CONSTRUCTION OF NORMATIVE DECISION MODELS USING ABSTRACT GRAPH GRAMMARS

A DISSERTATION

SUBMITTED TO THE PROGRAM IN MEDICAL INFORMATION SCIENCES

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

John W. Egar

March 1994

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Mark A. Musen
(Principal Adviser)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Ross D. Shachter
(Engineering–Economic Systems)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Edward H. Shortliffe
(Medicine)

Approved for the University Committee on Graduate Studies:

Dean of Graduate Studies

Abstract

This dissertation addresses automated assistance for decision analysis in medicine. In particular, I have investigated graph grammars as a representation for encoding how decision-theoretic models can be constructed from an unordered list of concerns. The modeling system that I have used requires a standard vocabulary to generate decision models; the models generated are qualitative, and require subsequent assessment of probabilities and utility values. This research has focused on the modeling of the qualitative structure of problems *given* a standard vocabulary and *given* that subsequent assessment of probabilities and utilities is possible. The usefulness of the graph-grammar representation depends on the graph-grammar formalism's ability to describe a broad spectrum of qualitative decision models, on its ability to maintain a high quality in the models it generates, and on its clarity in describing topological constraints to researchers who design and maintain the actual grammar. I have found that graph grammars can be used to generate automatically decision models that are comparable to those produced by decision analysts.

Acknowledgments

I am most grateful to Mark Musen, Ross Shachter, and Edward Shortliffe for their support and guidance. Henrik Eriksson supplied the original inspiration for this research, and he has always provided me with sage technical and theoretical advice. I thank Ingo Beinlich, John Breese, Keith Campbell, Robert Goldman, David Heckerman, Max Henrion, Eric Horvitz, Brian Kan, David Kinny, Harold Lehmann, Tse-Yun Leong, Blackford Middleton, Malcolm Pradhan, Angel Puerta, Geoff Rutledge, Yuval Shahar, Jaap Suermondt, Maria Tovar, Michael Walker, and Michael Wellman for their valuable discussions concerning my dissertation research. Jay Strain helped me with surgical terminology. The students, staff, and faculty of the Section on Medical Informatics provided me with abundant intellectual stimulation and comradery, which have kept me excited and sane over the past four years. I am especially grateful to Lyn Dupré for editing previous drafts of this dissertation, and for helping me to write far more clearly than I did four years ago.

I conducted this work with the support of the National Library of Medicine under grants LM-05157, LM-07033, and LM-05208. This work was also supported by computing resources provided by the Stanford University CAMIS project, which is funded under grant number LM-05305 from the National Library of Medicine.

Contents

List of Tables

List of Figures

Chapter 1

Models for Medical Decisions

Who shall decide, when doctors disagree,
And soundest casuists doubt, like you and me?

—*Alexander Pope, Ep. iii. To Lord Bathurst (1733)*

Over the centuries, physicians have had to weigh the potential benefits of their interventions against the risks. Also, in a world where medical resources are limited, costs to the patient and to society are important considerations. For decisions that are simple and that generalize across most patients, the best available therapeutic plan may be obvious. However, uncertain information is used in most of medicine, and the relative importance of different risks and potential benefits may depend on the individual patient's preferences and on their unique situation. Moreover, the effect of a decision on the patient involved can be momentous. Consequently, physicians, patients, and health-care workers may find it difficult to reach a decision, even when they are able to list the important considerations that need to be weighed.

To help physicians, patients, and other people make the best decision with the available information, decision analysts can clarify the issues involved, weigh the considerations, and derive ramifications of those beliefs that the decision makers find relatively easy to express. Such professional decision analysis is both expensive and time consuming. One aim of research in medical informatics is to provide automated assistance for decision analysis. The subject of my research is the automation of initial decision modeling. It is my hope that systems like the one described in this dissertation might someday help health-care workers and their patients to formulate decision problems, and thereby to find the wisest path through their difficult dilemmas.

1.1 Example Dilemma

As an example, suppose that a physician suspects that her patient has active tuberculosis, but has been unable to establish the diagnosis—all cultures taken from the patient's sputum and gastric aspirate have failed to show the causative bacteria. She is considering bronchoscopy to obtain additional samples for culturing, and she is also considering treatment for presumptive tuberculosis. However, both she and her patient wish to avoid any unnecessary testing and any unnecessary treatment because of the costs, risks, and nuisance involved. Both the physician and the patient must somehow weigh the likelihood of the disease, the seriousness of the disease, the benefit of the treatment, the costs of the treatment, the costs of the bronchoscopy, the risks of bronchoscopy, and the usefulness of the bronchoscopy in providing more information about the probability that the patient has tuberculosis. Since different patients may weigh the costs, risks, and benefits of a given medical procedure differently, a universal policy may not be appropriate for individual decisions. Our hypothetical physician and patient must somehow reach a decision for this particular case. How do they decide?

1.2 Decision Analysis

Decision theory addresses how an individual can weigh probable outcomes from alternatives to select the alternative—or the sequence of alternatives—that would tend to result in the most propitious outcome for that individual (von Neumann and Morgenstern, 1944; North, 1968). Decision theory is based on four axioms of rational[1] decision making:

1. The decision maker can established a complete ordering of preferences among outcomes from the best—most desirable—to the worst.

2. The decision maker can assign preferences to lotteries just as well as to outcomes.

[1] Decisions that are made in accordance with these axioms are also called **normative.**

3. Lotteries have no intrinsic value to the decision maker beyond their outcomes.

4. For any three outcomes, there exists a probability p such that the decision maker is indifferent between receiving the intermediate outcome and receiving a lottery with a p chance of the best outcome and a $1-p$ chance of the worst outcome.

From these assumptions, decision analysts have developed myriad tools to help people clarify decisions, weigh uncertainties, assess the desirability of different outcomes, test decision models, and derive decisions that are optimal according to the axioms. Although decision theory has been available for decades, and weighing risks and benefits of tests and interventions is a common task in everyday clinical medicine, decision analysis is not commonly used in medicine.

1.3 Impediments to Decision Modeling

Decision analysis typically requires that expectations, evidence, deductions, and conclusions about the state of the world be expressed as probabilities. Early research in applying probabilistic reasoning to medical decision making has relied on simplifications to the underlying model. For example, Warner and colleagues (1964) made the assumption that, for any given disease, the chance of finding a particular sign or symptom to be present is independent of the presence or absence of all other signs and symptoms. This simplification—called **conditional independence**—and other simplifications made the computation of probabilities easier, but either restricted the decision analysis to simple situations or forced the decision analyst to make unwarranted assumptions.

More recent research in probabilistic reasoning has led to the development of computational tools that deal with more complex models of conditional dependence (Pearl, 1988; Olesen et al., 1989; Beinlich et al., 1989; Heckerman and Nathwani, 1992). One way to represent the interrelationships among variables in a decision problem is to use an **influence diagram** (Howard and Matheson, 1984). An influence diagram is a directed acyclic graph in which the nodes represent either decisions to be made or uncertain events, and the arcs represent either information available at the time of a decision or the dependence of an event's probability on the state of another event or decision (see Section 3.5).

In our hypothetical case of a physician considering bronchoscopy, the decision maker must be able to draw a structural model of probabilistic dependencies, such as the model shown in Figure 1.1. Here, two square nodes—labeled Bronchoscopy and Tb treatment—represent the two decisions that must be made. The three circular nodes—labeled Bronchoscopy result, Tb, and Future Tb—represent chance events whose outcomes are relevant to the decision problem. The hexagonal node—labeled Value to patient—represents the deterministic **utility** function that computes the overall desirability of any combination of outcomes. As indicated by the arcs entering the utility node, Value to patient is a function of Tb and Future Tb outcomes, and of the two decisions, Bronchoscopy and Tb treatment. Similarly, the solid arcs with the target node Future Tb indicate that the probability distribution for the Future Tb random variable depends on the outcome of Tb and the alternative chosen for Tb treatment. The dashed arcs indicate that the outcome or the chosen alternative of the variable represented by the source node for the arc is known to the decision maker at the time that the target node's decision must be made.

