boldsymbol{S}$ over duration m, conditional on action $a \in \boldsymbol{A}$.

The first addend in (EQ 10) indicates the expected value achievable if the next transition out of state i occurs after time duration n, and the second addend indicates the expected value achievable if the next transition occurs before time duration n. In the specialized case of an MDP, (EQ 10) specializes into (EQ 2') as described in Section 2.3 and reproduced below:

$$V_i^*(n, \beta, t) = max_a \{ v_i^{(a)}(1) + \beta \sum_j P_{ij}^{(k)}(t) \cdot V_j^*(n-1, \beta, t+1) \};$$

$$n \geq 0, \quad i, j \in \boldsymbol{S}, \quad a \in \boldsymbol{A}, \quad t \in \boldsymbol{T} \qquad \textbf{(EQ 11)}$$

The value iteration solution method is to solve the optimality equation shown in (EQ 10) or (EQ 11). The solution to such an equation is an optimal policy

$$\pi^* = \{ \mu_0^*, \mu_1^*, \mu_2^*, \mu_3^*, \ldots \}$$

where $\mu_t^*: \boldsymbol{S} \rightarrow \boldsymbol{A}$ and $t \in \boldsymbol{T}$, that maximizes $V_{init}^*(N, \beta, 0)$, the optimal expected value or reward for an initial state over duration N, at time $t = 0$. Recall that $n = N - t$ is the remaining duration for the decision horizon.

For a decision horizon of duration N, the complexity of the value iteration algorithm for SMDP is $O(N^2 \cdot |\boldsymbol{A}| \cdot |\boldsymbol{S}|^2)$, since in each decision stage, we generally need to consider all time points t' such that $t \le t' \le t + n$. The complexity of the corresponding algorithm for MDP is $O(N \cdot |\boldsymbol{A}| \cdot |\boldsymbol{S}|^2)$; in this case there is a one-to-one correspondence between the duration n and the current time t: $n = N - t$.

As mentioned, all the default solution methods in existing dynamic decision modeling techniques are based on value iteration. While probabilistic inference techniques can also be employed for solving dynamic influence diagrams [Shachter and Peot, 1992], the graph reduction algorithm [Tatman and Shachter, 1990] corresponds directly to Markov value iteration. The graph rollback algorithm for stochastic trees [Hazen, 1992] also corresponds directly to value iteration. Cohort analysis and Monte Carlo simulation in Markov cycle trees, however, are based on conditional expectation in a forward manner; the complexity of these algorithms is $O(|\boldsymbol{S}| \cdot (|\boldsymbol{A}| \cdot |\boldsymbol{S}|)^N)$ since they do not make use of the memoized optimal substructures inherent in dynamic programming techniques.

Currently, four different versions of the value iteration method are implemented in DYNAMO: semi-Markov value iteration as described in (EQ 10) for solving discrimination problems and optimization problems, and Markov value iteration as described in (EQ 11) for discrimination problems and optimization problems. The discrimination versions simply ignore the maximization step involved and compute the expected value with respect to each controlled action $a \in \boldsymbol{A}$ individually.

The first clinical question for the case study is posed as a discrimination problem; the numeric parameters are assessed in accordance with an MDP, *i.e,* with constant holding times. The solution produced by the solver is a set of two policies, corresponding to the four strategies considered; each policy includes the expected value achievable for all possible starting states and all possible decision stages. We assume that the patient has a probability of 0.25 to be in atrial fibrillation, has no history of thromboembolism, and is not on warfarin at the beginning of the decision horizon. The results indicate that the strategy that administers only warfarin initially is the preferred strategy over a long-term decision horizon of 50 years or 600 months.

The second clinical question for the case study is posed as an optimization problem; the numerical parameters are assessed in accordance with an SMDP. The solution produced by the solver is an optimal policy for all possible starting states and all possible decision stages. We adopt the same assumptions about the condition of the patient as mentioned. For a short-term decision horizon of 5 years or 60 months, the results indicate that the strategy that administers only warfarin initially is the preferred strategy up to 8 months. After 8 months, if the patient remains in the same condition, the strategy that does not administer any drug initially is preferred.

7.1.2 Fundamental Matrix Solution

If the duration *N* for the decision horizon is large, the problem can be formulated with an infinite decision horizon. For infinite horizon problems with homogeneous or time-independent one-step transition functions, (EQ 10) simplifies to:

$$V_i^*(\beta) = max_a \{ \sum_j \sum_{m=1}^{\infty} q_{ij}^{(a)}(m) \left[\sum_{l=0}^{m-1} \beta^l y_i^{(a)}(l) + \beta^m b_{ij}^{(a)} + \beta^m V_j^*(\beta) \right] \}; \quad i, j \in \boldsymbol{S}; a \in \boldsymbol{A}$$

(EQ 12)

The corresponding equation for the Markov case is:

$$V_i^*(\beta) = max_a \{ v_i^{(a)}(1) + \beta \sum_j P_{ij}^{(a)} \cdot V_j^*(\beta) \}; \quad i, j \in \boldsymbol{S},\ a \in \boldsymbol{A}$$

(EQ 13)

7.1.3 Other Methods

Although not currently implemented in DYNAMO, DynaMoL admits many other solutions methods. Solution methods reported in the MDPs and SMDPs literature such as *policy iteration* [Howard, 1960], *adaptive aggregation* [Bertsekas, 1987], or *linear programming* are directly applicable if certain assumptions or conditions are met. These conditions include stationary policies, constant discount factors, homogeneous transition functions, *etc*. Some of these methods are summarized in Appendix B.

As mentioned in Section 2.5, recent research in decision-theoretic planning has also led to new solution methods for MDPs [Dean et al., 1993a] [Dean et al., 1993b] [Deardon and Boutilier, 1994] [Drummond and Bresina, 1990] [Kushmerick et al., 1994]. These methods address the trade-off between solution optimality and computation cost in complex and time-critical decision domains.

Besides the solution methods for MDPs and SMDPs, other solution methods that take advantage of the high level DynaMoL ontology can also be employed. We explore how these methods can be devised in Chapter 8.

In summary, by separating the modeling task supported by the decision grammar and graphical presentation, and the solution task supported by the mathematical representation, a large collection of solution methods can be employed in DynaMoL. Moreover, employing a new solution method does not involve any change to the high level modeling language itself; all solution methods reference only the mathematical representation of a model. Similarly, extending or adapting the decision grammar and graphical presentation do not affect the solution meth-

ods already applicable; these may, however, admit other solution methods that make use of the additional constructs.

7.2 Sensitivity Analyses

To assess the accuracy, conciseness, and clarity of a dynamic decision model formulated in DynaMoL, sensitivity analyses are often performed. Such analyses can lead to revision of the structural and numerical parameters of the model. Although not currently implemented, various model quality metrics described in Section 3.3.3 can be used in the DynaMoL framework. Adequate support for sensitivity analyses is essential for a practical dynamic decision modeling language. The current version of DynaMoL provides basic support for such analyses. Additional features can be incorporated as we extend the language and the implementation.

The graphical presentation convention and precise mathematical properties of the DynaMoL ontology facilitate analysis of the structural and numerical parameters of the model. Structural parametric analysis is done by removing or adding some event variables. The influence view supports such local structural changes. In addition to determining and analyzing the information involved in the changes, the revised models will also be translated into and solved as SMDPs to examine the changes in solution results. On the other hand, the transition view makes it easier to detect improperly specified parameters. For instance, there should not be a transition from a state where warfarin is ineligible to one where warfarin is administered. Currently there is no support for reverse translation, hence changes made to the underlying SMDP cannot be reflected in the high level components of the dynamic decision model.

Numerical parametric analysis is done by simply changing some of the transition and value functions in the SMDP and invoking the solution method. Usually no translation of the model is necessary.

In the case study, we conducted only numerical parametric analysis. The results are described in detail in Appendix C. For the long-term discrimination problem, the results indicate that the strategy of administering only warfarin but not Quinidine to the patient is the dominant strategy for all reasonable ranges of numerical parameters involved. For the short-term optimization problem, the results demonstrate that over the decision horizon, the preferred strategy shifts in a reasonable manner from initially administering warfarin to no drug, depending on the desirable and the undesirable effects of warfarin.

The mathematical representation of SMDP also facilitates analysis of the nature of a solution to a dynamic decision model. Given a specific solution method, much insights can be gained about the solution by examining the nature of the state space, action space, and other decision parameters. For instance, if absorbing

states are present in a well-formed SMDP, convergence is guaranteed in value iteration.

8 *Language Enhancement and Practical Development*

Besides addressing the trade-off between model transparency and solution efficiency, DynaMoL is a flexible framework designed to be extensible, adaptable, and practical. In Chapter 7, we have described how the current version of DynaMoL supports dynamic decision modeling with multiple perspective reasoning; all the capabilities mentioned are implemented in the DYNAMO system. In this chapter, we examine how the language features and system capabilities can be extended, adapted, and developed for general practical applications. The discussion highlights the desirable, the feasible, and the inherently difficult aspects of language enhancement and practical development of the DynaMoL framework.

8.1 Supporting Language Extension and Adaptation

Language enhancement includes extending and adapting the basic decision ontology in DynaMoL. Extension improves language expressiveness; it often involves addition or generalization of the dynamic decision grammar constructs, the graphical presentation perspectives, and the SMDP definitions. Adaptation improves language transparency or clarity; it often involves addition or reorganization of the grammar constructs and presentation perspectives, while the underlying SMDP definitions remain unchanged. This may in turn support solution methods that can make use of the rearranged representation.

DynaMoL supports incremental extension and adaptation. In general, language extension and adaptation are achieved by introducing a set of new grammar constructs or presentation formats or both, and devising a set of translators for them. This section sketches the approaches to addressing some of these issues.

8.1.1 Static vs. Dynamic Spaces

The current version of DynaMoL assumes static state space $\boldsymbol{S}$ and static action space $\boldsymbol{A}$. In a static state space, the same states are valid throughout the decision horizon. In a static action space, the same set of actions are applicable in each state over all time points or decision stages. We can model dynamic state space or action space by providing "dummy" states and "no-op" actions. Such extraneous entities, however, may compromise model clarity and solution efficiency in practice.

The DynaMoL decision grammar can be easily extended to incorporate dynamic state space and dynamic action space. New productions for the corresponding constructs can be written to incorporate the valid time or duration for each state and action. The graphical presentation convention remains mostly unchanged; only the valid entities are displayed for particular time points or decisions stages.

These new constructs directly correspond to the general classes of semi-Markov decision processes with dynamic state space and dynamic action space as defined in Section 3.3.1.

8.1.2 Automatic State Augmentation

State augmentation is a basic technique for incorporating extra information in the SMDP framework. Most of the declaratory and strategic constraints, for instance, can be represented as state attribute variables. In other words, the constraints are incorporated as additional dimensions of the state space. It is the user's responsibility to identify the relevant constraints and incorporate them into the state space directly; the associated numerical parameters are then assessed accordingly.

From a modeling point of view, constraints such as numeric restriction on some events affect only part of the original state space. For example, if one of the original state attribute variables indicates whether the patient has a stroke before, the constraint on the number of stroke only affects those states that involve a stroke. Therefore, it is more convenient to specify such constraints and their associated parameters separately, and have a set of translators to automatically incorporate the information. This involves working with an *abstract* state space, which will eventually be translated into the full state space for solution and analysis. This approach would allow us to incrementally and separately impose various constraints, without directly dealing with the resulting large and complex state space.

Automatic state augmentation can be achieved in DynaMoL by adding a set of grammar constructs for specifying the constraints, and a set of translators for translating them to form a state space of larger dimension. Similarly, additional constructs and translators are needed for the associated parameters.

This idea of modeling in an abstract state space for transparency is directly opposite to the idea of execution or solution in an abstract state space for effi-

ciency. The latter idea is adopted as *aggregation* methods in control theory [Bertsekas, 1987], and as *hierarchical* or *abstraction planning* methods in non-linear AI planning [Knoblock, 1991] [Korf, 1987] [Stefik, 1981] [Sacerdoti, 1974] [Sacerdoti, 1975] [Tate, 1977] and decision-theoretic planning [Deardon and Boutilier, 1994].

Sometimes it is desirable to freely access the multiple levels of abstraction in a dynamic decision model for both modeling and solution. Although we do not know if it is feasible in general, this may be achieved by devising a set of two-way translators in DynaMoL. This issue is left for future exploration.

8.1.3 Limited Memory

As compared to a Markov process, a semi-Markov process supports a second dimension of uncertainty: time spent in a state before transition. A semi-Markov process is only "semi-" Markov because there is a limited memory of the time duration since entry into any particular state. In some cases, memory about previous states or actions is important in a dynamic decision problem. For example, if a patient had a heart attack before, he would be more susceptible to a second heart attack during surgery. Such limited memory can be incorporated into a semi-Markov or Markov process by state augmentation, either directly or automatically as described above.

Repeated applications of state augmentation usually lead to an explosion of the state space size. To avoid such explosion, we can relegate the solution methods to keep track of limited memory. Only forward induction based solution methods, however, are applicable. In this approach, a new counter or binding called *memory* or *history* can be introduced to keep track of the relevant states that a process has visited. Calculations of the expected value with respect to the model parameters are now conditional on the history accumulated so far. In other words, the parameters are specified in terms of conditional statements. For instance, if a particular patient has a heart attack before, then the probability of his having another heart attack in the next year is 0.3, otherwise, it is 0.15.

A set of history constructs can be introduced in DynaMoL to keep track of the limited memory, and a set of translators for the conditional parametric specification are needed. The solution methods can then operate on both the SMDP and the history constructs.

8.1.4 Numeric and Ordering Constraints

Numeric and ordering constraints are closely related to the limited memory capabilities. There are several types of such constraints: Event or state numeric constraints specify that a certain number of events or states will lead to a specific consequence, *e.g.,* a patient is assumed dead if he has had three strokes. Action

numeric constraints restrict the number of times an action may be applied. Action ordering constraints specify that an action must always or never follow another action.

As in limited memory, there are two methods to incorporate such constraints into a dynamic decision model. The first method is to augment the state space to keep track of the number or the order of the decision factors. For instance, a translator for the numeric restriction constraint can be developed in the following manner: Add an additional state attribute variable for the constraint, *e.g.,* ***number of stroke***. For each event or state whose number is to be restricted, establish an influence from the corresponding event variable or state variable to the new state attribute variable. Finally, establish an influence from the new state attribute variable to every state in the next-state variable that is affected by the numeric restriction whose number is to be restricted.

The second method is to introduce a set of counters and let the solution methods worry about keeping track of the constraints. Again new constructs and translators are needed to incorporate the constraints. To facilitate proper translations, an external knowledge based system can be employed to choose the right translators by examining the nature of the constraints and solution methods.

8.1.5 Canonical Models of Combination

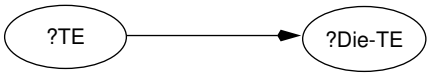
In the previous chapter, we have demonstrated how the partial-OR constraint on event combination can be incorporated in DynaMoL. Other general or domain specific constraints or canonical models can be introduced in a similar manner.

8.1.6 Presentation Convention

So far we have described two graphical presentation perspectives in DynaMoL: the transition view and the influence view. Each perspective illuminates the dynamic decision problem in a different way, and facilitates modeling at a different level of granularity. Other useful presentation perspectives can be incorporated by introducing new presentation conventions and translators.

For instance, asymmetric probabilistic relations are not structurally shown in the influence view; a tree-like perspective is more convenient for illuminating such relations. Consider the piece of knowledge: “A patient may have a chance of thromboembolization, the outcomes may be a stroke, a transient ischemic attack, or no thromboembolization. If he has a stroke, it may be permanent or transient, and there is a chance that he will die from the stroke.” Figure 8.1 shows a partial influence view that represents the above knowledge. The asymmetry involved is embedded in the conditional distribution table. If we can expand the event variables and the probabilistic relations involved into a tree, as shown in Figure 8.2,

however, the asymmetry is obvious from the tree structure. The tree structure also reflects the logical flow of represented knowledge more directly.



Conditional distribution for ?Die-TE event variable:

```
(distribution ?die-te/NoDrug)
#<Distribution-Table #:G721
    Type:                DISCRETE
    Matrix: #:G721
         "P(c|r)"                  DIE-TE           NOT-DIE-TE
       PERM-STR                    PDIE-PERM-STR    (- 1 PDIE-PERM-STR)
       TRANS-STR                   PDIE-TRANS-STR   (- 1 PDIE-TRANS-STR)
       TIA                         0.00000          1.00000
       NO-TE                       0.00000          1.00000
```

Figure 8.1 Asymmetric relations represented in an influence view.

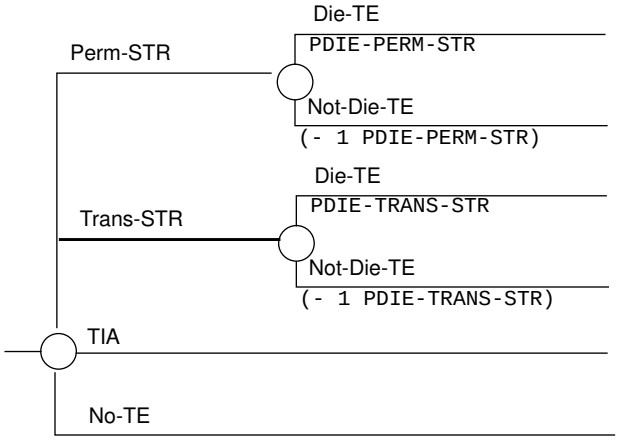


Figure 8.2 Asymmetric relations represented in a tree view.