Several computer programs (Shachter and Bertrand, 1987; Henrion, 1992; Applied Decision Analysis, 1992; Beinlich and Herskovits, 1990; Andersen et al., 1989; Cousins et al., 1992; Srinivas and Breese, 1990) accept an influence-diagram model from the user, and then perform computations to determine which decision alternatives are normatively preferred.[2] Other computer-based tools assist the user in further development and evaluation of the decision model using techniques such as probability assessment (Heckerman, 1991), sensitivity analysis (Howard, 1984b; Laskey, 1993), preference assessment (Farr and Shachter, 1992), value-of-information evaluation (Howard, 1984a), exact or approximate inference (Shachter and Peot, 1992; Chavez and Cooper, 1990; Jensen et al., 1990; Dagum and Galper, 1993), and explanation (Suermondt and Cooper, 1992).

[2]Although several of these programs are designed to answer queries about **belief-networks**—influence diagrams that lack decision nodes, such routines can be used to solve influence-diagram queries as well (Cooper, 1988).

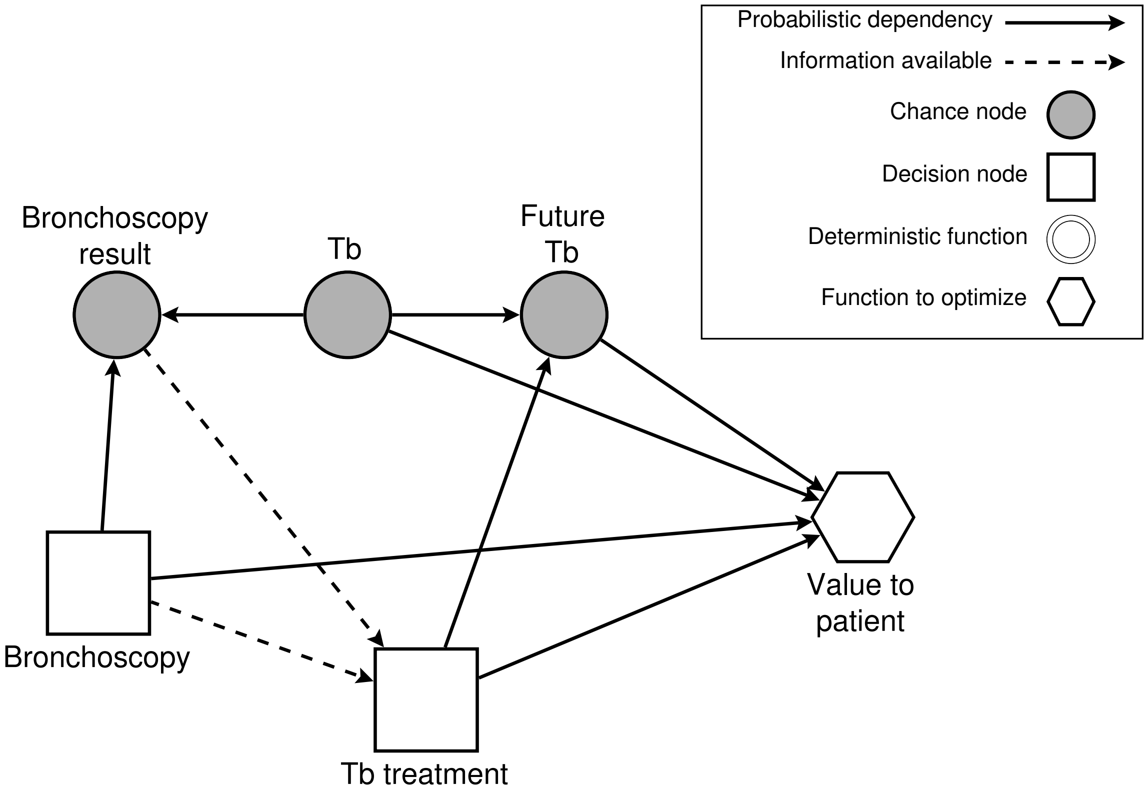

Figure 1.1: An influence-diagram model for the tuberculosis decision. The decision maker must decide whether perform a diagnostic test, and whether to begin treatment. Note that, after the Bronchoscopy, there is a future decision—Tb treatment—that depends on the results of the test.

Despite the technologies that assist with reasoning about complex probabilistic models, the decision maker must be able to formulate such models. Unfortunately, drawing the structure of the influence diagram for a problem is cognitively difficult, and an indefinitely large variety of models is needed to represent all the problems facing physicians. Moreover, clinical decision problems are often far more complex than is our example problem in Figure 1.1. Since the decision model is patient specific, there generally are no data on which statistical classification and machine-learning techniques (Herskovits and Cooper, 1990) might capitalize to derive a model. Moreover, the decision variables must be included in the model, and clinical databases generally lack full coverage of decision alternatives. Although some researchers have suggested a comprehensive template to cover all decision models (Heckerman and Horvitz, 1990), it is difficult to imagine a single topology that could contain all possible problem structures. Finally, decision analysts frequently work iteratively, adding and deleting nodes and arcs to refine the model. Neither the machine-learning approach nor the comprehensive-template approach is designed for such incremental modeling.

1.4 Automated Generation of an Initial Model

We must decrease the difficulty of initial model formulation if we are to encourage normative decision making in the clinical environment, where patients and physicians must make decisions in minutes, lacking the luxury of weeks to undertake decision analysis for a single problem. One hope for automated assistance lies in the observation that there exist topological patterns in influence diagrams that occur repeatedly in widely disparate medical dilemmas.[3] For example, there is a clinical maxim that an invasive test should not be ordered unless future treatment decisions might be changed based on the results of that test. Such a maxim does not rely on medical expertise or on the details of a particular clinical case. The maxim also has a topological equivalent in an influence diagram. In Figure 1.1, we see that the decision node for Bronchoscopy precedes the decision node for Tb treatment. Also, the result of the test provides information that becomes available to the treatment decision node.

In addition to typical topological patterns, each clinical concept tends to be associated with a typical role that it plays in a decision model. Tests and treatments tend to be decision nodes, which represent variables under the control of the decision maker. Results of tests tend to be observable chance nodes, where the corresponding stochastic variable's value is dependent on an uncertainty that the decision maker cannot evaluate directly. Treatments tend to be modeled as decreasing the probability or severity of some future disease state. Several other abstractions can be made that group medical concepts based on how they appear in influence-diagram models (Section 6.1). The key idea of my dissertation research is that these prototypical patterns and abstractions might help a computer to generate reasonable

[3]I suspect that these topological patterns reflect the ways that clinicians have classified observations and interventions; however, as discussed in Chapter 4, the patterns also reflect more general constraints inherent in decision theory.

influence-diagram models from a list of the concepts involved in the dilemma.

At this point, we would like to have a formalism for describing what the prototypical patterns are, and how they are to be connected to the rest of an evolving influence-diagram model. Fortunately, there exists a formalism that is specifically designed to encode such graph manipulations: the **graph-grammar** formalism. More precisely, there exist not one but dozens of graph-grammar formalisms (Ehrig et al., 1990). Although they differ in the class of graphs on which they operate, and in their ability to express different types of graph manipulations, all these approaches basically describe how a graph is to be altered. A **graph grammar** consists of **graph-rewrite production rules.**[4] Each production rule describes where in a host graph the manipulation can be applied, how the graph manipulation is to be performed, and what the effect of the manipulation is. The collection of graph-grammar rules describes a class of graphs that can be constructed from those rules. Consequently, one approach to constructing decision models automatically is to assign domain concepts—such as Bronchoscopy—to nodes and patterns that are typical for that concept.

The particular graph-grammar formalism that I have used is a modification of Göttler's operational graph grammar, suggested by Barthelmann (1990). This formalism can describe context-sensitive node manipulations of directed graphs with labeled nodes and arcs. The formalism has the advantage that the production rules

[4]Throughout this paper, we shall use the terms **production rule, rule,** and **production** interchangeably.

themselves can each be described by a single graph. Figure 1.2 provides a brief example of such a production rule and of its application.[5].

In Göttler's notation, a production is divided into four regions (Figure 3.3a): the left region (V_{L}), the right region (V_{R}), the region below (V_{B}), and the region above (V_{A}). Each production rule describes the following manipulation steps:

1. Locate all subgraphs in the host diagram where the nodes and arcs match the vertices and edges from the left and bottom regions of the production rule (Figure 3.3b).

2. Match zero or more subgraphs in the indeterminate region to subgraphs in the host diagram. Also match the edges between the indeterminate and left regions to corresponding arcs in the diagram.

3. Delete the nodes that matched V_{L} and delete their incident arcs (Figure 3.3c).

4. Add new nodes and arcs that correspond to the vertices and edges in the right region of the production rule (Figure 3.3d).

[5]Details of the graph-grammar formalism are discussed in Section 3.3

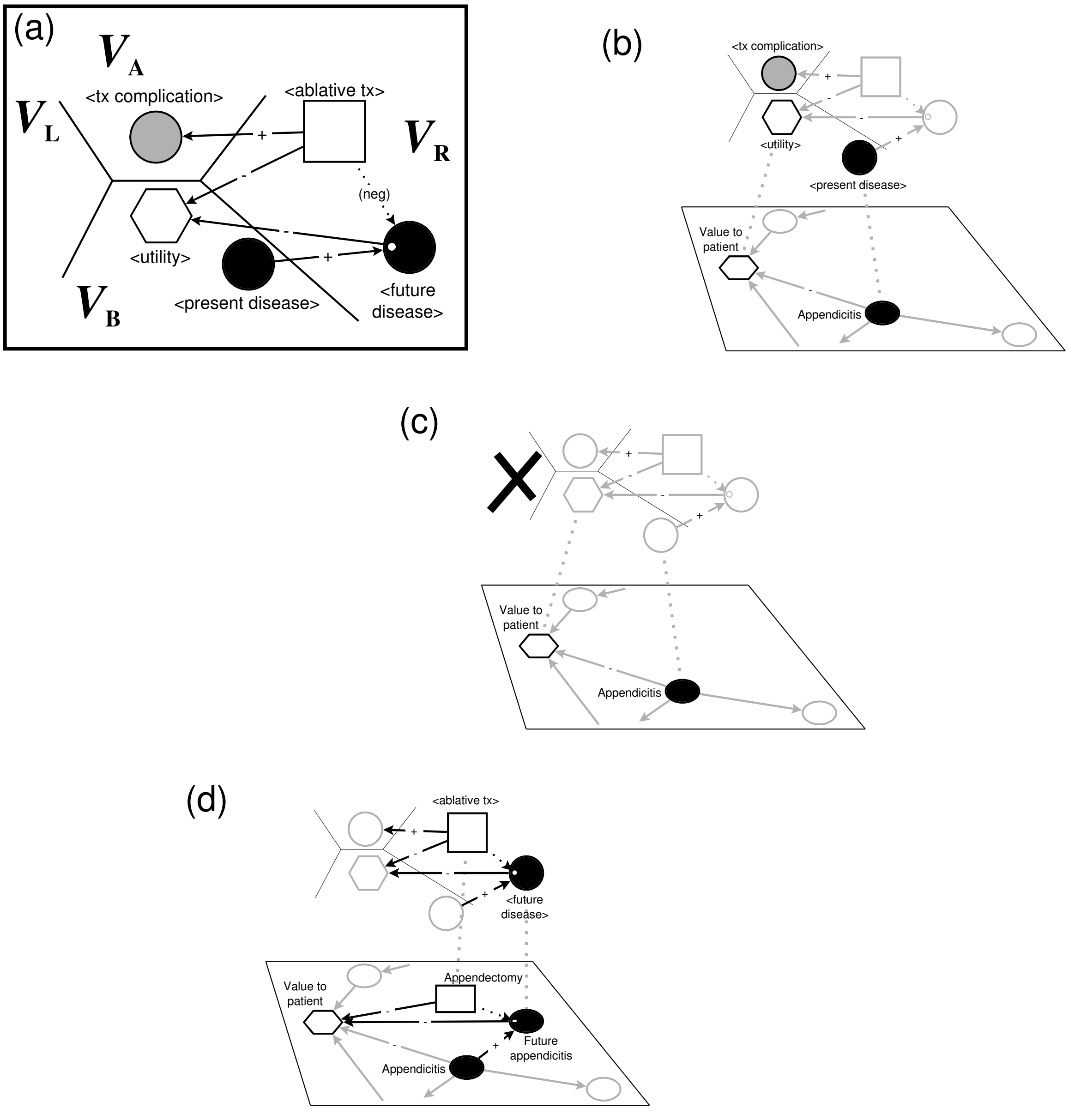

Figure 1.2: Sample application of the graph-grammar production rule. (a) This production rule describes how nodes of the type <ablative tx> can be added to the host graph. V_{L}, V_{A}, V_{B}, and V_{R} (left, above, below, and right) are the four regions of a graph-grammar production rule. (b) The first view of the host graph shows two nodes from the host diagram matching <utility> and <present disease> in the production. (c) If V_{L} contained vertices, a matching set of nodes would be removed. (d) Additional nodes Appendectomy and Future appendicitis are added to the QCID model. (tx = treatment; dotted arcs correspond to contingency arcs.)

1.5 Gramarye: Assistance for Modeling of Decisions

Influence diagrams represent a type of graph, and the prototypical patterns alluded to in the previous section can be viewed as subgraphs. Graph grammars are a formal system for building, inspecting, and changing a graph through subgraph manipulations. The prototypical patterns of medical influence diagrams can be represented as graph-grammar rules that introduce one or more nodes into an influence diagram. A derivation system might then use those patterns to develop an influence diagram from a list of nodes that the user desires to be in the diagram. Ideally, such a derivation system could follow the transactions in an electronic medical record, and, whenever the need for decision analysis arose, the system could produce a qualitative model for some context that the user defines. The qualitative model could then be used by the physician to direct further gathering of conditional probabilities and of value weights. In this dissertation, I address only the initial modeling of medical dilemmas. The basic result that I have found is this: *A graph-grammar derivation system can generate a reasonable qualitative decision model from an unordered list of medical concerns.*

I have implemented a graph-grammar derivation system, called **Gramarye,** which is a graph-grammar derivation system.[6] Gramarye converts a list of node labels into

[6] The derivation system runs under Common Lisp. I have also implemented a NeXTSTEP user interface for Gramarye.

a directed graph with labeled nodes and labeled arcs. As shown in Figure 1.3, it requires four static expressions to describe the domain and to describe the graphical modeling language on which it is operating:

1. An initial graph
2. A classification of node labels according to abstract symbols used in the productions
3. A collection of graph-grammar productions
4. A visual notation for each subclass of node

These four requirements are filled for influence-diagram models in the domain of medical decision making by

1. The utility node, Value to patient
2. A classification tree for a medical lexicon
3. A graph grammar for medical influence diagrams
4. The shapes **rectangle, circle,** and **hexagon** for **decision, chance,** and **utility** nodes, respectively

Once the domain and graphical knowledge representation are given, the system operates on lists of terms to generate graphical models. In this dissertation, I concentrate on the appropriate node-label classification and graph grammar for deriving

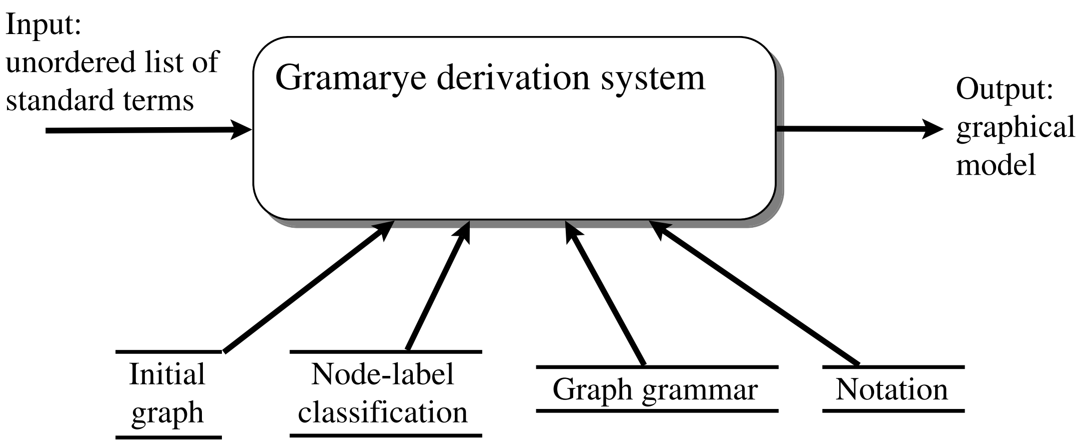

Figure 1.3: Static expressions used by Gramarye. At bottom are the four static expressions that define a graphical knowledge representation and a domain: the initial graph, the node-label classification, the graph grammar, and the notation.

qualitative influence diagrams in the domain of medicine.