Therefore, local expansion of the influence view into a tree view can facilitate specification of asymmetric relations; the structural granularity provided by the tree view is essential in these cases. Since there is a direct correspondence between an influence diagram and a decision tree, there is a direct correspondence between the influence view and the tree view. Writing a set of translators to incorporate the new presentation perspective in DynaMoL is straightforward.

8.2 Supporting Practical Development

Aiming to be a practical language, the DynaMoL design is modular, comprehensive, and explicit. As illustrated by the case study, the prototype DYNAMO system

can handle a class of actual dynamic decision problems in medicine. On the theoretical side, incorporating the possible language enhancement discussed above would allow the system to model and solve a larger class of problems. On the practical side, we propose a set of support tools that would make the system more convenient to use. This section summarizes the desirable support tools and sketches how these tools can be incorporated into the current DYNAMO architecture.

8.2.1 Editing and Specification Support

Structural parameters such as actions, states, and event variables can be specified in DYNAMO through either the graphical interface or Common Lisp code. On the other hand, numerical parameters, which mostly involve the conditional distributions of event variables, can only be specified through a simple table editor. Since many of the table entries are repetitive or related by probabilistic constraints, more sophisticated table editing support can be very helpful. Some examples of such support include spreadsheet editors and equation editors. These tools can be added directly to the DYNAMO system architecture.

8.2.2 Statistical Analysis

Numerical sensitivity analysis on the dynamic decision model is conducted through standard statistical techniques such as one-way or two-way regression. Such statistical capabilities can be added to DYNAMO. Alternately, external function interfaces or hooks can be incorporated to access a readily available statistical package. DYNAMO has a separately modifiable module for recording and printing a solution in different ways. The solution data can be easily piped to an external statistical module or package.

8.2.3 Data Visualization

Throughout this work, we have emphasized that information visualization is essential for facilitating dynamic decision modeling. Such visualization not only helps in debugging the model parameters, but also provides useful insights to the underlying natural phenomena. The presentation perspectives mentioned earlier illuminate both structural and numerical information in a model. Other presentation formats such as two- or three-dimensional Cartesian plots are helpful in visualizing the numerical data in different ways. The data plotters can be incorporated into the DYNAMO graphical interface in a straightforward manner.

9 Conclusion

We conclude by reflecting on the lessons learned in this research. First, we summarize the achievements and limitations of this work. Then, we compare our approaches and results with some related work. Finally, we point out some directions for future research.

9.1 Summary

We set out to compare and contrast existing methodologies for addressing a subclass of dynamic decision problems. We identified semi-Markov decision processes (SMDPs) as a common basis among the current techniques. Building on the common basis by integrating different features of these techniques, we proposed a new general methodology for dynamic decision making under uncertainty. We introduced multiple perspective reasoning and incremental language extension as the central ideas of this new methodology.

To informally evaluate the feasibility and effectiveness of the new ideas, we implemented them in a prototype system; we also conducted a detailed case study to formulate and solve two dynamic decision problems in medicine. The prototype system produced adequate solutions for the case study, and showed desirable support for general dynamic decision modeling. This exercise demonstrated the practical promise of the proposed framework.

9.1.1 The Analysis

The analysis of three major approaches to dynamic decision making under uncertainty highlighted their similarities and differences, as well as their strengths and weaknesses.

SMDPs are formal mathematical models with rigorous definitions but limited flexibility. While they can express a general class of stochastic models, they provide very little organizational support for the knowledge involved. The precisely encoded knowledge is difficult to specify and access since many assumptions are built in. The SMDP approach, therefore, is difficult to apply in practice.

Dynamic decision modeling provides graphical support for specifying and accessing the information involved. Different types of information, however, are more clearly depicted in different dynamic decision models. Moreover, the graphical model structures usually restrict the solution methods applicable.

Decision-theoretic planning techniques establish guidelines and methods to efficiently solve dynamic decision problems in large and complex domains. Unlike SMDPs and dynamic decision models, these techniques do not assume a completely enumerable state space or decision environment. There is no uniform planning framework. The underlying theoretical basis, however, is usually a subclass of SMDPs, the Markov decision processes (MDPs). Most of the insights and results are still experimental, and may not be applicable in general as SMDPs and dynamic decision models.

9.1.2 The Unifying View

Based on a uniform task definition for dynamic decision making under uncertainty, we explicated the representational and inferential support involved. We also explained how the other techniques can be formulated in terms of the SMDPs framework.

We propose that SMDPs be regarded the same role in dynamic decision making under uncertainty as first-order predicate calculus (FOPC) in deterministic knowledge representation. Recently, [Dean, 1994] has independently come up with the same idea for MDPs to be regarded as the basis for decision-theoretic planning. We have further illustrated the motivation for our proposal by devising a new methodology based on the idea.

9.1.3 The General Methodology

The new methodology adopted in DynaMoL aims to balance the trade-off between model transparency and solution efficiency. It supports information visualization and access from different angles, facilitating different aspects of problem solving. It also promotes the use of translators to enhance language ontology and differentiate the modeling task from the solution or execution task. We demonstrated how to write such translators, specifically with the development of the partial-OR gate constraint on event combination.

Model specification in DynaMoL is in terms of a higher level language than that in existing dynamic decision modeling frameworks. In frameworks such as

dynamic influence diagrams or Markov cycle trees, the model parameters need to be explicitly specified in detail. In particular, the declaratory and strategic constraints are explicitly incorporated into the graphical structure of the model; the probabilistic and temporal parameters are explicitly encoded for each time slice or period considered. The dynamic decision grammar in DynaMoL, on the other hand, supports abstract statements about the decision situation, *e.g.,* statements about how the events and states are logically related to each other. These abstract statements are analogous to macro constructs in conventional programming languages. By focusing on the decision problem ontology instead of the decision model components, DynaMoL provides a more concise and yet more transparent platform for supporting model construction.

The advantages of the graphical nature of existing dynamic decision modeling languages are preserved and extended in DynaMoL. By extending the graphical presentation convention, most model components and constraints can potentially be visualized. The different presentation perspectives further contribute to the visualization ease and clarity.

Theoretically, SMDPs can approximate most stochastic processes by state augmentation or other mechanisms. The resulting state space, however, may be too complex for direct manipulation or visualization. On the other hand, efficient solution methods may not exist for more general stochastic models. By distinguishing the specification grammar and the underlying mathematical model, DynaMoL preserves the clarity and expressiveness of the model structure, while at the same time admits a spectrum of solution methods.

While the different perspectives illuminates the model in different ways, however, loss of information may occur when the information is stored in a normal form, *e.g.,* as SMDPs. Unless all the perspective information is kept around, later retrieval of the information may be difficult. Therefore, there is an extra burden of information storage. There is also an overhead in translation time.

Moreover, the SMDP framework has some inherent limitations. Explosion of the state space size seems unavoidable when we introduce more constraints. Choosing an appropriate solution method and adapting it to handle the constraints can be a daunting task. We proposed some ideas on how such issues can be addressed.

9.1.4 The Prototype System

As illustrated with the case study, the DYNAMO system can currently handle a substantial class of dynamic decision problems; its performance is on par with existing programs. More support tools, however, are needed to enhance its utility for routine use.

We demonstrated the system capabilities with nonhomogeneous Markov and semi-Markov decision processes. The efficiency of the semi-Markov value iteration solution method needs to be improved in order to handle problems with longer decision horizons. With the advancement in computer technology and processing speed, however, we believe the method can be practically useful in many cases.

9.1.5 The Case Study

We conducted a detailed case study to informally evaluate the DynaMoL design and the DYNAMO implementation. The modeling experience involved and the solution results produced gave us confidence that the methodology would work well for a class of dynamic decision problems at least as complex as the ones in the case study.

Although not explicitly discussed, many aspects of the new methodology are useful for addressing different extensions to the case study. For instance, we could make further use of the information provided in the comprehensive policies produced. Such information can be used to predict, for example, disease evolution and treatment prognosis of a patient or a cohort of patients. Questions such as the following can be answered directly: "If the patient has a stroke 3 months later, and a bleeding episode 5 months later, and hence ineligible for warfarin therapy, what would be his expected QALE a year later?"

Working on the case study also illuminated some important aspects of practical decision modeling in a large domain in particular, and of interdisciplinary collaborative research in general. We summarize three main lessons learned from this experience:

First, a good dynamic decision modeling methodology should be both theoretically sound and practically useful. An elegant theory without a convenient interface is not going to be used to solve real problems.

Second, in system development, there should be ample communication between the language designer, the modeling expert, and in most cases, the domain expert. Each party will bring to the architecture design a different, important perspective of task requirement. Establishing effective communication can be very difficult at times due to the different technical background and proficiency of the parties involved.

Third, dynamic decision modeling is a knowledge- and labor-intensive task. Incorporating extensive domain knowledge in the system, *e.g.,* in terms a knowledge base, would facilitate the modeling process. An equally important desideratum, however, is to provide automated support for basic modeling requirements, *e.g.,* consistency checking of probability distributions.

9.2 Related Work

This work extends existing dynamic decision modeling techniques such as dynamic influence diagrams, Markov cycle trees, and stochastic trees. It also integrates many ideas in control theory and operations research for the mathematical representation and solution of semi-Markov decision processes. The notion of abstraction that lies in the core of the methodology design is a fundamental concept in computer science, and perhaps many other disciplines.

[Egar and Musen, 1993] has examined the grammar approach to specifying decision model variables and constraints. This grammar is based on the graphical structure of influence diagrams; it captures prototypical patterns at a high level abstraction for modeling trade-offs or dilemmas in clinical decision problems. The grammar, however, has not been extended to handle dynamic decision models.

The recent efforts in decision-theoretic planning have devised methods based on Markov decision processes for planning in stochastic domains; they apply the mathematical formulation directly, and do not address the ontology of general dynamic decision problems. Much can be learned from this line of work, however, both in navigating through large state spaces for finding optimal policies, and in identifying meta-level problem types for devising solution strategies.

The graphical representation of continuous-time semi-Markov processes has been explored in [Berzuini et al., 1989] and [Dean et al., 1992]; they focus only on the single-perspective presentation of relevant decision variables as Bayesian networks or influence diagrams.

On integrating dynamic decision model and SMDPs, [Provan, 1992] [Provan and Poole, 1991] [Provan and Clarke, 1993] have developed techniques that automatically construct dynamic influence diagrams from the data of underlying SMDPs, but they employ only solution algorithms for evaluating dynamic influence diagrams. The state attribute variables involved in each decision stage are represented as a Bayesian network; the overall structure corresponds to the dynamic influence diagram representation of an MDP or SMDP as discussed in Chapter 3. The parameters involved in each decision stage are tailored to the corresponding data in that stage, hence allowing only *parsimonious* information to be included over time. This approach is similar to our approach to model action effects in different influence views. Although currently the influence view structure for each action is constant over all decision stages in DynaMoL, extension to allow different views at different decision stages is straightforward. DynaMoL can also directly benefit from the structural and numerical sensitivity analysis techniques developed by Provan *et. al*.

9.3 Future Work

The future agenda of this project includes four major items. First, we want to pursue the language enhancement and practical development ideas outlined in Chapter 8. Second, we want to do an evaluation on a larger scale. Third, we want to examine how the conditional probabilities and transition functions can be automatically constructed from large database. Fourth, we want to investigate how DynaMoL can facilitate knowledge based model construction.

9.3.1 Language and System Development

Both the DynaMoL design and the DYNAMO architecture are adequate for supporting dynamic decision modeling of a general class of problems. To improve the language design and the system architecture, we will continue to pursue some of the ideas outlined in Chapter 8.

9.3.2 Large Scale Evaluation

To evaluate the capabilities of DynaMoL and to estimate the necessary language extension for handling general dynamic decision problems, we need to conduct a more comprehensive and larger scale evaluation. This can be done by building a user community that would provide feedback of the system.

9.3.3 Automatic Construction of Transition Functions

One of the most daunting task in dynamic decision modeling is to estimate and specify the numerical parameters involved. In general, given a state-space of size $|\boldsymbol{S}|$, an action-space of size $|\boldsymbol{A}|$, and a decision horizon of duration n, the number of probabilistic parameters to be assessed is of the order of $O(|A||S|^2 n)$. Subjective assessments from expert physicians may be adequate in some cases. When the decision situations are complex or the decision dimensions are large, however, the practicality of the modeling approach is limited by the lack of realistic estimations. On the other hand, given a large set of data, objective probabilities may not be easily calculated to support decision modeling; the recording formats, the measurement assumptions, and the processing errors associated with the data may complicate such derivations.

We wish to investigate the issues involved in automatic construction of one-step transition functions from databases. This exercise will provide some insights into the feasibility of such a task for supporting dynamic decision modeling. It will also illuminate some limiting constraints inherent in available databases. Hopefully, the lessons learned will contribute toward bridging the expectation gap between the dynamic decision modeler and the database builder. This will in turn encourage integration and advancement of the techniques and facilities provided

by both fields. Some preliminary insights and results from experimenting with a clinical database for insulin-dependent diabetes mellitus are shown in [Yeh, 1994] and [Yeh and Leong, 1994].

9.3.4 Supporting Knowledge Based Model Construction

Research in knowledge-based decision systems (KBDSs) combines decision analysis and artificial intelligence techniques to automate decision making. To ease the knowledge- and labor-intensive decision analysis process, knowledge-based techniques are employed to automate various decision model manipulations.

KBDSs employing knowledge-based model construction (KBMC), *e.g.,* ALTERID [Breese, 1992], FRAIL [Goldman and Charniak, 1990], SUDO-PLANNER [Wellman, 1990], and DYNASTY [Provan, 1992], advocate that the decision models for different problems should be constructed on demand from a knowledge base [Wellman et al., 1992]. While the decision models are governed by strict decision-theoretic semantics, the knowledge base captures more general facts and relations.

Each existing KBMC system synthesizes only one type of decision models, *e.g.,* influence diagrams. In dynamic decision models, the time-dependent constraints are usually translated into numbers, equations, or complicated substructures hard-wired into specific decision models; subsequent retrieval of the constraints is quite difficult, if not impossible, from these models.

We want to explore if the high level decision ontology in DynaMoL can facilitate KBMC. We believe DynaMoL provides a more expressive and explicit language for formulating dynamic decision problems. This will relieve the knowledge base ontology and organization from being restricted by the graphical structure of the target model. Moreover, the resulting model will support more detailed analysis and more efficient solution methods as argued before.

A Dynamic Decision Grammar

The dynamic decision grammar for Version 1.0 of DynaMoL is shown below.

Model	→	*name: Identifier;* *contexts: Context-list;* *definitions: Definition-list;* *constraints:Constraint-list;* *solution: Optimality-policy*
Identifier	→	*name:* ***String***
Context-list	→	*Context**
Context	→	*value:* ***String***
Definition-list	→	*time: Time-horizon;* *actions: Action-space;* *states: State-space;* *event-variables: Event-variable-space;* *transitions: Transition-table;* *influences: Influence-table;* *value-functions: Value-function-table*
Time-horizon	→	*time-duration: Time-duration;* *time-unit: Time-unit*
Time-duration	→	$\Im \mid \infty$
Time-unit	→	*value:* ***String***
Action-space	→	$Action^{+}$

Action	→	*name: Identifier;* *state: State;* *time: Time*
Time	→	$\Im$
State-space	→	*State*$^{+}$
State	→	*name: Identifier;* *value-function: Value-function;* *time: Time*
Event-variable-space	→	*Event-variable**
Event-variable	→	*name: Identifier;* *outcomes: Event-list;* *probabilities: Probability-list;* *time: Time*
Event-list	→	*Event*$^{+}$
Event	→	*name: Identifier*
Probabilitiy-list		*Probability*$^{+}$
Probability	→	*value:* ***[0,1]***
Transition-table	→	*Transition**
Transition	→	*transition-probability: Probability;* *holding-times-distribution: Distribution-function;* *time: Time*
Distribtuion-function	→	***Cumulative-distribution-function*** | ***Probability-mass-function***
Influence-table	→	*prob-influence-list: Prob-influence-list;* *info-influence-list: Info-influence-list*
Prob-influence-list	→	*Probabilistic-influence**
Probabilistic-influence	→	*from: Identifier;* *to: Identifier;* *cond-prob-table: Cond-prob-table;* *time: Time*
Cond-prob-table	→	*outcome-labels: Label-list;* *cond-outcome-labels: Label-list;* *cond-prob-list: Cond-prob-list*
Label-list	→	*Identifier*$^{+}$
Cond-prob-list	→	*Probability**
Info-influence-list	→	*Informational-influence**

Informational-influence	→	*from: Identifier;* *to: Identifier;* *time: Time*
Value-function-table	→	*Value-function**
Value-function	→	*value:* ***Function***
Constraint-list	→	*d-constraints:Declaratory-constraint-list;* *s-constraints: Strategic-constraint-list*
Declaratory-constraint-list	→	*Declaratory-constraint**
Declaratory-constraint	→	*name: Identifier;* *states: State-list;* *events: Event-list;* *relation:* ***Relation***
State-list	→	$State^{+}$
Strategic-constraint-list	→	*Strategic-constraint**
Strategic-constraint	→	*name: Identifier;* *states: State-list;* *actions: Action-list;* *relation:* ***Relation***
Action -list	→	$Action^{+}$
Optimal-policy	→	$Decision\text{-}rule^{+}$
Decision-rule	→	*state: Identifier;* *action: Identifier;* *time: Time;* *expected-value:* $\Re$

B Semi-Markov Decision Process: Definitions and Techniques

This appendix summarizes some common concepts and computational methods in semi-Markov decision processes. The descriptions below extend the definitions presented in Chapters 2points. The main references for this summary are [Heyman and Sobel, 1984], [Howard, 1971], and [Puterman, 1990].