Although I do not suggest that the derivation system I have built will by itself provide useful clinical decision support, a system that coordinated Gramarye with other information-gathering and decision-analysis tools could assist in clinical decision making and public-health management. In this idealized scenario, the modeling system would accept terms from a list of decisional considerations, and would classify the terms according to conceptual abstractions made in the graph grammar. The system would use the graph grammar to derive an initial influence diagram from those concepts. It then would glean probabilities from a database of patients, from an online repository of epidemiological information, or via a bibliographic search through the medical literature. It also would assign rough orders of importance to the various possible outcomes. For instance, the disutility of antituberculosis therapy would be, by default, less than the disutility of active pulmonary tuberculosis. The system would make further adjustments as sensitivity analysis and the user's direct inspection of the diagram showed disagreements between the decision model and the user's perception of the problem at hand. Then, as the more sensitive probabilities and utilities were uncovered in this analysis, the system and the user would assess these variables more accurately for the individual patient. From these direct assessments, the physician and her patient would compute a normative plan of action. Again, the derivation system discussed in this dissertation does not go beyond the first step: generating a qualitative model. The investigation of computer assistance for subsequent steps in the decision-analysis cycle is left to other researchers and is an active ongoing area of research.

1.6 Dissertation Overview

Chapter 2 describes related research and the historical background of this work. In Chapter 3, I discuss the Gramarye system. Chapter 4 investigates special properties that we might demand in reasonable decision models. In Chapter 5, I provide results gleaned from comparing the models generated by Gramarye to those constructed by decision analysts. I offer my appraisal of the derivation system, the graph grammar, and the system's vocabulary in Chapter 6. Appendices A and B explain in detail the graph-grammar and influence-diagram formalisms used in this research. Appendix C presents an actual transcript from a particularly complex derivation performed by Gramarye.

Chapter 2

Prior Work

"As to poetry, you know," said Humpty Dumpty, stretching out one of his great hands, "*I* can repeat poetry as well as other folk if it comes to that—"

"Oh, it needn't come to that!" Alice hastily said.

—*Lewis Carroll, Through the Looking Glass*

This work addresses research issues in decision theory, medical informatics, and computer-assisted modeling. My research is the only work of which I know to apply graph-grammar derivation systems to decision analysis,[1] and the only work to correlate clinical terminology with prototypical patterns in decision models. However, my research does rest on a body of work in the information sciences. In particular,

[1]Schocken and Jones (1993) have used a graph grammar in a syntax-driven editor for decision trees. However, their grammar imposes only symmetry constraints, which are unnecessary and irrelevant for influence diagram models.

it builds on research done on automated construction of decision models, on medical lexicography, and on graph-grammar derivation of semantic-network models.

2.1 Contribution to Decision Modeling

As currently practiced, decision analysis is extremely time consuming and is not practical for daily clinical decisions. In Figure 2.1, I depict decision analysis as an iterative procedure divided into three main tasks: **modeling, assessing,** and **testing.** The problem that my research addresses can be viewed as the final part of the modeling phase—namely, drawing the structural model. My interest in this particular step stems from the dearth of automated assistance for what can be an arduous task. In the assessing phase, investigators have developed automated assistance for assessing probabilities (Holtzman, 1988; Heckerman, 1991) and for assessing utilities (Farr and Shachter, 1992). Some researchers have used Boolean operators[2] and other prototypical causal interactions to assist the practitioner in assessing conditional probabilities (Rousseau, 1968; Wellman, 1990a; Díez, 1993; Srinivas, 1993; Heckerman, 1993; Pearl, 1988, pages 184–194). Within the testing and evaluation phase, researchers have developed tools to measure the sensitivity of variables (Howard, 1984b) and to compute the value of a given piece of information (Howard, 1984a). Also, several investigators have developed algorithms to perform probability-updating and

[2] Ingo Beinlich. Personal communication.

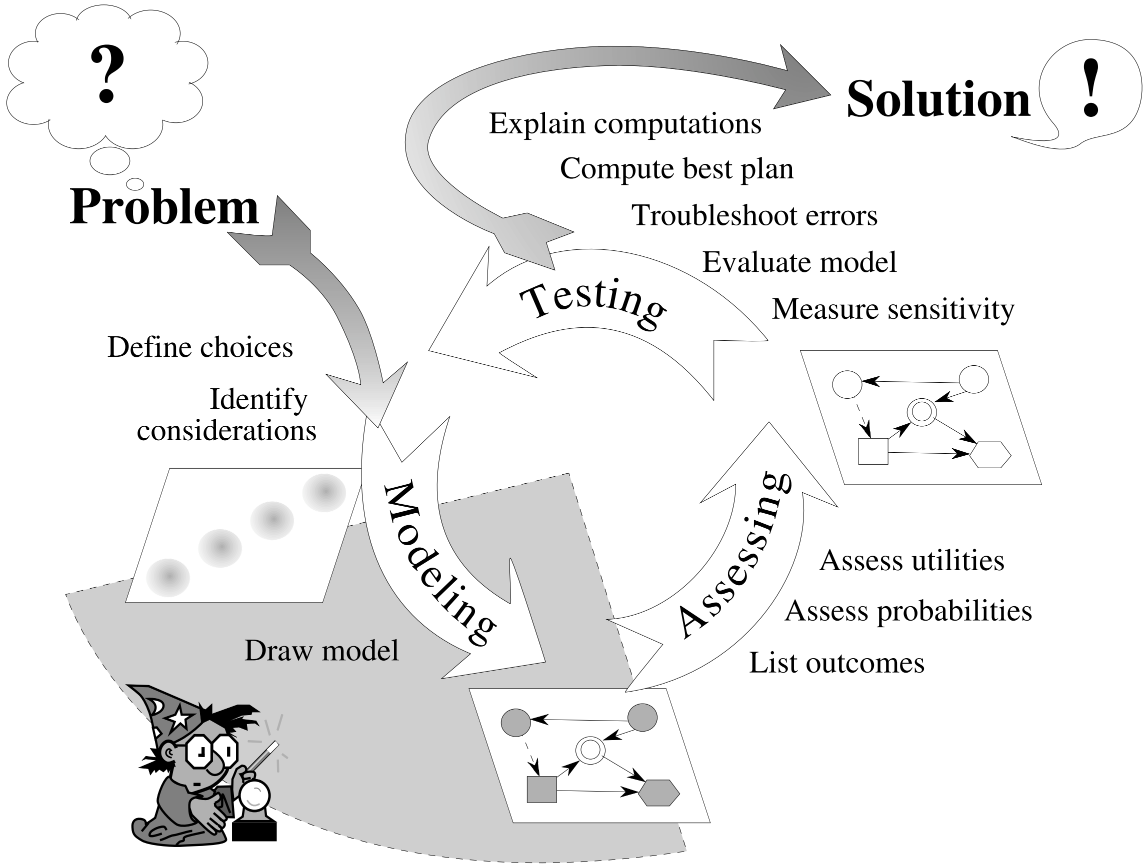

Figure 2.1: The decision-analysis cycle divided into three phases: modeling, assessing, and testing. In the modeling phase, the decision maker must enumerate her choices and the considerations on which those choices depend. In the assessing phase, she assigns numerical probabilities to chance events, and numerical utilities to possible outcomes. In the testing phase, she analyzes the sensitivities of specific variables and compares the model's behavior to her understanding; if she finds significant discrepancies, she may refine, reassess, and retest her model. Gramarye performs the graphical structuring of a model, as indicated by the shaded region in the lower left.

expected-value computations once the diagram has been constructed (Pearl, 1986; Lauritzen and Spiegelhalter, 1988; Chavez and Cooper, 1990). Others researchers have developed programs that generate explanations for the model's behavior and provide advice (Suermondt, 1991; Langlotz et al., 1988). In the modeling phase, however, researchers have not yet found many techniques to help them draw the qualitative structure of a problem, unless that structure already exists in a knowledge base.

The medical decision-making community, to a large extent, has relied on **static models,** where a decision model is defined completely before the model is made available to potential decision makers. Although such an approach may be appropriate for public-health policy making, and for situations that reoccur with little significant variation, it does not meet the needs of the physician and patient whose problem is not exactly the one anticipated. Consequently, many researchers argue for **constructive approaches** (Leong, 1991), where the model is not merely flexible, as in the work of Holtzman (1988), but rather is constructed anew for the individual patient's decision problem. Given the unbounded variety of dilemmas that arise in medicine, and given the inexorable advance of medical technology, such constructive modeling systems will be required before there can be any widespread adoption of normative decision-support tools. Researchers have derived a structure of dependencies directly from data (Herskovits and Cooper, 1990; Geiger, 1992;

Bouckaert, 1992), but this approach does not address the needs of a physician who faces dilemmas that have not been modeled already, dilemmas that do not remind the physician of predefined templates, and dilemmas for which data—including combinations of decision alternatives—are not available.

2.1.1 Construction of Influence Diagrams from Knowledge Bases

Several investigators have found that knowledge bases can imply a directed network of causality that can be interpreted as a graph of probabilistic dependency (Breese, 1992; Goldman and Breese, 1992; Hollenberg, 1984; Horsch and Poole, 1990; Langlotz et al., 1987; Laskey, 1990; Leong, 1992; Provan and Clarke, 1993; Wellman et al., 1992). Some of these researchers have extended the predicate-calculus format of a typical knowledge base to include tables of conditional probabilities (Horsch and Poole, 1990; Goldman and Breese, 1992; Breese, 1992). Laskey (1990) has suggested that probabilistic networks could be constructed from a knowledge base of richly structured and attributed arguments, with prior and conditional probabilities to be added for specific problems and queries. Leong (1992) has devised a semantic-network formalism to include several categorical and uncertain relations, where the uncertain relations include the qualitative probabilistic relations used by Wellman (1990b). Although she makes limited use of vague, categorical indicators that function in place of conditional probabilities and utilities (Leong, 1991), the model's structure must be composed from numerous queries from the user, and, since the actual probabilities and utilities are not stored in the knowledge base, the model must be assessed manually. Langlotz and his colleagues (1987) have used rule-based heuristics to generate possible therapy plans, and simulation models to derive plausible outcomes. In their approach, however, the structure of the decision problem is limited to a single decision—which plan to recommend—and several possible sequelae whose interdependencies are unknown and whose probabilistic distributions cannot be derived from the simulation models. BIFROST (Højsgaard and Thiesson, 1992) employs a simpler semantic classification to group variables into a sequence of causally ordered blocks. However, the necessary grouping in BIFROST must be performed manually by the user.

Wellman's SUDO-PLANNER (1990b) uses a sophisticated abstraction hierarchy, a richly structured knowledge base, and a wealth of planning machinery. A portion of the Wellman's knowledge base is shown in Figure 2.2. The decision models that SUDO-PLANNER constructs are all subgraphs of the exhaustive model shown in Figure 2.3. The structure of SUDO-PLANNER's decision models reflects the structure of the knowledge base; all domain information that SUDO-PLANNER requires to construct specific decision models must be encoded in the knowledge base.

In contrast, Gramarye possesses only a simple term classification and a small

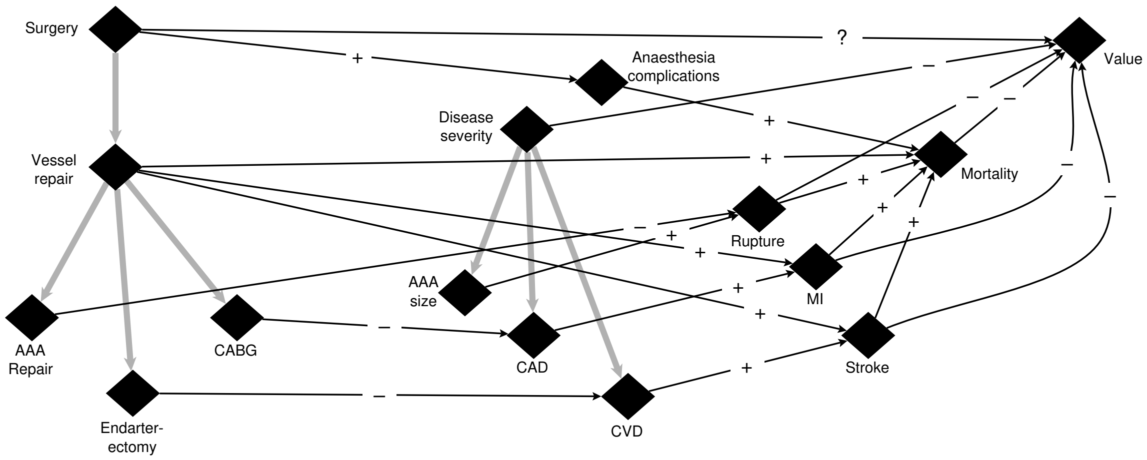

Figure 2.2: A portion of SUDO-PLANNER's knowledge base (Wellman, 1990b, pages 100–102). In this figure, inheritance relations are represented as thickened gray arcs. Domain relations are represented by thin, black arcs, with arc labels indicating whether the concept at the arc's source influences the target concept positively ("+") or negatively ("−"). A question-mark label ("?") indicates that the domain relation is unclear or nonmonotonic. Domain relations in SUDO-PLANNER correspond to qualitative probabilistic arcs in the models that SUDO-PLANNER generates. (AAA = abdominal aortic aneurysm; CABG = coronary-artery bypass graft; CAD = coronary-artery disease; CVD = cerebrovascular disease; MI = myocardial infarction.)

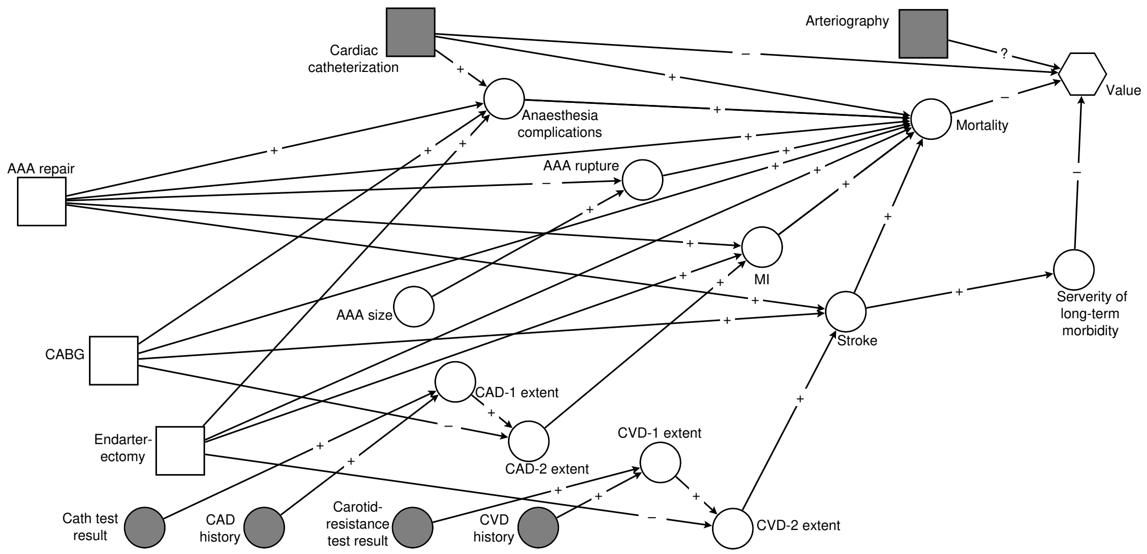

Figure 2.3: The exhaustive qualitative influence diagram produced by SUDO-PLANNER (Wellman, 1990b, page 19). In this figure, arcs represent qualitative probabilistic dependencies. (Details of the qualitative–influence-diagram representation are discussed in Section 3.5.) This model, which encompasses all models generated by SUDO-PLANNER, shows numerous structural similarities to SUDO-PLANNER's knowledge base, a portion of which is shown in Figure 2.2. Nodes that are colored gray represent decisions and probabilistic variables that derive from portions of SUDO-PLANNER's knowledge base that are not shown in Figure 2.2. (AAA = abdominal aortic aneurysm; CABG = coronary-artery bypass graft; Cath = cardiac catheterization; CAD = coronary-artery disease; CVD = cerebrovascular disease; MI = myocardial infarction.)

collection of graphical patterns, and it relies on the user to specify some of the relationships between specific concepts. Also, Gramarye leaves the generated model with no ordering on the decision nodes. I do not doubt that Gramarye's performance would be enhanced by the addition of domain knowledge and sophisticated reasoning. Nonetheless, I have found that Gramarye's syntactic guidance complements the user's specialized understanding of a situation, and that our grammar's contribution to model construction required relatively little knowledge engineering, and is fairly general across medical domains.