B.1 Components of a Semi-Markov Decision Process

A semi-Markov decision process SMDP is defined in terms of a time index set $\boldsymbol{T}$, a semi-Markov reward process $\{S(t)\,;\ t \in \boldsymbol{T}\}$ with respect to the state space $\boldsymbol{S}$, a decision process $\{D(t)\,;\ t \in \boldsymbol{T}\}$ with respect to the action space $\boldsymbol{A}$, a set of one-step transition functions, and a set of value functions.

B.1.1 Time Index Set

The time index set $\boldsymbol{T}$ can be finite or infinite, and its elements can be discrete or continuous. In this work, we only consider discrete time indices, *i.e.*, $\boldsymbol{T} = \{0, 1, 2, 3, \ldots\}$. The transition epochs $T_k \in \boldsymbol{T}$; $k = 1, 2, 3, \ldots$, indicate when transitions actually occur in the time horizon.

B.1.2 State Space

The state space $\boldsymbol{S}$ specifies the set of states that may occur for all time points $t \in \boldsymbol{T}$. In other words:

$$\boldsymbol{S} = \bigcup_{t \in \boldsymbol{T}} \boldsymbol{S}_t$$

where $\boldsymbol{S}_t$ is the set of states that may occur at time $t \in \boldsymbol{T}$. A static state space $\boldsymbol{S}$ contains only one set of states that are time-independent or constant for all time points $t \in \boldsymbol{T}$.

B.1.3 Action Space

The action space $\boldsymbol{A}$ specifies the set of actions that are valid for every state for all time points $t \in \boldsymbol{T}$. In other words:

$$\boldsymbol{A} = \bigcup_{t \in \boldsymbol{T}} \boldsymbol{A}_{S_t}$$

where

$$\boldsymbol{A}_{\boldsymbol{S}_t} = \bigcup_{t \in \boldsymbol{T},\, s \in \boldsymbol{S}} \boldsymbol{A}_{s,\,t}$$

such that $\boldsymbol{A}_{s,\,t}$ is the set of actions valid for state s at time $t \in \boldsymbol{T}$. A static action space $\boldsymbol{A}$ contains only one set of actions that are state-independent and time-independent; the actions are applicable for all states $s \in \boldsymbol{S}$, for all time points $t \in \boldsymbol{T}$.

B.1.4 One-Step Transition Functions

The one-step transition mass function $q_{ij}^{(a)}(m, t)$ is the joint probability that, given action *a*, a process enters state *i* at time *t* will make its transition to state *j* at duration *m*. In other words:

$$\begin{aligned} & q_{ij}^{(a)}(m, t) \\ & \quad = P\{S_{k+1} = j, T_{k+1} - T_k = m \mid S_k = i, D_k = a, T_k = t\} \\ & \quad = P\{S(T_{k+1}) = j, T_{k+1} - T_k = m \mid S(T_k) = i, D(T_k) = a, T_k = t\} \end{aligned}$$

$q_{ij}^{(a)}(m, t)$ is subject to the following conditions:

i. $\sum_j \sum_{m=0}^{\infty} q_{ij}^{(a)}(m, t) = 1;\ \forall t \in \boldsymbol{T}$

ii. $q_{ij}^{(a)}(0, t) = 0;\ \forall t \in \boldsymbol{T}$

iii. $q_{ij}^{(a)}(m, t) \geq 0;\ \forall m \geq 0, \forall t \in \boldsymbol{T}$

The one-step transition distribution function $Q_{ij}^{(a)}(m, t)$ is the joint probability that, given action *a*, a process enters state *i* at time *t* will make its transition to state *j* by duration *m*. In other words:

$$Q_{ij}^{(a)}(m,t)$$
$$= P\{S_{k+1} = j, T_{k+1} - T_k \le m \mid S_k = i, D_k = a, T_k = t\}$$
$$= P\{S(T_{k+1}) = j, T_{k+1} - T_k \le m \mid S(T_k) = i, D(T_k) = a, T_k = t\}$$

$Q_{ij}^{(a)}(m,t)$ is subject to the following conditions:

i. $\sum_j \lim_{m \to \infty} Q_{ij}^{(a)}(m,t) = 1;\ \forall t \in \boldsymbol{T}$

ii. $\lim_{m \to 0} Q_{ij}^{(a)}(m,t) = 0;\ \forall t \in \boldsymbol{T}$

iii. $Q_{ij}^{(a)}(m,t) \ge 0;\ \forall m \ge 0, \forall t \in \boldsymbol{T}$

B.1.5 Alternate Definitions Of One-Step Transition Functions

The one-step transition function as defined above is a joint probability function. This function quantifies two dimensions of uncertainty in a state transition: the transition destination and the time duration spent in the current state before the transition. There are two ways to assess the one-step transition functions: from transition probabilities and holding time functions, or from conditional transition probabilities and waiting time functions. We examine the alternate definitions and the corresponding methods in this section.

Transition Probabilities

The transition probability $P_{ij}^{(a)}(t)$ is the probability that, given action a, a process that enters the current state i at time t will eventually make the next transition to state j.

$$P_{ij}^{(a)}(t) = P\{S_{m+1} = j \mid S_m = i, D_m = a, T_m = t\}$$

which also satisfies the *Markovian property, i.e.,* it does not depend on how the process gets to the current state i:

$$P_{ij}^{(a)}(t) = P\{S_{k+1} = j \mid S_k = i, D_k = a, T_k = t\}$$
$$= P\{S_{k+1} = j \mid S_k = i, S_{k-1} = h, \ldots, D_k = a, T_k = t\}$$

$P_{ij}^{(a)}(t)$ is subject to the following conditions:

i. $\sum_j P_{ij}^{(a)}(t) = 1;\ \forall t \in \boldsymbol{T}$

ii. $P_{ij}^{(a)}(t) \ge 0;\ \forall t \in \boldsymbol{T}$

Conditional Transition Probabilities

The conditional transition probability $p_{ij}^{(a)}(m, t)$ is the probability that, given action a, a process that enters the current state i at time t and makes a transition at duration m will make that transition to state j.

$$p_{ij}^{(a)}(m, t) = P\{S_{k+1} = j \mid S_k = i, D_k = a, T_k = t, T_{k+1} - T_k = m\}$$

which also satisfies the *Markovian property*:

$$\begin{aligned} & p_{ij}^{(a)}(m, t) \\ & = P\{S_{k+1} = j \mid S_k = i, D_k = a, T_k = t, T_{k+1} - T_k = m\} \\ & = P\{S_{k+1} = j \mid S_k = i, S_{k-1} = h, \ldots, D_k = a, T_k = t, T_{k+1} - T_k = m\} \end{aligned}$$

$p_{ij}^{(a)}(m, t)$ is subject to the following conditions:

i. $\sum_j p_{ij}^{(a)}(m, t) = 1;\ \forall t \in \boldsymbol{T}$

ii. $p_{ij}^{(a)}(m, t) \geq 0;\ \forall t \in \boldsymbol{T}$

iii. $p_{ij}^{(a)}(0, t) = 0;\ \forall t \in \boldsymbol{T}$

Holding Time Mass and Cumulative Distribution Functions

The holding time $\tau_{ij}^{(a)}(t)$ is a random variable denoting the time duration that, given action a, a process which enters the current state i at time t will stay in i, if its next transition is to state j. Assume that $E[\tau_{ij}^{(a)}(t)]$ is finite, $\forall a \in \boldsymbol{A},\ \forall i, j \in \boldsymbol{S},\ \forall t \in \boldsymbol{T}$.

The probability mass function (PMF) for $\tau_{ij}^{(a)}(t)$ is defined as follows:

$$\begin{aligned} h_{ij}^{(a)}(m, t) &= P\{\tau_{ij}^{(a)}(t) = m\} \\ &= P\{T_{k+1} - T_k = m \mid S_k = i, S_{k+1} = j, D_k = a, T_k = t\} \end{aligned}$$

$h_{ij}^{(a)}(m, t)$ is subject to the following conditions:

i. $m = 0, 1, 2, 3, \ldots$

ii. $h_{ij}^{(a)}(0, t) = 0; \forall t \in \boldsymbol{T}$

iii. $\sum_{m=0}^{\infty} h_{ij}^{(a)}(m, t) = 1;\ \forall t \in \boldsymbol{T}$

iv. $h_{ij}^{(a)}(m, t) \geq 0;\ \forall m \geq 0, \forall t \in \boldsymbol{T}$

The cumulative distribution function (CDF) for $\tau_{ij}^{(a)}(t)$ is defined as follows:

$$\begin{aligned} H_{ij}^{(a)}(m,t) &= P\{\tau_{ij}^{(a)}(t) \le m\} \\ &= P\{T_{k+1} - T_k \le m \mid S_k = i, S_{k+1} = j, D_k = a, T_k = t\} \end{aligned}$$

$H_{ij}^{(a)}(m,t)$ is subject to the following conditions (as all CDFs):

i. $H_{ij}^{(a)}(m,t)$ is monotonically increasing and $0 \le H_{ij}^{(a)}(m,t) \le 1;\ \forall t \in \boldsymbol{T}$.

ii. $H_{ij}^{(a)}(m,t)$ is right continuous:

$$\lim_{\varepsilon \to 0} H_{ij}^{(a)}(m+\varepsilon, t) = H_{ij}^{(a)}(m,t);\ \forall t \in \boldsymbol{T}$$

iii. $\lim_{m \to 0} H_{ij}^{(a)}(m,t) = 0$ and $\lim_{m \to \infty} H_{ij}^{(a)}(m,t) = 1;\ \forall t \in \boldsymbol{T}$

Waiting Time Mass and Cumulative Distribution Functions

The waiting time or unconditional holding time $\tau_i^{(a)}(t)$ is a random variable denoting the time duration that, given action a, a process which enters the current state i at time t will stay in i before its next transition.

The PMF for $\tau_i^{(a)}$ is defined as follows:

$$\begin{aligned} w_i^{(a)}(m,t) &= P\{\tau_i^{(a)}(t) = m\} = \sum_j P_{ij}^{(a)}(t)\, h_{ij}^{(a)}(m,t) \\ &= P\{T_{k+1} - T_k = m \mid S_k = i, D_k = a, T_k = t\} \end{aligned}$$

$w_i^{(a)}(m,t)$ is subject to the following conditions:

i. $m = 0, 1, 2, 3, \ldots$

ii. $w_i^{(a)}(0,t) = 0;\ \forall t \in \boldsymbol{T}$

iii. $\sum_{m=0}^{\infty} w_i^{(a)}(m,t) = 1;\ \forall t \in \boldsymbol{T}$

iv. $w_i^{(a)}(m,t) \ge 0;\ \forall m \ge 0, \forall t \in \boldsymbol{T}$

The CDF for $\tau_i^{(a)}(t)$ is defined as follows:

$$\begin{aligned} W_i^{(a)}(m,t) &= P\{\tau_i^{(a)}(t) \le m\} = \sum_j P_{ij}^{(a)}(t)\, H_{ij}^{(a)}(m,t) \\ &= P\{T_{k+1} - T_k \le m \mid S_k = i, D_k = a, T_k = t\} \end{aligned}$$

$W_i^{(a)}(m,t)$ is subject to the following conditions (as all CDFs):

i. $W_i^{(a)}(m,t)$ is monotonically increasing and $0 \le W_i^{(a)}(m,t) \le 1;\ \forall t \in \boldsymbol{T}$.

ii. $W_i^{(a)}(m,t)$ is right continuous, *i.e.*,

$$\lim_{\varepsilon \to 0} W_i^{(a)}(m+\varepsilon,t) = W_i^{(a)}(m,t);\ \forall t \in \boldsymbol{T}$$

iii. $\lim_{m \to 0} W_i^{(a)}(m,t) = 0$ and $\lim_{m \to \infty} W_i^{(a)}(m,t) = 1;\ \forall t \in \boldsymbol{T}$

Calculating one-step transition functions: Method 1

The first method to calculate the one-step transition function is to select the transition destination first, then decide on the holding time.

For the PMF:

$$q_{ij}^{(a)}(m,t) = P_{ij}^{(a)}(t)\, h_{ij}^{(a)}(m,t);$$
$$a \in \boldsymbol{A}, i,j \in \boldsymbol{S}, t \in \boldsymbol{T}, m \ge 0$$

$q_{ij}^{(a)}(m,t)$ is subject to the following conditions:

i. $\sum_j q_{ij}^{(a)}(m,t) = \sum_j P_{ij}^{(a)}(t)\, h_{ij}^{(a)}(m,t) = w_i^{(a)}(m,t)$

ii. $\sum_{m=0}^{\infty} q_{ij}^{(a)}(m,t) = \sum_{m=0}^{\infty} P_{ij}^{(a)}(t)\, h_{ij}^{(a)}(m,t) = P_{ij}^{(a)}(t)$

For the CDF:

$$Q_{ij}^{(a)}(m,t) = P_{ij}^{(a)}(t)\, H_{ij}^{(a)}(m,t);$$
$$a \in \boldsymbol{A}, i,j \in \boldsymbol{S}, t \in \boldsymbol{T}, m \ge 0$$

$Q_{ij}^{(a)}(m,t)$ is subject to the following conditions:

i. $\sum_j Q_{ij}^{(a)}(m,t) = \sum_j P_{ij}^{(a)}(t)\, H_{ij}^{(a)}(m,t) = W_i^{(a)}(m,t)$

ii. $\lim_{m \to \infty} Q_{ij}^{(a)}(m,t) = \lim_{m \to \infty} P_{ij}^{(a)}(t)\, H_{ij}^{(a)}(m,t) = P_{ij}^{(a)}(t)$

Calculating one-step transition-functions: Method 2

The second method to calculate the one-step transition function is to select the waiting time first, then decide on the transition destination.

For the PMF:

$$q_{ij}^{(a)}(m,t) = p_{ij}^{(a)}(m,t)\, w_i^{(a)}(m,t);$$
$$a \in \boldsymbol{A}, i, j \in \boldsymbol{S}, t \in \boldsymbol{T}, m \geq 0$$

For the CDF:

$$Q_{ij}^{(a)}(m,t) = p_{ij}^{(a)}(m,t)\, W_i^{(a)}(m,t);$$
$$a \in \boldsymbol{A}, i, j \in \boldsymbol{S}, t \in \boldsymbol{T}, m \geq 0$$

$q_{ij}^{(a)}(m,t)$ and $Q_{ij}^{(a)}(m,t)$ are subject to the same conditions as defined earlier.

Relationships among the components of one-step transition functions

From the definitions above, we have:

$$P_{ij}^{(a)}(t)\, h_{ij}^{(a)}(m,t) = p_{ij}^{(a)}(m,t)\, w_i^{(a)}(m,t)$$

Therefore, to find:

$$p_{ij}^{(a)}(m,t) = \frac{P_{ij}^{(a)}(t)\, h_{ij}^{(a)}(m,t)}{w_i^{(a)}(m,t)} = \frac{P_{ij}^{(a)}(t)\, h_{ij}^{(a)}(m,t)}{\sum_j P_{ij}^{(a)}(t)\, h_{ij}^{(a)}(m,t)}$$

$$P_{ij}^{(a)}(t) = \sum_{m=0}^{\infty} p_{ij}^{(a)}(m,t)\, w_i^{(a)}(m,t)$$

$$h_{ij}^{(a)}(m,t) = \frac{p_{ij}^{(a)}(m,t)\, w_i^{(a)}(m,t)}{P_{ij}^{(a)}(t)}$$

The same calculations can be carried out with the PMFs for holding times and waiting times substituted by the CDFs.

Special Case 1: Independent Semi-Markov Decision Processes

In the special cases involving *independent* SMDPs, knowledge of when a transition occurs does not affect assignment of the transition destination. In other words:

$$h_{ij}^{(a)}(m,t) = w_i^{(a)}(m,t) \text{ or } p_{ij}^{(a)}(m,t) = P_{ij}^{(a)}(t).$$

Special Case 2: Real versus Virtual Transitions

Virtual transitions are self-transitions that may not be observable. Transformation of an SMDP with both real and virtual transitions into one with only real or observable transitions is as follows:

$$q'^{(a)}_{ij}(m,t) = \begin{cases} \dfrac{q^{(a)}_{ij}(m,t)}{1-q^{(a)}_{ii}(m,t)} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \quad ;\forall m \geq 0,\ \forall t \in \boldsymbol{T}$$

In practice, if the one-step transition functions are calculated from either methods described earlier, the transformation involves transforming the transition probabilities or conditional transition probabilities, and then assessing the holding time functions or waiting time functions accordingly.

Special Case 3: Geometric Holding Time Functions for Markov Decision Processes

Assuming the independence condition as defined above, a Markov process with virtual or self transitions and constant holding time functions can be transformed to one with no virtual transitions and geometric holding time mass or distribution functions as follows:

$$\begin{aligned} \sum_j P^{(a)}_{ij}(t)\, h^{(a)}_{ij}(m,t) &= \sum_j P^{(a)}_{ij}(t)\, w^{(a)}_i(m,t) = w^{(a)}_i(m,t) \\ &= (1-P^{(a)}_{ii}(t))\,(P^{(a)}_{ii})^{m-1} \end{aligned}$$

The definition indicates that the probability for a waiting time of m time units implies that all transitions up to m-1 time units are virtual or self transitions.

B.1.6 Value Functions

The value function $v^{(a)}_i(m)$ determines the objective values achievable by a process in state i, given action a, over time duration m. $v^{(a)}_i(m)$ is defined in terms of the transition value function $v^{(a)}_{ij}(m)$ which determines the objective values achievable by a process in state i, given action a, over time duration m if its next transition is to state j.

$$v^{(a)}_i(m) = \sum_j q(.)\, v^{(a)}_{ij}(m)$$

The transition value function $v^{(a)}_{ij}(m)$ is in turn defined in terms of a transition yield function $y^{(a)}_{ij}(l)$ and a transition bonus function $b^{(a)}_{ij}(m)$.