Goldman and Charniak (1990) use a system of rules that describes how to transform generic relations into probabilistic arcs, and how to expand the conditional probability matrices at the tail of the newly added arcs. Their system of rules dictates how arcs are to be added, but the conditions for *where* a rule can be applied do not describe classes of local topologies, as graph-grammar productions are able to do.

All these approaches rely on there being a knowledge base that encodes all the specific relationships that appear in the generated influence diagram. This stipulation is difficult to presume in medicine, where the variety of considerations is unbounded. In those specialties within medicine where the variety of considerations is small, it is unclear that constructing the requisite knowledge base should be preferred over constructing probabilistic network models. There is no medical knowledge base of sufficient breadth to generate decision models for arbitrary problems in internal medicine.

2.1.2 Extraction of Influence Diagrams from Larger Models

An approach that is related to constructing decision models from a knowledge base is deriving specific influence diagrams from a larger, more general influence diagram. Researchers (Holtzman, 1988; Jimison, 1990; Bradshaw et al., 1991) have modeled a set of decisions associated with a specific medical subject, such as treatment for infertility, and then have pruned the influence-diagram structure to leave only those considerations that are pertinent to an individual case. This approach requires that there be an exhaustive model that is roughly the graph union of all the decision models for a particular subject. On a grander scale, Heckerman and Horvitz (1990) propose a comprehensive decision model to include all of internal medicine, based on the QMR diagnostic model (Shwe et al., 1991). To model a specific problem, they include any disease with its treatment in a decision model if its posterior probability is greater than some previously defined threshold. If such a comprehensive influence diagram could be built, it would require constant changes as new tests and treatments are introduced into medicine.

2.1.3 Critique of Decision Models

Although the focus of this dissertation is the automated and *de novo* construction of influence diagrams, prior work on critiquing the quality of decision models is closely related: The constructor of a model should avoid those pitfalls identified by critics. In work closely related to mine, Wellman and his colleagues (1989) developed general rules used to critique an existing decision tree. Their program, called BUNYAN, examines a decision tree for signs of common modeling errors (Wellman et al., 1989). One of the rules that BUNYAN uses to critique a model of a medical dilemma is shown in Figure 2.4. The rule derives from the clinical heuristic that tests should not be ordered unless future decisions depend on their results. On the left side of Figure 2.4 is the top-level description that BUNYAN uses for this rule. The pattern language that it uses is defined elsewhere in the program. Moreover, the functions and variables used have suggestive labels and descriptions, but they are defined only in BUNYAN's source code. In contrast, Figure 2.4 also shows the corresponding rule in our graph grammar. I discuss the graph-grammar representation that I use in Section 3.3. However, the basic meaning of a graph-grammar rule is simple. The graph-grammar rule in Figure 2.4(b) has the following interpretation: *If* nodes of the types ⟨present disease⟩, ⟨treatment⟩, and ⟨utility⟩ are present in an evolving model, *then* additional nodes of types ⟨test⟩ and ⟨test result⟩ may be added to the model, along with new

```
(and (nodetype ?decision test-decision)
     (non-degenerate? ?decision)
     (member ?branch (branches ?decision))
     (function-value choice-action ?branch ?action)
     (typep ?action test)
     (not (null? ?action)
     (not (and (member ?x (downstream-nodes
                           (branch-node ?branch)))
               (typep ?x test-result?)
               (function-value
                          test-result-test ?x ?branch)
               (function-value some-branch
                          ?x ?branch1)
               (function-value get-plan
                          ?branch1 ?plan1)
               (member ?branch2 (branches ?x))
               (function-value get-plan
                          ?branch2 ?plan2)
               (not (plan-equal?
                          ?plan1 ?plan2)))))
```

(a)

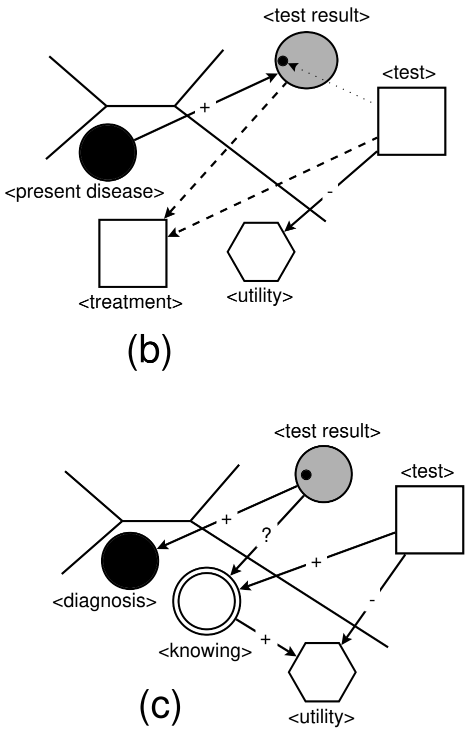

Figure 2.4: Two different representations for clinical maxims. (a) The way that BUNYAN models the clinical rule that no test should be ordered unless a future decision depends on the test results (Wellman et al., 1989). This representation is based on the Lisp programming language, and on numerous functions and variables, which are defined elsewhere in the BUNYAN program. (b) The corresponding rule from my graph grammar. The graph-grammar representation is discussed in Section 3.3. (c) A graph-grammar rule that allows exceptions, where there is an inherent value to knowing the test result. This last rule is not included in the currently used graph grammar.

arcs corresponding to the four shown in Figure 2.4(b). In this graph-grammar representation, nodes with abstract (angle-bracketed) labels, such as ⟨present disease⟩ and ⟨treatment⟩, can be replaced by any of thousands of specific concepts that are listed in the grammar under those respective categories. Thus, the graph-grammar formalism provides a more terse description of structural constraints than can traditional textual programming. Also, the derivation system that I have developed *constructs* models using constraints that subsume those in BUNYAN, thereby eliminating the need for subsequent critiquing.

2.2 Contribution to Medical Lexicography

Existing clinical vocabularies, such as SNOMED International (Rothwell et al., 1993; Côté et al., 1993), group terms into classification hierachies. Other vocacularies, such as CPT (Finkel, 1990), provide a short textual description for individual terms. Neither of these approaches provide terms with the operational semantics that a computer would need in order to perform inferences and queries that are more sophisticated than matching terms. Some researchers have introduced semantic networks that identify relationships that exist among terms (McCray and Hole, 1990; Campbell et al., 1994; Volot et al., 1993; Friedman et al., 1993; Bernauer, 1991; Nolan et al., 1991). However, these semantic networks have provided few operational definitions for the different types of relationships. Consequently, it is still unclear to me how computational inference could be performed over these semantic networks.

My research may also help to clarify medical lexicons by introducing abstractions based on decision-analytic considerations. An abstraction—such as the abstract node label ⟨ablative treatment⟩—is defined by a structural pattern found in decision models. Many of these abstractions are common to clinical parlance, and are found in clinical maxims, but are not listed in medical vocabularies. An abstraction can also be defined by the list of terms that represent decisional considerations. Thus, Gramarye provides definitions that are both **intensional** (based on attributes that are necessary for an example, such as "all states of the U.S.A. that are surrounded by water") and **extensional** (based on the examples to which the term is applicable, such as the set {Hawaii}). The intensional definition can be restricted to properties that can be observed in influence diagrams—namely, where such terms typically fit in a decision model.