The transition yield function determines the objective values achievable, after entering state i, at time l for occupying state i during the time interval $(0, l]$, given that the action is a and the next state is j.

The transition bonus function determines the objective values achievable at time of transition from state i to state j, given action a and after a holding time duration of m in state i.

B.2 Solutions of a Semi-Markov Decision Process

The solution of an SMDP is an optimal policy π^*; an optimal policy is a set of decision rules μ_t^* across the decision horizon **T**. In other words: $\pi^* = \{\mu_0^*, \mu_1^*, \mu_2^*, \ldots\}$

The decision rule $\mu_t^* : \boldsymbol{S} \rightarrow \boldsymbol{A}$ prescribes the optimal action $a \in \boldsymbol{A}$ for every state $s \in \boldsymbol{S}$ at time $t \in \boldsymbol{T}$. A stationary decision rule $\mu_t^{a^*} : \boldsymbol{S} \rightarrow a$ is one in which the optimal action is the same for all the states for each time index $t \in \boldsymbol{T}$. A stationary optimal policy $\pi^* = \{\mu^*, \mu^*, \mu^*, \ldots\}$ is one in which the decision rules are the same for all time indices $t \in \boldsymbol{T}$.

Value Iteration

The value iteration method is detailed in Chapter 7.

Policy Iteration

For infinite horizon problems, assuming optimal stationary policies, the policy iteration method is usually more efficient than brute-force value iteration. The algorithm for this technique on MDPs is briefly summarized below [Bertsekas, 1987]:

1. (Initialization) Guess an initial stationary policy $\pi^0 = \{\mu^0, \mu^0, \mu^0, \ldots\}$;
2. (Policy evaluation) Given the stationary policy $\pi^m = \{\mu^m, \mu^m, \mu^m, \ldots\}$, compute the corresponding value function $J_i^{\mu^m}$ from the linear system of equations, which follows from the stationary property of the policy:

$$J_i^{\mu^m} = v_i^{\mu^m(i)} + \beta \sum_j P_{ij}^{\mu^m(i)} J_j^{\mu^m}; \ \forall i, j \in \boldsymbol{S}$$

3. (Policy improvement) Obtain a new stationary policy $\pi^{m+1} = \{\mu^{m+1}, \mu^{m+1}, \mu^{m+1}, \ldots\}$ satisfying:

$$v_i^{\mu^{m+1}(i)} + \beta \sum_j P_{ij}^{\mu^{m+1}(i)} J_j^{\mu^m} = max_a \{ v_i^{(a)} + \beta \sum_j P_{ij}^{(a)} J_j^{\mu^m} \}; \ \forall a \in \boldsymbol{A}, i \in \boldsymbol{S}$$

   If $J_i^{\mu^m} = max_a \{ v_i^{(a)} + \beta \sum_j P_{ij}^{(a)} J_j^{\mu^m} \}; \ \forall a \in \boldsymbol{A}, i, j \in \boldsymbol{S}$,

   stop and declare $\pi^m = \{\mu^m, \mu^m, \mu^m, \ldots\}$ the optimal policy; otherwise go to step 2 and repeat with the improved policy $\pi^{m+1} = \{\mu^{m+1}, \mu^{m+1}, \mu^{m+1}, \ldots\}$.

Since the decision space and the state space are finite, the collection of all stationary policies is also finite. Moreover, at each iteration an improved policy is obtained. It follows that unlike value iteration, the policy iteration algorithm will

finitely terminate. The overhead for this method, however, could still be very high if the state space size is large.

Adaptive Aggregation

To ease the overhead imposed by large state spaces in the policy iteration method, an approximation technique called adaptive aggregation can be employed. The key idea of this method, which will not be elaborated here, is to solve a system of equations with smaller dimension; this can be achieved by lumping together the original states in $\boldsymbol{S}$ into a smaller set of aggregate states. This is assuming that the expected value achievable in all the original states in any aggregate state differs from that achievable in the aggregated form by a constant factor.

More than one aggregation may be carried out in this method, and the aggregate states can change from one iteration to the next. Each aggregation step can also be used together with value iteration for a finite number of steps to further improve the efficiency [Bertsekas, 1987].

Linear Programming

The linear programming method is based on the characteristics of the transition matrix P of the underlying semi-Markov process. In particular, let the ordered eigenvalues of the transition matrix be λ_j, $1 \leq j \leq |\boldsymbol{S}|$, such that $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_{|\boldsymbol{S}|}|$ with $\lambda_1 = 1$. For MDPs, it follows after some manipulations on the optimality equation that the optimal policy corresponding to J_i^*, $\forall i \in \boldsymbol{S}$ solves the following minimization problem:

$$min \sum_{i=1}^{|S|} \lambda_i$$

subject to

$$\lambda_i \geq v_i^{(a)} + \beta \sum_j P_{ij}^{(a)} \lambda_j \;;\; \forall i, j \in \boldsymbol{S}, a \in \boldsymbol{A}$$

The rationale behind the method is detailed in [Bertsekas, 1987]. We shall only note here that polynomial time algorithms, with respect to the number of states, exist for solving a system of linear equations. Although the method becomes unwieldy when the decision space and state space sizes get large, to the order of hundreds, say, we believe it would work reasonably well for a subclass of problems in the medical domain considered in this work.

C Dynamic Decision Model for the Case Study

This chapter details the data and results for the case study. The data are specified through either the DYNAMO graphical interface or Common Lisp code.

C.1 The Action Space

There are two controlled actions: No drug and Quinidine, and two embedded actions: warfarin on and warfarin off, corresponding to the following four strategies:

1. Strategy "None": Start without any drug, with warfarin administered and stopped as necessary;
2. Strategy "Warfarin": Start with warfarin, which can be stopped when necessary;
3. Strategy "Quinidine": Start with Quinidine, with warfarin administered and stopped as necessary; and
4. Strategy "Both": Start with both Quinidine and warfarin, with warfarin stopped when necessary.

C.2 The State Space

The state space contains 19 states with associated value functions as described in Table C.1. The keys to the table are as follows:

Keys to state descriptions

NSR	=	Normal sinus rhythm
AF	=	Atrial fibrillation
N	=	No thromboembolization
TIA	=	Transient isthmic attack

STR = Stroke
WOFF = Warfarin off
WON = Warfarin on
WNE = Warfarin not eligible

Keys to yield rates

UWarf = 0.9800
UStroke = 0.5000

Table C.1 The states and corresponding value functions.

State i	Yield rate y_i	Initial value $V_i(0)$
NSR-N-WOFF	1.0000	-0.5000
NSR-N-WON	UWarf	-0.5000*UWarf
NSR-N-WNE	1.0000	-0.5000
NSR-TIA-WOFF	1.0000	-0.5000
NSR-TIA-WON	UWarf	-0.5000*UWarf
NSR-TIA-WNE	1.0000	-0.5000
NSR-STR-WOFF	UStroke	-0.5000*UStroke
NSR-STR-WON	UStroke*UWarf	-0.5000*UStroke*UWarf
NSR-STR-WNE	UStroke	-0.5000*UStroke
AF-N-WOFF	1.0000	-0.5000
AF-N-WON	UWarf	-0.5000*UWarf
AF-N-WNE	1.0000	-0.5000
AF-TIA-WOFF	1.0000	-0.5000
AF-TIA-WON	UWarf	-0.5000*UWarf
AF-TIA-WNE	1.0000	-0.5000
AF-STR-WOFF	UStroke	-0.5000*UStroke
AF-STR-WON	UStroke*UWarf	-0.5000*UStroke*UWarf
AF-STR-WNE	UStroke	-0.5000*Ustroke
DEAD	0.0000	0.0000

C.3 Numerical Variables

The numerical variables defined for computing the probability distributions are described in the following Common Lisp file. The yearly rates *r* shown in the table are converted to monthly probabilities *P* according to (EQ 6), as reproduced below:

$$P = 1 - e^{-r\tau};\ \tau = 1/12 \qquad \textbf{(EQ 6)}$$

```
;;; -*- Mode: LISP; Syntax: Common-Lisp; Package: DYNAMO-INTERFACE; Base: 10 -*-

;;; ***************************************************************************
;;; CASE-STUDY
;;;
;;;
;;; Quinidine Case Study Data File
;;; Creation date: April 19, 1994
;;;
;;; Copyright (c) 1994, Tze-Yun Leong, Massachusetts Institute of Technology
;;; ***************************************************************************

(in-package :dynamo-interface)

;;; The life table.
(read-life-table "/u/leong/dynamo/table/mmale.tab2")

;;; All the probabilities are monthly probabilities. All the rates are yearly
;;;  rates. Appropriate conversion to monthly probabilities are needed.

;;; The Variable Bindings.

(defvar IIM (/ 1 12.0))     ; The time duration or interval in month
(defvar StartAge 51.0000)   ; Initial age of patient
(defvar AFinit 0.2500)      ; Initial probability of being in AF

(defvar pdiestroke 0.1000)  ; Probability of dying of stroke
(defvar pdiechm 0.6200)     ; Probability of dying of cerebral hemorrhage
(defvar pdiegib 0.1000)     ; Probability of dying of gastro-intestinal bleeding
(defvar pperm 0.4800)       ; Probability of a stroke being permanent
(defvar pstroke 0.8800)     ; Probability of developing a stroke

(defvar muquin 0.0120)      ; Excess mortality rate of Quinidine
(defvar rCerehem 0.0006)    ; Rate of developing cerebral hemorrhage
(defvar rEmb 0.0093)        ; Rate of developing thromboembolism
(defvar rGib 0.0060)        ; Rate of developing gastro-intestinal bleeding

(defvar STRfact 2.0000)     ; Relative risk of stroke on thromboembolism
(defvar rrEmbAF 5.0000)     ; Relative risk of AF on thromboembolism
(defvar rrwarf 0.3000)      ; Relative risk of warfirin on thromboembolism
(defvar rrbleed 2.5000)     ; Relative risk of warfarin on bleeding
(defvar rrquin 0.6000)      ; Relative risk of Quinidine on AF

(defvar stmGIB 0.2500)      ; Short term morbidity of gastro-intestinal bleeding
(defvar stmSTROKE 0.7500)   ; Short term morbidity of stroke
(defvar stmTIA 0.2500)      ; Short term morbidity of transient ischemic attack

;;; The Variable Bindings dependent on the Action

;;; Time-dependent probability of dying and its complement.
(defvar pmusum '(prob (read-from-table (+ StartAGE (* time iim))) iim))
(defvar pmusum-comp (selective-sub (list 1 pmusum)))

;;; Time-depedent probability of dying given Quinidine and its complement.
(defvar pmusum-quin
        '(prob (+ (read-from-table (+ StartAGE (* time iim))) muquin) iim))
(defvar pmusum-quin-comp (selective-sub (list 1 pmusum-quin)))

(defvar pcerehem (prob rcerehem iim))  ; Probability of cerebral hemorrhage
```

```
;;; Probability of developing cerebral hemorrhage given warfarin and its complement
;;; Markov case: assume constant rrbleed
(defvar pcerehem-warf (prob (* rcerehem rrbleed) iim))
(defvar pcerehem-warf-comp PCEREHEM-WARF-COMP)
;;; Semi-Markov case: assume varying rrbleed
(defvar pcerehem-warf
        '(prob (* rcerehem (+ 2.5 (* 22.5 (exp (- 0.4 duration))))) iim))
(defvar pcerehem-warf-comp (
        '(- 1.0 (prob (* rcerehem (+ 2.5 (* 22.5 (exp (- 0.4 duration))))) iim)))

(defvar pgib (prob rGib iim)); Probability of developing gastro-intestinal bleeding
;;; Probability of gastro-intestinal bleeding given warfarin and its complement
;;; Markov case: assume constant rrbleed
(defvar pgib-warf (prob (* rGib rrbleed) iim))
(defvar pgib-warf-comp (- 1.0 pgib-warf)
;;; Semi-Markov case: assume varying rrbleed
(defvar pgib-warf
        '(prob (* rgib (+ 2.5 (* 22.5 (exp (- 0.4 duration))))) iim))
(defvar pgib-warf-comp
        '(- 1.0 (prob (* rgib (+ 2.5 (* 22.5 (exp (- 0.4 duration))))) iim)))

;;; The Variable Bindings dependent on both the Action and current State.

;;; Probability of thromboembolism given various factors: AF, stroke, warfarin
(defvar pemb (prob remb iim))
(defvar pemb-af (prob (* remb rrEmbAf) iim))
(defvar pemb-warf (prob (* remb rrwarf) iim))
(defvar pemb-af-warf (prob (* remb rrembAF rrwarf) iim))
(defvar pemb-str (prob (* remb strfact) iim))
(defvar pemb-af-str (prob (* remb rrembAF strfact) iim))
(defvar pemb-str-warf (prob (* remb strfact rrwarf) iim))
(defvar pemb-all (prob (* remb rrembAF strfact rrwarf) iim))

;;; Probability of specific thromboembolization: permanent stroke, transient
;;; stroke, transient ischemic attack, given various factors: AF, stroke, warfarin.
(defvar pperm-str (* pemb pstroke pperm))
(defvar ptrans-str (* pemb pstroke (- 1 pperm)))
(defvar ptia (* pemb (- 1 pstroke)))

(defvar pperm-str-a (* pemb-af pstroke pperm))
(defvar ptrans-str-a (* pemb-af pstroke (- 1 pperm)))
(defvar ptia-a (* pemb-af (- 1 pstroke)))

(defvar pperm-str-w (* pemb-warf pstroke pperm))
(defvar ptrans-str-w (* pemb-warf pstroke (- 1 pperm)))
(defvar ptia-w (* pemb-warf (- 1 pstroke)))

(defvar pperm-str-aw (* pemb-af-warf pstroke pperm))
(defvar ptrans-str-aw (* pemb-af-warf pstroke (- 1 pperm)))
(defvar ptia-aw (* pemb-af-warf (- 1 pstroke)))

(defvar pperm-str-s (* pemb-str pstroke pperm))
(defvar ptrans-str-s (* pemb-str pstroke (- 1 pperm)))
(defvar ptia-s (* pemb-str (- 1 pstroke)))

(defvar pperm-str-as (* pemb-af-str pstroke pperm))
(defvar ptrans-str-as (* pemb-af-str pstroke (- 1 pperm)))
(defvar ptia-as (* pemb-af-str (- 1 pstroke)))

(defvar pperm-str-sw (* pemb-str-warf pstroke pperm))
(defvar ptrans-str-sw (* pemb-str-warf pstroke (- 1 pperm)))
```

```
(defvar ptia-sw (* pemb-str-warf (- 1 pstroke)))

(defvar pperm-str-all (* pemb-all pstroke pperm))
(defvar ptrans-str-all (* pemb-all pstroke (- 1 pperm)))
(defvar ptia-all (* pemb-all (- 1 pstroke)))

(defvar pafib 0.2000)         ; Probability of getting AF
(defvar pafib-stay 0.5000)    ; Probability of staying in AF

(defvar pafib-quin (* rrquin pafib)); Probability of getting AF given Quinidine
(defvar pafib-quin-stay (* rrquin rstayAF)); Probability of staying in AF,given
                                          ; Quinidine
```