2.3 Derivation of Semantic-Network Models

Researchers who work with graph grammars have lamented the lack of "successful 'real' applications" (Göttler, 1990). Other researchers have used the graph-grammar derivation system to build graph-based models other than influence diagrams. Göttler (1992; 1990) has used graph-grammar derivation systems to build computer-aided design systems and graph-editing software. Jones (1990; 1991) has developed a graph-grammar–based derivation system that has generated specialized editors for vehicle routing, production planning, simulation modeling, and decision-tree building. For all these applications, the editors generated are interpretive and graphical, and a set of graph-grammar production rules provides the constraints on what editing operations a user can perform. My research investigates modeling constraints that are strong enough to dictate the entire model construction from a list of elements to be included in the model.

2.4 Pattern-Directed Knowledge Acquisition

Although Gramarye does not help engineers to formulate domain-specific *rules* for a knowledge base, it does help decision makers to organize domain-specific *relations*—here, conditional probabilities—for a decision model. As part of his work on TEIRESIAS, Davis (1984) employed **rule models**—metarules based on the typical format of existing rules in a knowledge base—to critique the *pattern* of premises and actions in new rules that were entered by an expert. **Rule models** comprise four parts: examples, a description, pointers to more general models, and pointers to more specific models. The **examples** are actual rules in the knowledge base that fit the pattern for a particular **rule model.** In a graph-grammar derivation system, these **examples** correspond to the list of concepts that the system can add to a model by using the pattern described by a particular graph-grammar production. The graph-grammar production resembles the **description** of a **rule model,** insofar as both describe the conditions under which the production (or model) is applicable, and the effect of its application. Furthermore, the general classes of symbols recognized by the graph-grammar system are organized into a classification tree, much as **rule models** in TEIRESIAS are organized into a tree according to specificity of a model's pattern. As in TEIRESIAS, Gramarye selects patterns by traversing the classification tree to find the most specific rule that is applicable. Gramarye, however, accomplishes this traversal by ascending from the leaf-node concept to the nearest class that can be added by an applicable graph-grammar production.

There are several differences between the approach that I have taken and that used in TEIRESIAS. In TEIRESIAS, model-based understanding of the knowledge base depends on observed patterns in an existing knowledge base, and these automatically generated **rule models** are "continually revised as a by-product of [interactions] with the expert" (Davis, 1984, pages 187). Gramarye, on the other hand, relies on a static grammar that the designer devises and updates manually. TEIRESIAS infers which patterns are to be regarded as typical by counting their occurrences in an existing knowledge base, whereas Gramarye requires a grammar that specifies typical patterns in terms of a semantic network. As a result of this specification, the patterns that are regarded as acceptable by Gramarye can have theoretical justification and a well-defined interpretation in terms of the underlying semantic network representation. Also, Gramarye does not require an existing knowledge base from which to discern patterns.

Van Heijst and others have used a context-sensitive rewrite grammar to describe several different inferential methods in a system called KEW (van Heijst et al., 1993). In their grammar, nonterminal symbols represent generalized knowledge sources, which are broad categories of problem-solving mechanisms. Terminal symbols represent knowledge-sources (specific problem-solving mechanisms) and metaclasses (broad categories of data that are to be used in the problem-solving task). KEW uses the grammar to direct the incremental construction of both a problem-solving method and a corresponding knowledge base for a given task. Although KEW and Gramarye both use syntax to guide the process of modeling and knowledge acquisition, the two systems address different knowledge-acquisition problems, and they use different types of grammars. The models that KEW produces are abstract frameworks for expert systems; Gramarye produces influence-diagram decision models for specific clinical situations. KEW's grammar is a context-sensitive string grammar, where the terminal symbols represent computational procedures and data structures; Gramarye uses a graph-grammar formalism to build declarative models, with no specification of inferential procedures. The terminal symbols in Gramarye are all random variables, with clearly defined denotational semantics, as discussed in Section 3.5.

2.5 Summary

It is my hope that this investigation of graph-grammar derivation systems will provide new tools for semantic-network modeling, for decision analysis, and for medical lexicography. Graph grammars are a promising notation for describing abstract patterns that are found in graphical knowledge representations, and they might lead to novel insights into how large models and knowledge bases can be acquired, organized, and managed. Although neither the graph-grammar formalism nor the influence-diagram notation are original contributions, the investigation of prototypical patterns in medical decision making, and the derivation of qualitative decision models from such patterns, are two research areas that have hitherto been unexplored, and that promise substantial practical benefits to applied medical informatics.

Chapter 3

Gramarye: A Graph-Grammar Derivation System

> He learnt . . . that there was no law of transmutation . . . , but that every word in both Latin and Greek was to be individually committed to memory at the cost of years of plodding.
>
> —*Thomas Hardy, Jude the Obscure*

A graph grammar consists of a set of production rules that dictates how a graph can be transformed and rewritten. These production rules are quite different from the productions used in rule-based expert systems: Graph-grammar rules can specify a wide range of contexts for which they are applicable, and can describe different graph manipulations for those different contexts. A graph grammar specifies a language over a set of symbols, the members of which are elements of a graph. Graph grammars can provide high-level abstractions that help users to manage complexities. I have found that graph grammars offer an expressive and concise way to represent prototypical forms for modeling dilemmas. Also, graph grammars can provide high-level abstractions that help users to manage complexities.

A graph grammar is merely a specification—a set of bounds on the space of acceptable graphs in a language. Some system must direct the actual graph manipulations and coordinate the sequence in which graph-grammar production rules are applied. Furthermore, the development of the grammar itself is a process of trial and error, and requires a convenient user interface to facilitate testing of how the grammar behaves for various input lists. The derivation system that I have developed accepts a list of node labels and, if possible, produces a graph that includes a node for each node label entered, and that fulfills the constraints specified in the graph grammar.

Derivation of influence diagrams is by no means the only use for graph grammars: They can be used for translation between graphical languages; for graph simplification; or for compilation of graphical input into a linear, textual language. These other applications of graph grammars require strategies for selecting and applying individual productions different from the list-directed graph-building strategy that Gramarye employs.

3.1 Terminology

Throughout this chapter, I shall use the terms **production rule**, **rule**, and **production** interchangeably. I shall distinguish elements of a graph in a production from elements in the emerging influence-diagram model by referring to the former as **vertices** and **edges,** and to the latter as **nodes** and **arcs.** Also, I shall use the term **host graph** to refer to the emerging influence-diagram model.

3.2 Graph-Grammar Formalisms and Uses

Over the past 20 years, a plethora of formalisms has been developed to describe rewriting procedures for graphs. These formalisms include algebraic graph grammars, array grammars, collage grammars, Δ grammars, edge-label–controlled grammars, expression grammars, graphic sequential rewriting systems, hyperedge-replacement grammars, map grammars, neighborhood-controlled embedding grammars, node-label–controlled grammars, picture-layout grammars, plex grammars, precedence graph grammars, relation grammars, shape grammars, and web grammars (Ehrig et al., 1986; Ehrig et al., 1990; Feder, 1971; Pavlidis, 1972). The various formalisms have expressive abilities that range from the simplest node-label–controlled formalism to the most expressive plex-grammar formalism. Graph grammars have been used in domains as widely varying as botanical morphology, developmental biology, pattern recognition, programming-language semantics, database design, visual–programming-language compilers, and kitchen design.

Researchers have adopted various definitions of context-freeness for graph grammars (Feder, 1971; Pavlidis, 1972; Nagl, 1976; Montanari and Rossi, 1986), and several of the formalisms have been shown to be expressible in terms of other formalisms (Bunke, 1982; Montanari and Rossi, 1986; Habel, 1992). Corradini et al. (1990) have shown that operational semantics of any logic program can be duplicated by a context-free hypergraph grammar. Also, Nagl (1976) has described the following classes of graph languages, in descending order of generality: **unrestricted, monotone, context sensitive, context free, normal, regular,** and **regular in normal form.**

The particular formalism that I have used is a modification of Göttler's operational graph grammars (Barthelmann, 1990); it is capable of representing languages in any of Nagl's classes. My reasons for choosing this particular formalism are that its representation of a production as a graph facilitates the direct inspection of productions by the grammar's designer, and that its expressive power is sufficient for adding or deleting one or more nodes, with labeled, directed arcs. My current grammars are monotone—that is, no rule yields a net decrease in the number of host-graph nodes. The languages that they describe, however, are expressible in grammars of lower order.[1] If I were interested in parsing graphs, such restrictions on the grammar Gramarye uses would be more important than they are, given that Gramarye generates graphs. My primary consideration is how easy is it for the designer of the grammar to express patterns that he notices in instances of a particular graphical language. My chosen formalism has been satisfactory for modeling patterns, and the implementation of a derivation system for this formalism has been straightforward.