C.4 The Event Variable Space and Conditional Distributions

The event variable space for the case study and its associated conditional distributions are described in the following printout of the parameters defined.

```
;;; -*- Mode: LISP; Syntax: Common-Lisp; Package: DYNAMO-INTERFACE; Base: 10 -*-

;;; ****************************************************************************
;;; CASE-STUDY-DISTRIBUTIONS
;;; Module: Cases
;;;
;;; Distributions for case study
;;; Creation date: May 25, 1994
;;;
;;; Copyright (c) 1994, Tze-Yun Leong, Massachusetts Institute of Technology
;;; ****************************************************************************

> ;;; Distributions for action NODRUG.

(distribution state-n/nodrug)
#<Distribution-Table #:G723
    Type:               DISCRETE
    Matrix: #:G723

        NSR-N-WOFF-N         0.05556
         NSR-N-WON-N         0.05556
         NSR-N-WNE-N         0.05556
       NSR-TIA-WOFF-N        0.05556
        NSR-TIA-WON-N        0.05556
        NSR-TIA-WNE-N        0.05556
       NSR-STR-WOFF-N        0.05556
        NSR-STR-WON-N        0.05556
        NSR-STR-WNE-N        0.05556
          AF-N-WOFF-N        0.05556
           AF-N-WON-N        0.05556
           AF-N-WNE-N        0.05556
        AF-TIA-WOFF-N        0.05556
         AF-TIA-WON-N        0.05556
         AF-TIA-WNE-N        0.05556
        AF-STR-WOFF-N        0.05556
         AF-STR-WON-N        0.05556
         AF-STR-WNE-N        0.05556
```

```
> (distribution ?die/nodrug)
#<Distribution-Table #:G741
    Type:               DISCRETE
    Matrix: #:G741
         "Pr{c|r}"                DIE              NOT-DIE
      NSR-N-WOFF-N                PMUSUM           PMUSUM-COMP
       NSR-N-WON-N                PMUSUM           PMUSUM-COMP
       NSR-N-WNE-N                PMUSUM           PMUSUM-COMP
    NSR-TIA-WOFF-N                PMUSUM           PMUSUM-COMP
     NSR-TIA-WON-N                PMUSUM           PMUSUM-COMP
     NSR-TIA-WNE-N                PMUSUM           PMUSUM-COMP
    NSR-STR-WOFF-N                PMUSUM           PMUSUM-COMP
     NSR-STR-WON-N                PMUSUM           PMUSUM-COMP
     NSR-STR-WNE-N                PMUSUM           PMUSUM-COMP
       AF-N-WOFF-N                PMUSUM           PMUSUM-COMP
        AF-N-WON-N                PMUSUM           PMUSUM-COMP
        AF-N-WNE-N                PMUSUM           PMUSUM-COMP
     AF-TIA-WOFF-N                PMUSUM           PMUSUM-COMP
      AF-TIA-WON-N                PMUSUM           PMUSUM-COMP
      AF-TIA-WNE-N                PMUSUM           PMUSUM-COMP
     AF-STR-WOFF-N                PMUSUM           PMUSUM-COMP
      AF-STR-WON-N                PMUSUM           PMUSUM-COMP
      AF-STR-WNE-N                PMUSUM           PMUSUM-COMP

> (distribution ?te/nodrug)
#<Distribution-Table #:G739
    Type:               DISCRETE
    Matrix: #:G739
  "Pr{c|r}"     PERM-STR          TRANS-STR         TIA         NOT-TE
 NSR-N-WOFF-N   PPERM-STR         PTRANS-STR        PTIA        (- 1 PEMB)
  NSR-N-WON-N   PPERM-STR-W       PTRANS-STR-W      PTIA-W      (- 1 PEMB-WARF)
  NSR-N-WNE-N   PPERM-STR         PTRANS-STR        PTIA        (- 1 PEMB)
NSR-TIA-WOFF-N  PPERM-STR-S       PTRANS-STR-S      PTIA-S      (- 1 PEMB-STR)
 NSR-TIA-WON-N  PPERM-STR-SW      PTRANS-STR-SW     PTIA-SW     (- 1 PEMB-STR-WARF)
 NSR-TIA-WNE-N  PPERM-STR-S       PTRANS-STR-S      PTIA-S      (- 1 PEMB-STR)
NSR-STR-WOFF-N  PPERM-STR-S       PTRANS-STR-S      PTIA-S      (- 1 PEMB-STR)
 NSR-STR-WON-N  PPERM-STR-SW      PTRANS-STR-SW     PTIA-SW     (- 1 PEMB-STR-WARF)
 NSR-STR-WNE-N  PPERM-STR-S       PTRANS-STR-S      PTIA-S      (- 1 PEMB-STR)
   AF-N-WOFF-N  PPERM-STR-A       PTRANS-STR-A      PTIA-A      (- 1 PEMB-AF)
    AF-N-WON-N  PPERM-STR-AW      PTRANS-STR-AW     PTIA-AW     (- 1 PEMB-AF-WARF)
    AF-N-WNE-N  PPERM-STR-A       PTRANS-STR-A      PTIA-A      (- 1 PEMB-AF)
 AF-TIA-WOFF-N  PPERM-STR-AS      PTRANS-STR-AS     PTIA-AS     (- 1 PEMB-AF-STR)
  AF-TIA-WON-N  PPERM-STR-ALL     PTRANS-STR-ALL    PTIA-ALL    (- 1 PEMB-ALL)
  AF-TIA-WNE-N  PPERM-STR-AS      PTRANS-STR-AS     PTIA-AS     (- 1 PEMB-AF-STR)
AF-STR-WOFF-N   PPERM-STR-AS      PTRANS-STR-AS     PTIA-AS     (- 1 PEMB-AF-STR)
 AF-STR-WON-N   PPERM-STR-ALL     PTRANS-STR-ALL    PTIA-ALL    (- 1 PEMB-ALL)
 AF-STR-WNE-N   PPERM-STR-AS      PTRANS-STR-AS     PTIA-AS     (- 1 PEMB-AF-STR)

> (distribution ?cerehem/nodrug)
#<Distribution-Table #:G747
    Type:               DISCRETE
    Matrix: #:G747
         "Pr{c|r}"                CHM              NOT-CHM
      NSR-N-WOFF-N                PCEREHEM         (- 1 PCEREHEM)
       NSR-N-WON-N                PCEREHEM-WARF    PCEREHEM-WARF-COMP
       NSR-N-WNE-N                PCEREHEM         (- 1 PCEREHEM)
    NSR-TIA-WOFF-N                PCEREHEM         (- 1 PCEREHEM)
     NSR-TIA-WON-N                PCEREHEM-WARF    PCEREHEM-WARF-COMP
     NSR-TIA-WNE-N                PCEREHEM         (- 1 PCEREHEM)
    NSR-STR-WOFF-N                PCEREHEM         (- 1 PCEREHEM)
     NSR-STR-WON-N                PCEREHEM-WARF    PCEREHEM-WARF-COMP
```

```
     NSR-STR-WNE-N                PCEREHEM          (- 1 PCEREHEM)
       AF-N-WOFF-N                PCEREHEM          (- 1 PCEREHEM)
        AF-N-WON-N                PCEREHEM-WARF     PCEREHEM-WARF-COMP
        AF-N-WNE-N                PCEREHEM          (- 1 PCEREHEM)
     AF-TIA-WOFF-N                PCEREHEM          (- 1 PCEREHEM)
      AF-TIA-WON-N                PCEREHEM-WARF     PCEREHEM-WARF-COMP
      AF-TIA-WNE-N                PCEREHEM          (- 1 PCEREHEM)
     AF-STR-WOFF-N                PCEREHEM          (- 1 PCEREHEM)
      AF-STR-WON-N                PCEREHEM-WARF     PCEREHEM-WARF-COMP
      AF-STR-WNE-N                PCEREHEM          (- 1 PCEREHEM)

> (distribution ?gi-bleeding/nodrug)
#<Distribution-Table #:G727
    Type:               DISCRETE
    Matrix: #:G727
         "Pr{c|r}"                GIB               NOT-GIB
       NSR-N-WOFF-N               PGIB              (- 1 PGIB)
        NSR-N-WON-N               PGIB-WARF         PGIB-WARF-COMP
        NSR-N-WNE-N               PGIB              (- 1 PGIB)
    NSR-TIA-WOFF-N                PGIB              (- 1 PGIB)
     NSR-TIA-WON-N                PGIB-WARF         PGIB-WARF-COMP
     NSR-TIA-WNE-N                PGIB              (- 1 PGIB)
    NSR-STR-WOFF-N                PGIB              (- 1 PGIB)
     NSR-STR-WON-N                PGIB-WARF         PGIB-WARF-COMP
     NSR-STR-WNE-N                PGIB              (- 1 PGIB)
       AF-N-WOFF-N                PGIB              (- 1 PGIB)
        AF-N-WON-N                PGIB-WARF         PGIB-WARF-COMP
        AF-N-WNE-N                PGIB              (- 1 PGIB)
     AF-TIA-WOFF-N                PGIB              (- 1 PGIB)
      AF-TIA-WON-N                PGIB-WARF         PGIB-WARF-COMP
      AF-TIA-WNE-N                PGIB              (- 1 PGIB)
     AF-STR-WOFF-N                PGIB              (- 1 PGIB)
      AF-STR-WON-N                PGIB-WARF         PGIB-WARF-COMP
      AF-STR-WNE-N                PGIB              (- 1 PGIB)

> (distribution ?die-te/nodrug)
#<Distribution-Table #:G721
    Type:               DISCRETE
    Matrix: #:G721
         "Pr{c|r}"                DIE-TE            NOT-DIE-TE
          PERM-STR                PDSTROKE          (- 1 PDSTROKE)
         TRANS-STR                PDSTROKE          (- 1 PDSTROKE)
               TIA                0.00000           1.00000
            NOT-TE                0.00000           1.00000

> (distribution ?die-cerehem/nodrug)
#<Distribution-Table #:G745
    Type:               DISCRETE
    Matrix: #:G745
         "Pr{c|r}"                DIE-CHM           NOT-DIE-CHM
               CHM                PDIECHM           (- 1 PDIECHM)
          NOT-CHM                 0.00000           1.00000

> (distribution ?die-gi-bleeding/nodrug)
#<Distribution-Table #:G735
    Type:               DISCRETE
    Matrix: #:G735
         "Pr{c|r}"                DIE-GIB           NOT-DIE-GIB
               GIB                PDIEGIB           (- 1 PDIEGIB)
          NOT-GIB                 0.00000           1.00000
```

```
> (distribution afib?/nodrug)
#<Distribution-Table #:G737
    Type:               DISCRETE
    Matrix: #:G737
         “Pr{c|r}”                NSR-A            AFIB-A
      NSR-N-WOFF-N                (- 1 PAFIB)      PAFIB
       NSR-N-WON-N                (- 1 PAFIB)      PAFIB
       NSR-N-WNE-N                (- 1 PAFIB)      PAFIB
    NSR-TIA-WOFF-N                (- 1 PAFIB)      PAFIB
     NSR-TIA-WON-N                (- 1 PAFIB)      PAFIB
     NSR-TIA-WNE-N                (- 1 PAFIB)      PAFIB
    NSR-STR-WOFF-N                (- 1 PAFIB)      PAFIB
     NSR-STR-WON-N                (- 1 PAFIB)      PAFIB
     NSR-STR-WNE-N                (- 1 PAFIB)      PAFIB
       AF-N-WOFF-N                (- 1 PAFIB-STAY) PAFIB-STAY
        AF-N-WON-N                (- 1 PAFIB-STAY) PAFIB-STAY
        AF-N-WNE-N                (- 1 PAFIB-STAY) PAFIB-STAY
     AF-TIA-WOFF-N                (- 1 PAFIB-STAY) PAFIB-STAY
      AF-TIA-WON-N                (- 1 PAFIB-STAY) PAFIB-STAY
      AF-TIA-WNE-N                (- 1 PAFIB-STAY) PAFIB-STAY
     AF-STR-WOFF-N                (- 1 PAFIB-STAY) PAFIB-STAY
      AF-STR-WON-N                (- 1 PAFIB-STAY) PAFIB-STAY
      AF-STR-WNE-N                (- 1 PAFIB-STAY) PAFIB-STAY

> (distribution status?/nodrug)
#<Distribution-Table #:G725
    Type:               DISCRETE
    Matrix: #:G725
         “Pr{c|r}”                                ALIVE-A          DEAD-A
[DIE DIE-GIB DIE-CHM DIE-TE]                      0.00000          1.00000
(NOT-DIE NOT-DIE-GIB NOT-DIE-CHM NOT-DIE-TE)      1.00000          0.00000
> (distribution stroke?/nodrug)
#<Distribution-Table #:G743
    Type:               DISCRETE
    Matrix: #:G743
         “Pr{c|r}”                STR-A            TIA-A            NO-STR-A
[CHM PERM-STR NSR-STR-WOFF-N
  NSR-STR-WON-N NSR-STR-WNE-N
  AF-STR-WOFF-N AF-STR-WON-N
  AF-STR-WNE-N]                   1.00000          0.00000          0.00000
(NOT-CHM TRANS-STR NSR-N-WOFF-N)  0.00000          1.00000          0.00000
(NOT-CHM TIA NSR-N-WOFF-N)        0.00000          1.00000          0.00000
(NOT-CHM NOT-TE NSR-N-WOFF-N)     0.00000          0.00000          1.00000
(NOT-CHM TRANS-STR NSR-N-WON-N)   0.00000          1.00000          0.00000
(NOT-CHM TIA NSR-N-WON-N)         0.00000          1.00000          0.00000
(NOT-CHM NOT-TE NSR-N-WON-N)      0.00000          0.00000          1.00000
(NOT-CHM TRANS-STR NSR-N-WNE-N)   0.00000          1.00000          0.00000
(NOT-CHM TIA NSR-N-WNE-N)         0.00000          1.00000          0.00000
(NOT-CHM NOT-TE NSR-N-WNE-N)      0.00000          0.00000          1.00000
(NOT-CHM TRANS-STR NSR-TIA-WOFF-N) 0.00000         1.00000          0.00000
(NOT-CHM TIA NSR-TIA-WOFF-N)      0.00000          1.00000          0.00000
(NOT-CHM NOT-TE NSR-TIA-WOFF-N)   0.00000          1.00000          0.00000
(NOT-CHM TRANS-STR NSR-TIA-WON-N) 0.00000          1.00000          0.00000
(NOT-CHM TIA NSR-TIA-WON-N)       0.00000          1.00000          0.00000
(NOT-CHM NOT-TE NSR-TIA-WON-N)    0.00000          1.00000          0.00000
(NOT-CHM TRANS-STR NSR-TIA-WNE-N) 0.00000          1.00000          0.00000
(NOT-CHM TIA NSR-TIA-WNE-N)       0.00000          1.00000          0.00000
(NOT-CHM NOT-TE NSR-TIA-WNE-N)    0.00000          1.00000          0.00000
(NOT-CHM TRANS-STR AF-N-WOFF-N)   0.00000          1.00000          0.00000
(NOT-CHM TIA AF-N-WOFF-N)         0.00000          1.00000          0.00000
(NOT-CHM NOT-TE AF-N-WOFF-N)      0.00000          0.00000          1.00000
```

```
(NOT-CHM TRANS-STR AF-N-WON-N)      0.00000          1.00000          0.00000
(NOT-CHM TIA AF-N-WON-N)            0.00000          1.00000          0.00000
(NOT-CHM NOT-TE AF-N-WON-N)         0.00000          0.00000          1.00000
(NOT-CHM TRANS-STR AF-N-WNE-N)      0.00000          1.00000          0.00000
(NOT-CHM TIA AF-N-WNE-N)            0.00000          1.00000          0.00000
(NOT-CHM NOT-TE AF-N-WNE-N)         0.00000          0.00000          1.00000
(NOT-CHM TRANS-STR AF-TIA-WOFF-N)   0.00000          1.00000          0.00000
(NOT-CHM TIA AF-TIA-WOFF-N)         0.00000          1.00000          0.00000
(NOT-CHM NOT-TE AF-TIA-WOFF-N)      0.00000          1.00000          0.00000
(NOT-CHM TRANS-STR AF-TIA-WON-N)    0.00000          1.00000          0.00000
(NOT-CHM TIA AF-TIA-WON-N)          0.00000          1.00000          0.00000
(NOT-CHM NOT-TE AF-TIA-WON-N)       0.00000          1.00000          0.00000
(NOT-CHM TRANS-STR AF-TIA-WNE-N)    0.00000          1.00000          0.00000
(NOT-CHM TIA AF-TIA-WNE-N)          0.00000          1.00000          0.00000
(NOT-CHM NOT-TE AF-TIA-WNE-N)       0.00000          1.00000          0.00000

> (distribution warfarin?/nodrug)
#<Distribution-Table #:G733
    Type:               DISCRETE
    Matrix: #:G733
         “Pr{c|r}”                  WARFON-A         WARFOFF-A
[NSR-N-WNE-N NSR-TIA-WNE-N
  NSR-STR-WNE-N AF-N-WNE-N
  AF-TIA-WNE-N AF-STR-WNE-N]        0.00000          1.00000
(PERM-STR NSR-N-WOFF-N)             0.00000          1.00000
(TRANS-STR NSR-N-WOFF-N)            0.00000          1.00000
(TIA NSR-N-WOFF-N)                  0.00000          1.00000
(NOT-TE NSR-N-WOFF-N)               0.00000          1.00000
(PERM-STR NSR-N-WON-N)              1.00000          0.00000
(TRANS-STR NSR-N-WON-N)             1.00000          0.00000
 (TIA NSR-N-WON-N)                  1.00000          0.00000
(NOT-TE NSR-N-WON-N)                1.00000          0.00000
(PERM-STR NSR-TIA-WOFF-N)           0.00000          1.00000
(TRANS-STR NSR-TIA-WOFF-N)          0.00000          1.00000
(TIA NSR-TIA-WOFF-N)                0.00000          1.00000
(NOT-TE NSR-TIA-WOFF-N)             0.00000          1.00000
(PERM-STR NSR-TIA-WON-N)            1.00000          0.00000
(TRANS-STR NSR-TIA-WON-N)           1.00000          0.00000
(TIA NSR-TIA-WON-N)                 1.00000          0.00000
(NOT-TE NSR-TIA-WON-N)              1.00000          0.00000
(PERM-STR NSR-STR-WOFF-N)           0.00000          1.00000
(TRANS-STR NSR-STR-WOFF-N)          0.00000          1.00000
(TIA NSR-STR-WOFF-N)                0.00000          1.00000
(NOT-TE NSR-STR-WOFF-N)             0.00000          1.00000
(PERM-STR NSR-STR-WON-N)            1.00000          0.00000
(TRANS-STR NSR-STR-WON-N)           1.00000          0.00000
(TIA NSR-STR-WON-N)                 1.00000          0.00000
(NOT-TE NSR-STR-WON-N)              1.00000          0.00000
(PERM-STR AF-N-WOFF-N)              1.00000          0.00000
(TRANS-STR AF-N-WOFF-N)             1.00000          0.00000
(TIA AF-N-WOFF-N)                   1.00000          0.00000
(NOT-TE AF-N-WOFF-N)                0.00000          1.00000
(PERM-STR AF-N-WON-N)               1.00000          0.00000
(TRANS-STR AF-N-WON-N)              1.00000          0.00000
 (TIA AF-N-WON-N)                   1.00000          0.00000
(NOT-TE AF-N-WON-N)                 1.00000          0.00000
(PERM-STR AF-TIA-WOFF-N)            1.00000          0.00000
(TRANS-STR AF-TIA-WOFF-N)           1.00000          0.00000
(TIA AF-TIA-WOFF-N)                 1.00000          0.00000
(NOT-TE AF-TIA-WOFF-N)              0.00000          1.00000
(PERM-STR AF-TIA-WON-N)             1.00000          0.00000
```

```
(TRANS-STR AF-TIA-WON-N)            1.00000          0.00000
(TIA AF-TIA-WON-N)                  1.00000          0.00000
(NOT-TE AF-TIA-WON-N)               1.00000          0.00000
(PERM-STR AF-STR-WOFF-N)            1.00000          0.00000
(TRANS-STR AF-STR-WOFF-N)           1.00000          0.00000
(TIA AF-STR-WOFF-N)                 1.00000          0.00000
(NOT-TE AF-STR-WOFF-N)              0.00000          1.00000
(PERM-STR AF-STR-WON-N)             1.00000          0.00000
(TRANS-STR AF-STR-WON-N)            1.00000          0.00000
(TIA AF-STR-WON-N)                  1.00000          0.00000
(NOT-TE AF-STR-WON-N)               1.00000          0.00000

> (distribution warfarin-ok?/nodrug)
#<Distribution-Table #:G731
    Type:               DISCRETE
    Matrix: #:G731
         “Pr{c|r}”                  WARFOK-A         WARFNE-A
[CHM GIB NSR-N-WNE-N NSR-TIA-WNE-N
  NSR-STR-WNE-N AF-N-WNE-N
  F-TIA-WNE-N AF-STR-WNE-N]         0.00000          1.00000
(NOT-CHM NOT-GIB NSR-N-WOFF-N)      1.00000          0.00000
(NOT-CHM NOT-GIB NSR-N-WON-N)       1.00000          0.00000
(NOT-CHM NOT-GIB NSR-TIA-WOFF-N)    1.00000          0.00000
(NOT-CHM NOT-GIB NSR-TIA-WON-N)     1.00000          0.00000
(NOT-CHM NOT-GIB NSR-STR-WOFF-N)    1.00000          0.00000
(NOT-CHM NOT-GIB NSR-STR-WON-N)     1.00000          0.00000
(NOT-CHM NOT-GIB AF-N-WOFF-N)       1.00000          0.00000
(NOT-CHM NOT-GIB AF-N-WON-N)        1.00000          0.00000
(NOT-CHM NOT-GIB AF-TIA-WOFF-N)     1.00000          0.00000
(NOT-CHM NOT-GIB AF-TIA-WON-N)      1.00000          0.00000
(NOT-CHM NOT-GIB AF-STR-WOFF-N)     1.00000          0.00000
(NOT-CHM NOT-GIB AF-STR-WON-N)      1.00000          0.00000

;;; Output of different format for this large table.
> (distribution state/nodrug)
#<Distribution-Table #:G729
    Type:               DISCRETE
    Matrix: #:G729
         Distribution-Table #:G601

For OUTCOME: NSR-N-WOFF
NSR-N-WOFF |   (NO-STR-A NSR-A WARFOFF-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: NSR-N-WON
NSR-N-WON |    (NO-STR-A NSR-A WARFON-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: NSR-N-WNE
NSR-N-WNE |    (NO-STR-A NSR-A WARFON-A WARFNE-A ALIVE-A): 1.00000
NSR-N-WNE |    (NO-STR-A NSR-A WARFOFF-A WARFNE-A ALIVE-A): 1.00000

For OUTCOME: NSR-TIA-WOFF
NSR-TIA-WOFF |  (TIA-A NSR-A WARFOFF-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: NSR-TIA-WON
NSR-TIA-WON |  (TIA-A NSR-A WARFON-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: NSR-TIA-WNE
NSR-TIA-WNE |  (TIA-A NSR-A WARFON-A WARFNE-A ALIVE-A): 1.00000
NSR-TIA-WNE |  (TIA-A NSR-A WARFOFF-A WARFNE-A ALIVE-A): 1.00000
```

```
For OUTCOME: NSR-STR-WOFF
NSR-STR-WOFF |  (STR-A NSR-A WARFOFF-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: NSR-STR-WON
NSR-STR-WON |  (STR-A NSR-A WARFON-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: NSR-STR-WNE
NSR-STR-WNE |  (STR-A NSR-A WARFON-A WARFNE-A ALIVE-A): 1.00000
NSR-STR-WNE |  (STR-A NSR-A WARFOFF-A WARFNE-A ALIVE-A): 1.00000

For OUTCOME: AF-N-WOFF
AF-N-WOFF |    (NO-STR-A AFIB-A WARFOFF-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: AF-N-WON
AF-N-WON |     (NO-STR-A AFIB-A WARFON-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: AF-N-WNE
AF-N-WNE |     (NO-STR-A AFIB-A WARFON-A WARFNE-A ALIVE-A): 1.00000
AF-N-WNE |     (NO-STR-A AFIB-A WARFOFF-A WARFNE-A ALIVE-A): 1.00000

For OUTCOME: AF-TIA-WOFF
AF-TIA-WOFF |  (TIA-A AFIB-A WARFOFF-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: AF-TIA-WON
AF-TIA-WON |   (TIA-A AFIB-A WARFON-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: AF-TIA-WNE
AF-TIA-WNE |   (TIA-A AFIB-A WARFON-A WARFNE-A ALIVE-A): 1.00000
AF-TIA-WNE |   (TIA-A AFIB-A WARFOFF-A WARFNE-A ALIVE-A): 1.00000

For OUTCOME: AF-STR-WOFF
AF-STR-WOFF |  (STR-A AFIB-A WARFOFF-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: AF-STR-WON
AF-STR-WON |   (STR-A AFIB-A WARFON-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: AF-STR-WNE
AF-STR-WNE |   (STR-A AFIB-A WARFON-A WARFNE-A ALIVE-A): 1.00000
AF-STR-WNE |   (STR-A AFIB-A WARFOFF-A WARFNE-A ALIVE-A): 1.00000

For OUTCOME: DEAD
DEAD |         DEAD-A: 1.00000

> ;;; Distributions for action QUINIDINE.

(distribution state-n/quinidine)
#<Distribution-Table #:G767
    Type:               DISCRETE
    Matrix: #:G767

        NSR-N-WOFF-N         0.05556
         NSR-N-WON-N         0.05556
         NSR-N-WNE-N         0.05556
      NSR-TIA-WOFF-N         0.05556
       NSR-TIA-WON-N         0.05556
       NSR-TIA-WNE-N         0.05556
      NSR-STR-WOFF-N         0.05556
       NSR-STR-WON-N         0.05556
       NSR-STR-WNE-N         0.05556
         AF-N-WOFF-N         0.05556
          AF-N-WON-N         0.05556
```

```
            AF-N-WNE-N         0.05556
          AF-TIA-WOFF-N        0.05556
           AF-TIA-WON-N        0.05556
           AF-TIA-WNE-N        0.05556
          AF-STR-WOFF-N        0.05556
           AF-STR-WON-N        0.05556
           AF-STR-WNE-N        0.05556

> (distribution ?die/quinidine)
#<Distribution-Table #:G765
    Type:              DISCRETE
    Matrix: #:G765
         "Pr{c|r}"                DIE              NOT-DIE
      NSR-N-WOFF-N                PMUSUM-QUIN      PMUSUM-QUIN-COMP
       NSR-N-WON-N                PMUSUM-QUIN      PMUSUM-QUIN-COMP
       NSR-N-WNE-N                PMUSUM-QUIN      PMUSUM-QUIN-COMP
     NSR-TIA-WOFF-N               PMUSUM-QUIN      PMUSUM-QUIN-COMP
      NSR-TIA-WON-N               PMUSUM-QUIN      PMUSUM-QUIN-COMP
      NSR-TIA-WNE-N               PMUSUM-QUIN      PMUSUM-QUIN-COMP
     NSR-STR-WOFF-N               PMUSUM-QUIN      PMUSUM-QUIN-COMP
      NSR-STR-WON-N               PMUSUM-QUIN      PMUSUM-QUIN-COMP
      NSR-STR-WNE-N               PMUSUM-QUIN      PMUSUM-QUIN-COMP
       AF-N-WOFF-N                PMUSUM-QUIN      PMUSUM-QUIN-COMP
        AF-N-WON-N                PMUSUM-QUIN      PMUSUM-QUIN-COMP
        AF-N-WNE-N                PMUSUM-QUIN      PMUSUM-QUIN-COMP
      AF-TIA-WOFF-N               PMUSUM-QUIN      PMUSUM-QUIN-COMP
       AF-TIA-WON-N               PMUSUM-QUIN      PMUSUM-QUIN-COMP
       AF-TIA-WNE-N               PMUSUM-QUIN      PMUSUM-QUIN-COMP
      AF-STR-WOFF-N               PMUSUM-QUIN      PMUSUM-QUIN-COMP
       AF-STR-WON-N               PMUSUM-QUIN      PMUSUM-QUIN-COMP
       AF-STR-WNE-N               PMUSUM-QUIN      PMUSUM-QUIN-COMP

> (distribution ?te/quinidine)
#<Distribution-Table #:G761
    Type:              DISCRETE
    Matrix: #:G761
     "Pr{c|r}"     PERM-STR          TRANS-STR         TIA        NOT-TE
  NSR-N-WOFF-N     PPERM-STR         PTRANS-STR        PTIA       (- 1 PEMB)
   NSR-N-WON-N     PPERM-STR-W       PTRANS-STR-W      PTIA-W     (- 1 PEMB-WARF)
   NSR-N-WNE-N     PPERM-STR         PTRANS-STR        PTIA       (- 1 PEMB)
NSR-TIA-WOFF-N     PPERM-STR-S       PTRANS-STR-S      PTIA-S     (- 1 PEMB-STR)
 NSR-TIA-WON-N     PPERM-STR-SW      PTRANS-STR-SW     PTIA-SW    (- 1 PEMB-STR-WARF)
 NSR-TIA-WNE-N     PPERM-STR-S       PTRANS-STR-S      PTIA-S     (- 1 PEMB-STR)
NSR-STR-WOFF-N     PPERM-STR-S       PTRANS-STR-S      PTIA-S     (- 1 PEMB-STR)
 NSR-STR-WON-N     PPERM-STR-SW      PTRANS-STR-SW     PTIA-SW    (- 1 PEMB-STR-WARF)
 NSR-STR-WNE-N     PPERM-STR-S       PTRANS-STR-S      PTIA-S     (- 1 PEMB-STR)
   AF-N-WOFF-N     PPERM-STR-A       PTRANS-STR-A      PTIA-A     (- 1 PEMB-AF)
    AF-N-WON-N     PPERM-STR-AW      PTRANS-STR-AW     PTIA-AW    (- 1 PEMB-AF-WARF)
    AF-N-WNE-N     PPERM-STR-A       PTRANS-STR-A      PTIA-A     (- 1 PEMB-AF)
 AF-TIA-WOFF-N     PPERM-STR-AS      PTRANS-STR-AS     PTIA-AS    (- 1 PEMB-AF-STR)
  AF-TIA-WON-N     PPERM-STR-ALL     PTRANS-STR-ALL    PTIA-ALL   (- 1 PEMB-ALL)
  AF-TIA-WNE-N     PPERM-STR-AS      PTRANS-STR-AS     PTIA-AS    (- 1 PEMB-AF-STR)
 AF-STR-WOFF-N     PPERM-STR-AS      PTRANS-STR-AS     PTIA-AS    (- 1 PEMB-AF-STR)
  AF-STR-WON-N     PPERM-STR-ALL     PTRANS-STR-ALL    PTIA-ALL   (- 1 PEMB-ALL)
  AF-STR-WNE-N     PPERM-STR-AS      PTRANS-STR-AS     PTIA-AS    (- 1 PEMB-AF-STR)
```

```
> (distribution ?cerehem/quinidine)
#<Distribution-Table #:G773
    Type:               DISCRETE
    Matrix: #:G773
         "Pr{c|r}"                  CHM               NOT-CHM
      NSR-N-WOFF-N                  PCEREHEM          (- 1 PCEREHEM)
       NSR-N-WON-N                  PCEREHEM-WARF     PCEREHEM-WARF-COMP
       NSR-N-WNE-N                  PCEREHEM          (- 1 PCEREHEM)
    NSR-TIA-WOFF-N                  PCEREHEM          (- 1 PCEREHEM)
     NSR-TIA-WON-N                  PCEREHEM-WARF     PCEREHEM-WARF-COMP
     NSR-TIA-WNE-N                  PCEREHEM          (- 1 PCEREHEM)
    NSR-STR-WOFF-N                  PCEREHEM          (- 1 PCEREHEM)
     NSR-STR-WON-N                  PCEREHEM-WARF     PCEREHEM-WARF-COMP
     NSR-STR-WNE-N                  PCEREHEM          (- 1 PCEREHEM)
       AF-N-WOFF-N                  PCEREHEM          (- 1 PCEREHEM)
        AF-N-WON-N                  PCEREHEM-WARF     PCEREHEM-WARF-COMP
        AF-N-WNE-N                  PCEREHEM          (- 1 PCEREHEM)
     AF-TIA-WOFF-N                  PCEREHEM          (- 1 PCEREHEM)
      AF-TIA-WON-N                  PCEREHEM-WARF     PCEREHEM-WARF-COMP
      AF-TIA-WNE-N                  PCEREHEM          (- 1 PCEREHEM)
     AF-STR-WOFF-N                  PCEREHEM          (- 1 PCEREHEM)
      AF-STR-WON-N                  PCEREHEM-WARF     PCEREHEM-WARF-COMP
      AF-STR-WNE-N                  PCEREHEM          (- 1 PCEREHEM)

> (distribution ?gi-bleeding/quinidine)
#<Distribution-Table #:G757
    Type:               DISCRETE
    Matrix: #:G757
         "Pr{c|r}"                  GIB               NOT-GIB
      NSR-N-WOFF-N                  PGIB              (- 1 PGIB)
       NSR-N-WON-N                  PGIB-WARF         PGIB-WARF-COMP
       NSR-N-WNE-N                  PGIB              (- 1 PGIB)
    NSR-TIA-WOFF-N                  PGIB              (- 1 PGIB)
     NSR-TIA-WON-N                  PGIB-WARF         PGIB-WARF-COMP
     NSR-TIA-WNE-N                  PGIB              (- 1 PGIB)
    NSR-STR-WOFF-N                  PGIB              (- 1 PGIB)
     NSR-STR-WON-N                  PGIB-WARF         PGIB-WARF-COMP
     NSR-STR-WNE-N                  PGIB              (- 1 PGIB)
       AF-N-WOFF-N                  PGIB              (- 1 PGIB)
        AF-N-WON-N                  PGIB-WARF         PGIB-WARF-COMP
        AF-N-WNE-N                  PGIB              (- 1 PGIB)
     AF-TIA-WOFF-N                  PGIB              (- 1 PGIB)
      AF-TIA-WON-N                  PGIB-WARF         PGIB-WARF-COMP
      AF-TIA-WNE-N                  PGIB              (- 1 PGIB)
     AF-STR-WOFF-N                  PGIB              (- 1 PGIB)
      AF-STR-WON-N                  PGIB-WARF         PGIB-WARF-COMP
      AF-STR-WNE-N                  PGIB              (- 1 PGIB)

> (distribution ?die-te/quinidine)
#<Distribution-Table #:G751
    Type:               DISCRETE
    Matrix: #:G751
         "Pr{c|r}"                  DIE-TE            NOT-DIE-TE
          PERM-STR                  PDSTROKE          (- 1 PDSTROKE)
         TRANS-STR                  PDSTROKE          (- 1 PDSTROKE)
               TIA                  0.00000           1.00000
            NOT-TE                  0.00000           1.00000
```

```
> (distribution ?die-cerehem/quinidine)
#<Distribution-Table #:G759
    Type:               DISCRETE
    Matrix: #:G759
         “Pr{c|r}”                 DIE-CHM          NOT-DIE-CHM
              CHM                  PDIECHM          (- 1 PDIECHM)
            NOT-CHM                0.00000          1.00000

> (distribution ?die-gi-bleeding/quinidine)
#<Distribution-Table #:G771
    Type:               DISCRETE
    Matrix: #:G771
         “Pr{c|r}”                 DIE-GIB          NOT-DIE-GIB
              GIB                  PDIEGIB          (- 1 PDIEGIB)
            NOT-GIB                0.00000          1.00000

> (distribution afib?/quinidine)
#<Distribution-Table #:G753
    Type:               DISCRETE
    Matrix: #:G753
         “Pr{c|r}”                 NSR-A                  AFIB-A
      NSR-N-WOFF-N                 (- 1 PAFIB-QUIN)       PAFIB-QUIN
       NSR-N-WON-N                 (- 1 PAFIB-QUIN)       PAFIB-QUIN
       NSR-N-WNE-N                 (- 1 PAFIB-QUIN)       PAFIB-QUIN
    NSR-TIA-WOFF-N                 (- 1 PAFIB-QUIN)       PAFIB-QUIN
     NSR-TIA-WON-N                 (- 1 PAFIB-QUIN)       PAFIB-QUIN
     NSR-TIA-WNE-N                 (- 1 PAFIB-QUIN)       PAFIB-QUIN
    NSR-STR-WOFF-N                 (- 1 PAFIB-QUIN)       PAFIB-QUIN
     NSR-STR-WON-N                 (- 1 PAFIB-QUIN)       PAFIB-QUIN
     NSR-STR-WNE-N                 (- 1 PAFIB-QUIN)       PAFIB-QUIN
       AF-N-WOFF-N                 (- 1 PAFIB-QUIN-STAY)  PAFIB-QUIN-STAY
        AF-N-WON-N                 (- 1 PAFIB-QUIN-STAY)  PAFIB-QUIN-STAY
        AF-N-WNE-N                 (- 1 PAFIB-QUIN-STAY)  PAFIB-QUIN-STAY
     AF-TIA-WOFF-N                 (- 1 PAFIB-QUIN-STAY)  PAFIB-QUIN-STAY
      AF-TIA-WON-N                 (- 1 PAFIB-QUIN-STAY)  PAFIB-QUIN-STAY
      AF-TIA-WNE-N                 (- 1 PAFIB-QUIN-STAY)  PAFIB-QUIN-STAY
     AF-STR-WOFF-N                 (- 1 PAFIB-QUIN-STAY)  PAFIB-QUIN-STAY
      AF-STR-WON-N                 (- 1 PAFIB-QUIN-STAY)  PAFIB-QUIN-STAY
      AF-STR-WNE-N                 (- 1 PAFIB-QUIN-STAY)  PAFIB-QUIN-STAY

> (distribution status?/quinidine)
#<Distribution-Table #:G775
    Type:               DISCRETE
    Matrix: #:G775
         “Pr{c|r}”                                 ALIVE-A          DEAD-A
[DIE DIE-GIB DIE-CHM DIE-TE]                       0.00000          1.00000
(NOT-DIE NOT-DIE-GIB NOT-DIE-CHM NOT-DIE-TE)       1.00000          0.00000

> (distribution stroke?/quinidine)
#<Distribution-Table #:G769
    Type:               DISCRETE
    Matrix: #:G769
         “Pr{c|r}”                 STR-A            TIA-A            NO-STR-A
[CHM PERM-STR NSR-STR-WOFF-N
  NSR-STR-WON-N NSR-STR-WNE-N
  AF-STR-WOFF-N AF-STR-WON-N
  AF-STR-WNE-N]                    1.00000          0.00000          0.00000
(NOT-CHM TRANS-STR NSR-N-WOFF-N)   0.00000          1.00000          0.00000
(NOT-CHM TIA NSR-N-WOFF-N)         0.00000          1.00000          0.00000
(NOT-CHM NOT-TE NSR-N-WOFF-N)      0.00000          0.00000          1.00000
(NOT-CHM TRANS-STR NSR-N-WON-N)    0.00000          1.00000          0.00000
```

```
(NOT-CHM TIA NSR-N-WON-N)           0.00000          1.00000          0.00000
(NOT-CHM NOT-TE NSR-N-WON-N)        0.00000          0.00000          1.00000
(NOT-CHM TRANS-STR NSR-N-WNE-N)     0.00000          1.00000          0.00000
(NOT-CHM TIA NSR-N-WNE-N)           0.00000          1.00000          0.00000
(NOT-CHM NOT-TE NSR-N-WNE-N)        0.00000          0.00000          1.00000
(NOT-CHM TRANS-STR NSR-TIA-WOFF-N)  0.00000          1.00000          0.00000
(NOT-CHM TIA NSR-TIA-WOFF-N)        0.00000          1.00000          0.00000
(NOT-CHM NOT-TE NSR-TIA-WOFF-N)     0.00000          1.00000          0.00000
(NOT-CHM TRANS-STR NSR-TIA-WON-N)   0.00000          1.00000          0.00000
(NOT-CHM TIA NSR-TIA-WON-N)         0.00000          1.00000          0.00000
(NOT-CHM NOT-TE NSR-TIA-WON-N)      0.00000          1.00000          0.00000
(NOT-CHM TRANS-STR NSR-TIA-WNE-N)   0.00000          1.00000          0.00000
(NOT-CHM TIA NSR-TIA-WNE-N)         0.00000          1.00000          0.00000
(NOT-CHM NOT-TE NSR-TIA-WNE-N)      0.00000          1.00000          0.00000
(NOT-CHM TRANS-STR AF-N-WOFF-N)     0.00000          1.00000          0.00000
(NOT-CHM TIA AF-N-WOFF-N)           0.00000          1.00000          0.00000
(NOT-CHM NOT-TE AF-N-WOFF-N)        0.00000          0.00000          1.00000
(NOT-CHM TRANS-STR AF-N-WON-N)      0.00000          1.00000          0.00000
(NOT-CHM TIA AF-N-WON-N)            0.00000          1.00000          0.00000
(NOT-CHM NOT-TE AF-N-WON-N)         0.00000          0.00000          1.00000
(NOT-CHM TRANS-STR AF-N-WNE-N)      0.00000          1.00000          0.00000
(NOT-CHM TIA AF-N-WNE-N)            0.00000          1.00000          0.00000
(NOT-CHM NOT-TE AF-N-WNE-N)         0.00000          0.00000          1.00000
(NOT-CHM TRANS-STR AF-TIA-WOFF-N)   0.00000          1.00000          0.00000
(NOT-CHM TIA AF-TIA-WOFF-N)         0.00000          1.00000          0.00000
(NOT-CHM NOT-TE AF-TIA-WOFF-N)      0.00000          1.00000          0.00000
(NOT-CHM TRANS-STR AF-TIA-WON-N)    0.00000          1.00000          0.00000
(NOT-CHM TIA AF-TIA-WON-N)          0.00000          1.00000          0.00000
(NOT-CHM NOT-TE AF-TIA-WON-N)       0.00000          1.00000          0.00000
(NOT-CHM TRANS-STR AF-TIA-WNE-N)    0.00000          1.00000          0.00000
(NOT-CHM TIA AF-TIA-WNE-N)          0.00000          1.00000          0.00000
(NOT-CHM NOT-TE AF-TIA-WNE-N)       0.00000          1.00000          0.00000

> (distribution warfarin?/quinidine)
#<Distribution-Table #:G749
    Type:               DISCRETE
    Matrix: #:G749
         “Pr{c|r}”                  WARFON-A         WARFOFF-A
[NSR-N-WNE-N NSR-TIA-WNE-N
  NSR-STR-WNE-N AF-N-WNE-N
  AF-TIA-WNE-N AF-STR-WNE-N]        0.00000          1.00000
(PERM-STR NSR-N-WOFF-N)             0.00000          1.00000
(TRANS-STR NSR-N-WOFF-N)            0.00000          1.00000
(TIA NSR-N-WOFF-N)                  0.00000          1.00000
(NOT-TE NSR-N-WOFF-N)               0.00000          1.00000
(PERM-STR NSR-N-WON-N)              1.00000          0.00000
(TRANS-STR NSR-N-WON-N)             1.00000          0.00000
(TIA NSR-N-WON-N)                   1.00000          0.00000
(NOT-TE NSR-N-WON-N)                1.00000          0.00000
(PERM-STR NSR-TIA-WOFF-N)           0.00000          1.00000
(TRANS-STR NSR-TIA-WOFF-N)          0.00000          1.00000
(TIA NSR-TIA-WOFF-N)                0.00000          1.00000
(NOT-TE NSR-TIA-WOFF-N)             0.00000          1.00000
(PERM-STR NSR-TIA-WON-N)            1.00000          0.00000
(TRANS-STR NSR-TIA-WON-N)           1.00000          0.00000
(TIA NSR-TIA-WON-N)                 1.00000          0.00000
(NOT-TE NSR-TIA-WON-N)              1.00000          0.00000
(PERM-STR NSR-STR-WOFF-N)           0.00000          1.00000
(TRANS-STR NSR-STR-WOFF-N)          0.00000          1.00000
(TIA NSR-STR-WOFF-N)                0.00000          1.00000
(NOT-TE NSR-STR-WOFF-N)             0.00000          1.00000
```

```
(PERM-STR NSR-STR-WON-N)            1.00000          0.00000
(TRANS-STR NSR-STR-WON-N)           1.00000          0.00000
(TIA NSR-STR-WON-N)                 1.00000          0.00000
(NOT-TE NSR-STR-WON-N)              1.00000          0.00000
(PERM-STR AF-N-WOFF-N)              1.00000          0.00000
(TRANS-STR AF-N-WOFF-N)             1.00000          0.00000
(TIA AF-N-WOFF-N)                   1.00000          0.00000
(NOT-TE AF-N-WOFF-N)                0.00000          1.00000
(PERM-STR AF-N-WON-N)               1.00000          0.00000
(TRANS-STR AF-N-WON-N)              1.00000          0.00000
 (TIA AF-N-WON-N)                   1.00000          0.00000
(NOT-TE AF-N-WON-N)                 1.00000          0.00000
(PERM-STR AF-TIA-WOFF-N)            1.00000          0.00000
(TRANS-STR AF-TIA-WOFF-N)           1.00000          0.00000
(TIA AF-TIA-WOFF-N)                 1.00000          0.00000
(NOT-TE AF-TIA-WOFF-N)              0.00000          1.00000
(PERM-STR AF-TIA-WON-N)             1.00000          0.00000
(TRANS-STR AF-TIA-WON-N)            1.00000          0.00000
(TIA AF-TIA-WON-N)                  1.00000          0.00000
(NOT-TE AF-TIA-WON-N)               1.00000          0.00000
(PERM-STR AF-STR-WOFF-N)            1.00000          0.00000
(TRANS-STR AF-STR-WOFF-N)           1.00000          0.00000
(TIA AF-STR-WOFF-N)                 1.00000          0.00000
(NOT-TE AF-STR-WOFF-N)              0.00000          1.00000
(PERM-STR AF-STR-WON-N)             1.00000          0.00000
(TRANS-STR AF-STR-WON-N)            1.00000          0.00000
(TIA AF-STR-WON-N)                  1.00000          0.00000
(NOT-TE AF-STR-WON-N)               1.00000          0.00000

> (distribution warfarin-ok?/quinidine)
#<Distribution-Table #:G763
    Type:               DISCRETE
    Matrix: #:G763
         “Pr{c|r}”                  WARFOK-A         WARFNE-A
[CHM GIB NSR-N-WNE-N NSR-TIA-WNE-N
  NSR-STR-WNE-N AF-N-WNE-N
  AF-TIA-WNE-N AF-STR-WNE-N]        0.00000          1.00000
(NOT-CHM NOT-GIB NSR-N-WOFF-N)      1.00000          0.00000
(NOT-CHM NOT-GIB NSR-N-WON-N)       1.00000          0.00000
(NOT-CHM NOT-GIB NSR-TIA-WOFF-N)    1.00000          0.00000
(NOT-CHM NOT-GIB NSR-TIA-WON-N)     1.00000          0.00000
(NOT-CHM NOT-GIB NSR-STR-WOFF-N)    1.00000          0.00000
(NOT-CHM NOT-GIB NSR-STR-WON-N)     1.00000          0.00000
(NOT-CHM NOT-GIB AF-N-WOFF-N)       1.00000          0.00000
(NOT-CHM NOT-GIB AF-N-WON-N)        1.00000          0.00000
(NOT-CHM NOT-GIB AF-TIA-WOFF-N)     1.00000          0.00000
(NOT-CHM NOT-GIB AF-TIA-WON-N)      1.00000          0.00000
(NOT-CHM NOT-GIB AF-STR-WOFF-N)     1.00000          0.00000
(NOT-CHM NOT-GIB AF-STR-WON-N)      1.00000          0.00000

;;; Different output format for this large table.
> (distribution state/quinidine)
#<Distribution-Table #:G755
    Type:               DISCRETE
    Matrix: #:G755
         Distribution-Table #:G630

For OUTCOME: NSR-N-WOFF
NSR-N-WOFF |   (NO-STR-A NSR-A WARFOFF-A WARFOK-A ALIVE-A): 1.00000
```

```
For OUTCOME: NSR-N-WON
NSR-N-WON |    (NO-STR-A NSR-A WARFON-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: NSR-N-WNE
NSR-N-WNE |    (NO-STR-A NSR-A WARFON-A WARFNE-A ALIVE-A): 1.00000
NSR-N-WNE |    (NO-STR-A NSR-A WARFOFF-A WARFNE-A ALIVE-A): 1.00000

For OUTCOME: NSR-TIA-WOFF
NSR-TIA-WOFF |  (TIA-A NSR-A WARFOFF-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: NSR-TIA-WON
NSR-TIA-WON |  (TIA-A NSR-A WARFON-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: NSR-TIA-WNE
NSR-TIA-WNE |  (TIA-A NSR-A WARFON-A WARFNE-A ALIVE-A): 1.00000
NSR-TIA-WNE |  (TIA-A NSR-A WARFOFF-A WARFNE-A ALIVE-A): 1.00000

For OUTCOME: NSR-STR-WOFF
NSR-STR-WOFF |  (STR-A NSR-A WARFOFF-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: NSR-STR-WON
NSR-STR-WON |  (STR-A NSR-A WARFON-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: NSR-STR-WNE
NSR-STR-WNE |  (STR-A NSR-A WARFON-A WARFNE-A ALIVE-A): 1.00000
NSR-STR-WNE |  (STR-A NSR-A WARFOFF-A WARFNE-A ALIVE-A): 1.00000

For OUTCOME: AF-N-WOFF
AF-N-WOFF |    (NO-STR-A AFIB-A WARFOFF-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: AF-N-WON
AF-N-WON |     (NO-STR-A AFIB-A WARFON-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: AF-N-WNE
AF-N-WNE |     (NO-STR-A AFIB-A WARFON-A WARFNE-A ALIVE-A): 1.00000
AF-N-WNE |     (NO-STR-A AFIB-A WARFOFF-A WARFNE-A ALIVE-A): 1.00000

For OUTCOME: AF-TIA-WOFF
AF-TIA-WOFF |  (TIA-A AFIB-A WARFOFF-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: AF-TIA-WON
AF-TIA-WON |   (TIA-A AFIB-A WARFON-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: AF-TIA-WNE
AF-TIA-WNE |   (TIA-A AFIB-A WARFON-A WARFNE-A ALIVE-A): 1.00000
AF-TIA-WNE |   (TIA-A AFIB-A WARFOFF-A WARFNE-A ALIVE-A): 1.00000

For OUTCOME: AF-STR-WOFF
AF-STR-WOFF |  (STR-A AFIB-A WARFOFF-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: AF-STR-WON
AF-STR-WON |   (STR-A AFIB-A WARFON-A WARFOK-A ALIVE-A): 1.00000

For OUTCOME: AF-STR-WNE
AF-STR-WNE |   (STR-A AFIB-A WARFON-A WARFNE-A ALIVE-A): 1.00000
AF-STR-WNE |   (STR-A AFIB-A WARFOFF-A WARFNE-A ALIVE-A): 1.00000

For OUTCOME: DEAD
DEAD |         DEAD-A: 1.00000
```

C.5 The Transition Values

The following Common Lisp code is used to specify the transition values once the transitions are defined. The transition values constitute the bonus components b_{ij} in the value functions $v_i(.)$. The transition value definitions can be subject to sensitivity analysis as other numerical parameters. The definitions are as follows:

- Short term morbidities are defined for stroke, transient ischemic attack, and bleeding.
- If a transition is made to a different state where any of the above events are indicated, the short term morbidities are deducted accordingly.
- If a transition is made to the same state with any of the above events except stroke, the corresponding short term morbidity multiplied by a small factor of 0.05 is deducted.
- If a transition is made to the same state with stroke, the corresponding short term morbidity multiplied by a slightly larger factor of 0.25 is deducted. We assume that a stroke has more serious consequences than the other events.

```
;;; Full negative bonus for first indication. Factored negative bonus for repeated
;;; indications.

(defun load-transition-values5 (action)
  (let ((n-set (list nsr-n-woff nsr-n-won af-n-woff af-n-won))
        (tia-set (list nsr-tia-woff nsr-tia-won af-tia-woff af-tia-won ))
        (str-set (list nsr-str-woff nsr-str-won af-str-woff af-str-won))
        (n-ne-set (list nsr-n-wne af-n-wne))
        (tia-ne-set (list nsr-tia-wne af-tia-wne))
        (str-ne-set (list nsr-str-wne af-str-wne)))

    (set-transition-values action n-set n-ne-set (- stmGIB))

    (set-transition-values action n-set tia-set (- stmTIA))
    (set-transition-values action n-set tia-ne-set
                                (- (+ stmTIA stmGIB)))
    (set-transition-values action n-set str-set
                                (- (- stmSTROKE (- 1 Ustroke))))
    (set-transition-values action n-set str-ne-set
                                (- (+ (- stmSTROKE (- 1 Ustroke)) stmGIB)))

    (set-transition-values action n-ne-set n-ne-set (- (* 0.05 stmGIB)))

    (set-transition-values action n-ne-set tia-ne-set
                                (- (+ stmTIA (* 0.05 stmGIB))))
    (set-transition-values action n-ne-set str-ne-set
                                (- (+ (- stmSTROKE (- 1 Ustroke))
                                              (* 0.05 stmGIB))))

    (set-transition-values action tia-set tia-set (- (* 0.05 stmTIA)))
    (set-transition-values action tia-set tia-ne-set
                                (- (+ (* 0.05 stmTIA) stmGIB)))
    (set-transition-values action tia-set str-set
                                (- (- stmSTROKE (- 1 Ustroke))))
    (set-transition-values action tia-set str-ne-set
                                (- (+ (- stmSTROKE (- 1 Ustroke)) stmGIB)))
```

```
(set-transition-values action tia-ne-set tia-ne-set
                           (- (+ (* 0.05 stmTIA) (* 0.05 stmGIB))))
(set-transition-values action tia-ne-set str-ne-set
                           (- (+ (- stmSTROKE (- 1 Ustroke))
                                 (* 0.05 stmGIB))))
(set-transition-values action str-set str-set
                           (- (* 0.25 (- stmSTROKE (- 1 Ustroke)))))
(set-transition-values action str-set str-ne-set
                           (- (+ (* 0.25 (- stmSTROKE (- 1 Ustroke)))
                                 stmGIB)))

(set-transition-values action str-ne-set str-ne-set
                           (- (+ (* 0.25 (- stmSTROKE (- 1 Ustroke)))
                                 (* 0.05 stmGIB))))))
```

C.6 Solution and Analysis

Recall that the two clinical questions addressed in the case study are:

Problem 1: Quinidine decreases the proportion of time that the patient spends in AF. *Does this decrease his risk of embolic complications enough to justify the increased risk of sudden death?*

Problem 2: What is the optimal course of treatment in the next five years, taking into account the varying relative risk of bleeding for warfarin with respect to the duration of treatment?

Solution and Analysis of Long-Term Discriminatory Problem

The solution produced by the Markov value iteration solver for discriminatory problems is a set of two policies corresponding to the four strategies considered; each policy includes the expected value achievable in all possible starting states for all possible decision stages. Assuming that the patient is initially in either state NSR-N-WOFF or state AF-N-WOFF. He has a probability of 0.25 to be in atrial fibrillation, with no thromboembolization history, and not on warfarin at the beginning of the decision horizon. The solution to the first clinical question is detailed in Table C.2. The solution suggests that Quinidine does not decrease the risk of embolic complications enough to justify the increased risk of sudden death. Although not shown here, similar answers with respect to different starting states can also be directly derived from the policies information produced by the DYNAMO solver.

Table C.2 Solution to long-term discriminatory problem in case study.

Strategy	QALY (600 stages)	Relative Value
None	22.2664	-0.3594
Warfarin	22.6258	0
Quinidine	19.3532	-3.2726
Both	19.3781	-3.2477

Sensitivity analysis on some of the numerical parameters shows that the Warfarin strategy dominates except in the following conditions.

The None strategy, *i.e.,* administer warfarin only when indicated and stop when necessary, becomes the preferred choice under three circumstances: First, when the *therapeutic efficacy* of warfarin on preventing thromboembolism falls below 47%, where the efficacy is calculated by:

$$Efficacy = 1 - Relative\ risk\ .$$

Second, when the complication risks associated with warfarin are very serious, as when the relative risk of bleeding exceeds 3.76 (50% increase from base case). the risk of cerebral hemorrhage exceeds 0.0017 (200%), and the risk of gastro-intestinal bleeding exceeds 0.015 (150%). Third, when the risk of thromboembolization is very low, as when the rate of thromboembolization falls below 0.0068 (27% decrease from base case).

The Quinidine strategy becomes the preferred choice only when the excess mortality rate for Quinidine falls below 0.0005 (96% decrease from base case).

Solution and Analysis of Short-Term Optimization Problem

The solution produced by the semi-Markov value iteration solver for optimization problems is an optimal policy indicating the optimal expected value achievable in all possible starting states for all possible decision stages. Assuming that the patient is initially in either state NSR-N-WOFF or state AF-N-WOFF. He has a probability of 0.25 to be in atrial fibrillation, with no thromboembolization history, and not on warfarin at the beginning of the decision horizon. The solution to the second clinical question is partially detailed in Table C.3. In the table, the unit of the decision stages are in months. The notation “N” indicates the None strategy; the notation “W” indicates the Warfarin strategy. Recall that the decision stage 60 is at time 0.

The solution shows that the strategy that administers only warfarin initially is the preferred strategy up to 8 months. After 8 months, if the patient remains in the same condition, the strategy that does not administer any drug initially is preferred. Again, although not shown here, similar answers with respect to different starting states can also be directly derived from the policies information produced by the DYNAMO solver. This would allow an optimal treatment course be derived according to the varying condition of the patient, *e.g.,* what is the best treatment strategy a year from now, if he suffers a stroke in the first 3 months and a bleeding episode in the seventh month?

Table C.3 Partial solution to short-term optimization problem.

Stage	QALY/ Strategy		Stage	QALY/ Strategy	
5	0.37389	N	35	2.80835	N
10	0.78636	N	40	3.20565	N
15	1.19594	N	45	3.60087	N
20	1.60278	N	50	3.99412	N
25	2.00703	N	55	4.39177	W
30	2.40885	N	60	4.78974	W

Sensitivity analysis on some of the numerical parameters shows that the Warfarin strategy becomes the preferred strategy over more decision stages when the risk of thromboembolization increases or when the relative risk of bleeding for warfarin decreases. The same pattern is also observed with waiting time functions for the states affected by warfarin that lessen its adverse effects. In the base case, for instance, a patient is more likely to leave his current state sooner, when the probability of bleeding due to warfarin is higher, than later, when the probability of bleeding due to warfarin is lower. The adverse effects of warfarin are lessened if the patient is equally likely to leave his current state sooner or later.

D Glossary

This glossary contains a list of simplified medical concepts from [Bantam, 1990] and [Taber, 1989]. Cross-references within the glossary are in *italic*.

Angina
A sense of suffocation or suffocating pain.

Antiarrhythmic agent
A drug that controls or prevents cardiac arrhythmias or irregular heartbeats.

Anticoagulant
An agent that prevents the clotting of blood. It is used to prevent the formation of blood clots or to break up clots in blood vessels in conditions such as *thrombosis* and *embolism*.

Atherosclerosis
A disease of the arteries in which fatty plaques develop on their inner walls, with eventual obstruction of blood flow.

Cerebral hemorrhage
Bleeding from a cerebral artery into the tissue of the brain.

Chronic ischemic heart disease
Heart disease of gradual onset that limits blood supply to the heart muscles.

Coronary artery bypass graft
A surgical procedure that establishes a shunt that permits blood to travel from the aorta to a branch of the coronary artery at a point past an obstruction.

Embolism
The condition in which a mass of material such as blood clot becomes lodged in an artery and obstructs its blood flow.

Gangrene
Death of tissue, usually due to deficient or absent blood supply.

Gastrointestinal bleeding
Bleeding in the stomach and the intestines.

Myocardial infarction
Heart attack. Death of a segment of heart muscle, which follows interruption of its blood supply.

Percutaneous transluminal coronary angioplasty
A method of treating localized coronary artery narrowing by applying a special catheter with a cylindrical balloon that surrounds a portion of it. Inflation of the balloon dilates the narrowed vessel.

Paroxysmal atrial fibrillation
A kind of cardiac arrhymia or abnormal heartbeat of sudden onset. It results from irregular and rapid randomized contractions of the atria working independently of the ventricles.

Restenosis
The reccurrence of a stenosis or the condition of constriction or narrowing of a passage or orifice.

Quinidine
An *antiarrhythmic agent* that slows down the activity of the heart and is administered by mouth to control abnormal and increased heart rhythm.

Sinus rhythm
The normal cardiac rhythm or heartbeat rate.

Thromboembolism
An *embolism*; the blocking of a blood vessel by a blood clot or thrombus that has become detached from its site of formation.

Thrombosis
The formation, development, or existence of a blood clot or thrombus within the vascular system.

Warfarin
An *anticoagulant* used mainly to treat coronary or venous *thrombosis* to reduce the risk of *embolism*. It is given by mouth or injection.

References

Bantam (1990). The bantam medical dictionary, revised edition. Bantam Books.

Beck, J. R. and Pauker, S. G. (1983). The Markov process in medical prognosis. *Medical Decision Making*, 3:419–458.

Bellman, R. A. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Bertsekas, D. P. (1987). *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Englewood Cliffs, NJ.

Berzuini, C., Bellazzi, R., and Quaglini, S. (1989). Temporal reasoning with probabilities. In *Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence*, pages 14–21. Association for Uncertainty in Artificial Intelligence.

Breese, J. S. (1992). Construction of belief and decision networks. *Computational Intelligence*, 8:624–647.

Chan, B. Y. and Shachter, R. D. (1992). Structural controllability and observability in influence diagrams. In Dubois, D., Wellman, M. P., D'Ambrosio, B. D., and Smets, P., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Eighth Conference*, pages 25–32, San Mateo, CA. Morgan Kaufmann.

Chapman, D. (1987). Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377.

Clemen, R. T. (1991). *Making Hard Decisions: An Introduction to Decision Analysis*. PWS-KENT, Boston, MA.

Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405.

Dean, T. (1994). Decision-theoretic planning and markov decision processes. Submitted for publication.

Dean, T., Kaelbling, L. P., Kirman, J., and Nicholson, A. (1993a). Deliberation scheduling for time-critical sequential decision making. In Heckerman, D. and Mamdami, A., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 309–316, San Mateo, CA. Morgan Kaufmann.

Dean, T., Kaelbling, L. P., Kirman, J., and Nicholson, A. (1993b). Planning with deadlines in stochastic domains. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 574–579.

Dean, T., Kriman, J., and Kanazawa, K. (1992). Probabilistic network representations of continuous-time stochastic processes for applications in planning and control. In *Proceedings of the First International Conference on AI Planning Systems*.

Dean, T. L. and Wellman, M. P. (1991). *Planning and Control*. Morgan Kaufmann, San Mateo, CA.

Deardon, R. and Boutilier, C. (1994). Integrating planning and execution in stochastic domains. In *Uncertainty in Artificial Intelligence: Proceedings of the Tenth Conference*, San Mateo, CA. Morgan Kaufmann.

Draper, D., Hanks, S., and Weld, D. (1994). A probabilistic model of action for least-commitment planning with information gathering. In *Uncertainty in Artificial Intelligence: Proceedings of the Tenth Conference*.

Drummond, M. and Bresina, J. (1990). Ayntime synthetic projection: Maximizing the probability of goal satisfaction. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 138–144.

DT-Planning (1994). *Proceedings of AAAI Spring Symposium on Decision-Theoretic Planning*. American Association for Artificial Intelligence.

Egar, J. W. and Musen, M. A. (1993). Graph-grammar assistance for automated generation of influence diagrams. In Heckerman, D. and Mamdami, A., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 235–242, San Mateo, CA. Morgan Kaufmann.

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

Fishman, M. C., Hoffman, A. R., Klausner, R. D., and Thaler, M. S. (1985). Cardiac arrhythmias. In *Medicine*, chapter 4, pages 29–39. J. B. Lippincott Company, Philadelphia, PA.

Genesereth, M. R. and Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA.

Goldman, R. P. and Charniak, E. (1990). Dynamic construction of belief networks. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 90–97.

Haddawy, P. and Hanks, S. (1990). Issues in decision-theoretic planning: Symbolic goals and numeric utilities. In *Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control*, pages 48–58.

Haddawy, P. and Hanks, S. (1992). Representations for decision-theoretic planning: Utility functions for deadline goals. In Nebel, B., Rich, C., and Swartout, W., editors, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 71–82, San Mateo, CA. Morgan Kaufmann.

Hanks, S. (1990). Practical temporal projection. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 158–163. American Association for Artificial Intelligence.

Hazen, G. B. (1992). Stochastic trees: A new technique for temporal medical decision modeling. *Medical Decision Making*, 12:163–178.

Heyman, D. P. and Sobel, M. J. (1984). *Stochastic Models in Operations Research: Stochastic Optimization*, volume 2. McGraw-Hill.

Hollenberg, J. P. (1984). Markov cycle trees: A new representation for complex Markov processes. *Medical Decision Making*, 4(4). Abstract from the Sixth Annual Meeting of the Society for Medical Decision Making.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA.

Howard, R. A. (1971). *Dynamic Probabilistic Systems*, volume 1 & 2. John Wiley and Sons, New York.

Howard, R. A. (1988). Decision analysis: Practice and promise. *Management Science*, 34:679–695.

Howard, R. A. and Matheson, J. E. (1984). Influence diagrams. In Howard, R. A. and Matheson, J. E., editors, *The Principles and Applications of Decision Analysis*, volume 2, pages 719–762. Strategic Decisions Group, Menlo Park, CA.

Janssen, J., editor (1986). *Semi-Markov Models: Theory and Applications*. Plenum Press, New York, NY.

Jewell, W. S. (1963). Markov renewal programming: I. formulations, finite return models II. infinite return models, example. *Operations Research*, 11:938–971.

Kassirer, J. P., Moskowitz, A. J., Lau, J., and Pauker, S. G. (1987). Decision analysis: A progress report. *Annals of Internal Medicine*, 106:275–291.

Knoblock, C. A. (1991). Search reduction in hierarchical problem solving. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 686–691.

Korf, R. E. (1987). Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88.

Kushmerick, N., Hanks, S., and Weld, D. (1994). An algorithm for probabilistic least-commitment planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*. American Association for Artificial Intelligence.

Leong, T.-Y. (1993). Dynamic decision modeling in medicine: A critique of existing formalisms. In *Proceedings of the Seventeenth Annual Symposium on Computer Applications in Medical Care*, pages 478–484. IEEE.

Lifschitz, V. (1986). On the semantics of sc strips. In Georgeff, M. P. and Lansky, A. L., editors, *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, pages 1–9. Morgan Kaufmann.

Mansell, T. M. (1993). A method for planning given uncertain and incomplete information. In Heckerman, D. and Mamdani, A., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 350–358, San Mateo, CA. Morgan Kauffman.

Meyer, B. (1990). *Introduction to the Theory of Programming Languages*. Prentice Hall.

Myers, B. A., Giuse, D. A., Dannenberg, B. V. Z., Kosbie, D. S., Pervin, E., Mickish, A., and Marchal, P. (1990). Garnet: Comprehensive support for graphical, highly-interactive user interfaces. *IEEE Computer*, 23(11):71–85.

Neapolitan, R. E. (1993). Computing the confidence in a medical decision obtained from an influence diagram. *Artificial Intelligence in Medicine*, 5:341–363.

Newell, A., Shaw, J. C., and Simon, H. A. (1960). Report on a general problem-solving program. In *Proceedings of the International Conference on Information Processing*, pages 256–264. UNESCO.

Newell, A. and Simon, H. A. (1972). *Human Problem Solving*. Prentice-Hall.

Pauker, S. G. and Kassirer, J. P. (1987). Medical progress: Decision analysis. *New England Journal of Medicine*, 316:250–258.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

Provan, G. M. (1992). Modeling the evolution of acute abdominal pain using temporal influence diagrams. Technical Report MIS-CS-92-69, University of Pennsylvania, Dept. of Computer and Information Science.

Provan, G. M. and Clarke, J. R. (1993). Dynamic network construction and updating techniques for the diagnosis of acute abdominal pain. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):299–307.

Provan, G. M. and Poole, D. (1991). A utility-based analysis of consistency-based diagnosis. In Allen, J., Fikes, R., and Sandewall, E., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference (KR'91)*, pages 461–472, San Mateo, CA. Morgan Kaufmann.

Pugh, A. L. (1970). *DYNAMO II Users' Manual*. MIT Press, Cambridge, MA.

Puterman, M. L. (1990). Markov decision processes. In Heyman, D. P. and Sobel, M. J., editors, *Handbooks in Operations Research and Management Science, volume 2: Stochastic Models*, chapter 8, pages 331–434. Elsevier Science Publishers B. V.(North-Holland).

Raiffa, H. (1968). *Decision Analysis: Introductory Lectures on Choices Under Uncertainty.* Addison-Wesley, Reading, MA.

Rosenschein, S. J. (1981). Plan synthesis: A logical perspective. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence,* pages 331–337.

Sacerdoti, E. D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135.

Sacerdoti, E. D. (1975). The nonlinear nature of plans. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pages 206–214.

Shachter, R. D. (1986). Evaluating influence diagrams. *Operations Research*, 34:871–882.

Shachter, R. D. and Peot, M. A. (1992). Decision making using probabilistic inference methods. In Dubois, D., Wellman, M. P., D'Ambrosio, B. D., and Smets, P., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Eighth Conference*, pages 276–283. Morgan Kaufmann.

Shapley, L. S. (1953). Stochastic games. *Proceedings of the National Academy of Science*, 39:1095–1100.

Sonnenberg, F. A. and Beck, J. R. (1993). Markov models in medical decision making: A practical guide. *Medical Decision Making*, 13(4):322–338.

Srinivas, S. (1992). Generalizing the noisy or model to n-ary variables. Technical Memorandum 79, Rockwell International Science Center, Palo Alto Laboratory, Palo Alto, CA.

Stefik, M. (oopsort1981a}1981). Planning with constraints (sc molgen: Part 1). *Artificial Intelligence,* 16:111–140.

Taber (1989). Taber's cyclopedic medical dictionary, edition 16. F. A. Davis Company.

Tate, A. (1977). Generating project networks. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 888–893.

Tate, A., Hendler, J., and Drummond, M. (1990). A review of ai planning techniques. In Allen, J., Hendler, J., and Tate, A., editors, *Readings in Planning*, pages 26–49. Morgan Kaufmann, San Mateo, CA.

Tatman, J. A. and Shachter, R. D. (1990). Dynamic programming and influence diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):365–379.

Taylor, H. M. and Karlin, S. (1994). *An Introduction to Stochastic Modeling*. Academic Press, revised edition edition.

Wellman, M. P. (1990). *Formulation of Tradeoffs in Planning Under Uncertainty.* Pitman and Morgan Kaufmann.

Wellman, M. P., Breese, J. S., and Goldman, R. P. (1992). From knowledge bases to decision models. *The Knowledge Engineerting Review,* 7(1):35–53.

Wellman, M. P. and Doyle, J. (1991). Preferential semantics for goals. In *Proceedings of the National Conference on Artificial Intelligence*, pages 698–703.

Wellman, M. P. and Doyle, J. (1992). Modular utility representation for decision-theoretic planning. In *Proceedings of the First International Conference on AI Planning Systems*.

Willard, K. E. and Critchfield, G. C. (1986). Probabilistic analysis of decision trees using symbolic algebra. *Medical Decision Making*, 6.

Wong, J. B., Sonnenberg, F. A., Salem, D. M., and Pauker, S. G. (1990). Myocardial revascularization for chronic stable angina: Analysis of the role of percutaneous transluminal coronary angioplasty based on data available in 1989. *Annals of Internal Medicine*, 113(11):852–871.

Yeh, S. S. (1994). Constructing decision models for diabetes from clinical data. Master's thesis, MIT, Cambridge, MA.

Yeh, S. S. and Leong, T.-Y. (1994). Automatic generation of transition probabilities in dynamic decision modeling: A case study. In *Proceedings of the AAAI Spring Symposium on Artificial Intelligence in Medicine*.