3.3 Göttler's Operational Graph-Grammar Formalism

Each **production rule** in a grammar describes a legal graph manipulation. In Göttler's formalism, we write these productions as graphs divided into four **regions** (Figure 3.1), which partition the vertices into four sets: those in the left region, V_{L}; those in the right region, V_{R}; those in the **indeterminate** region above, V_{A}; and those in the **determinate** region below, V_{B}. The two sets V_{A} and V_{B} are referred to as the **embedding part,** since, together, they describe the space of possible neighboring nodes for the subgraph manipulation. The four regions of a production may be thought of as a pattern to be deleted (V_{L}), a pattern to be added (V_{R}), a mandatory context for the manipulation (V_{B}), and an optional context for the manipulation

[1] Nagl has shown (Nagl, 1976) that all monotone graph grammars have equivalent context-sensitive graph grammars.

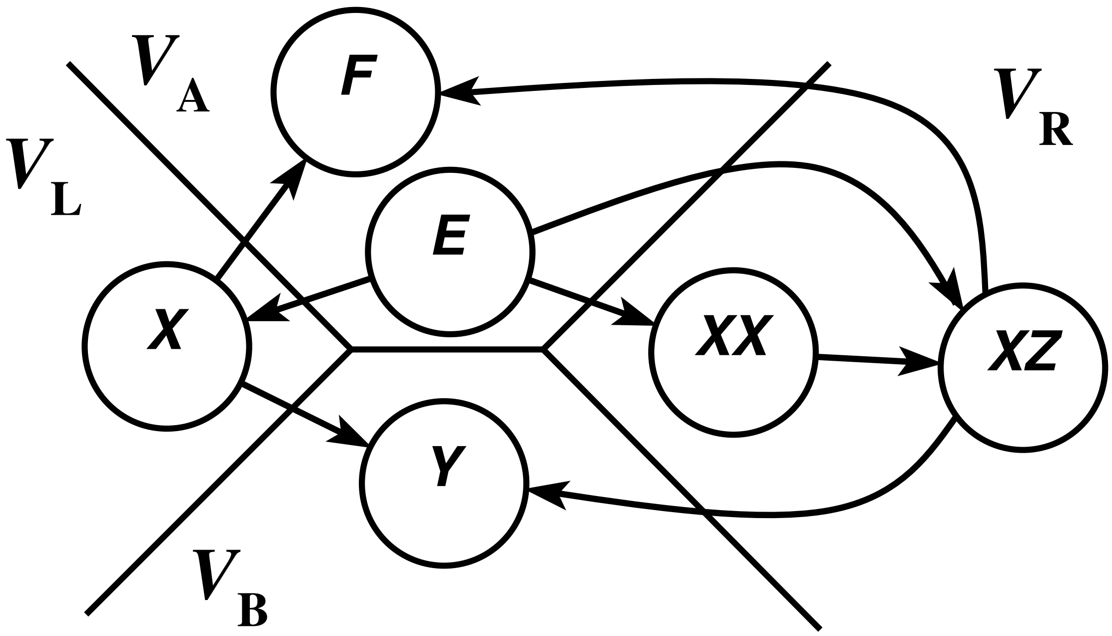

Figure 3.1: Sample graph-grammar production rule. The production denotes that some node of type X in the host graph is to be replaced by new instances of XX and XZ. V_{L}, V_{A}, V_{B}, and V_{R} (left, above, below, and right) are the four regions of a graph-grammar production rule.

(V_{A}). All vertices (V) in the production and the host graph have labels (L_V) and a mapping ($l_V : V \rightarrow L_V$) from vertices to their labels. There is a finite set of edge labels (L_E) for the directed edges ($E \subseteq V \times V \times L_E$) of productions and host graphs. The graph manipulation described by such a production is as follows: Find nodes matching the left region, V_{L}, and replace them with nodes matching the right region, V_{R}. The procedure consists of these four steps:

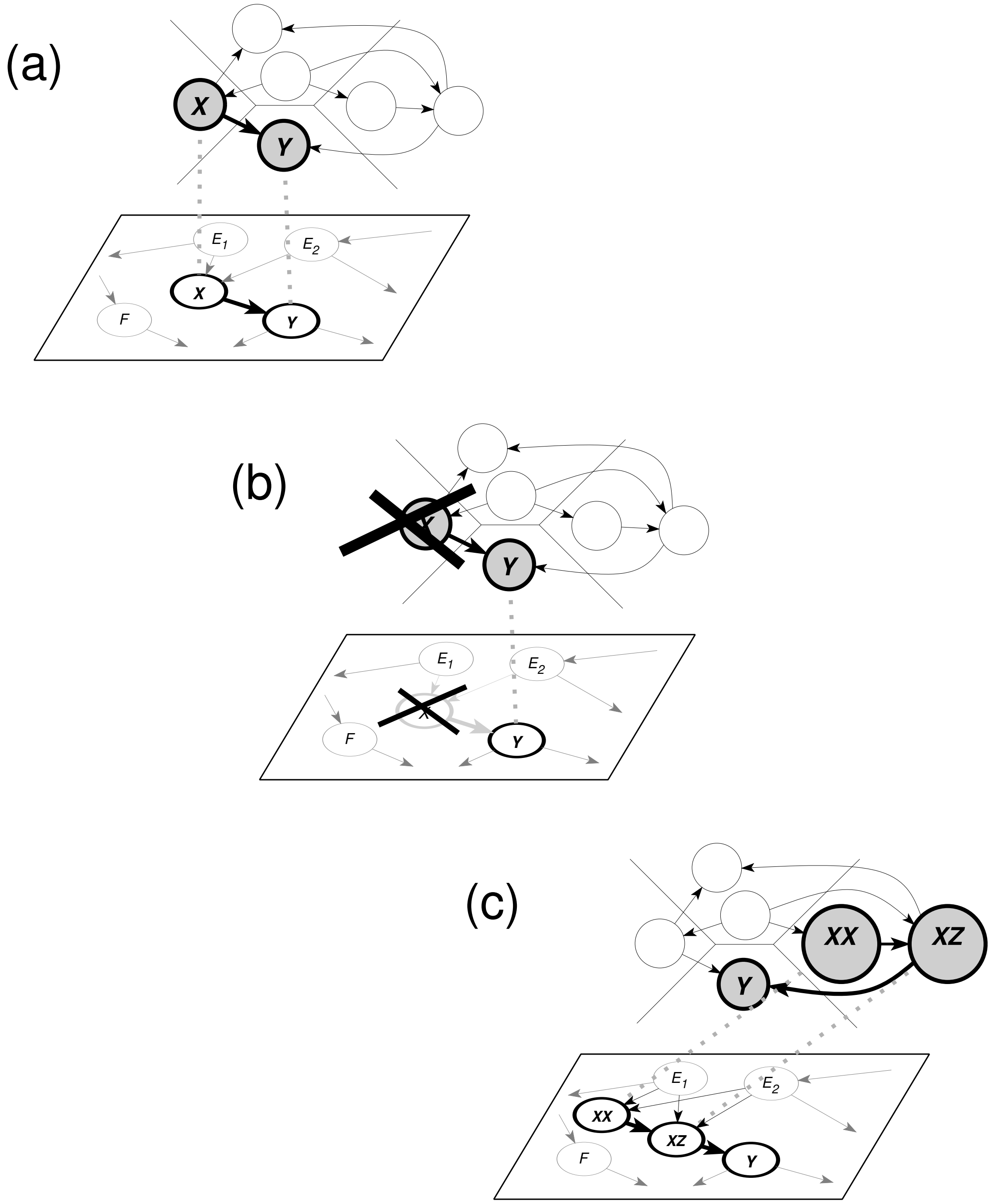

Figure 3.2: Sample application of the graph-grammar rule from Figure 3.1. (a) The first view of the host graph shows two nodes matching X and Y in the production. (b) The node matching X is removed. (c) Additional nodes XX and XZ are added.

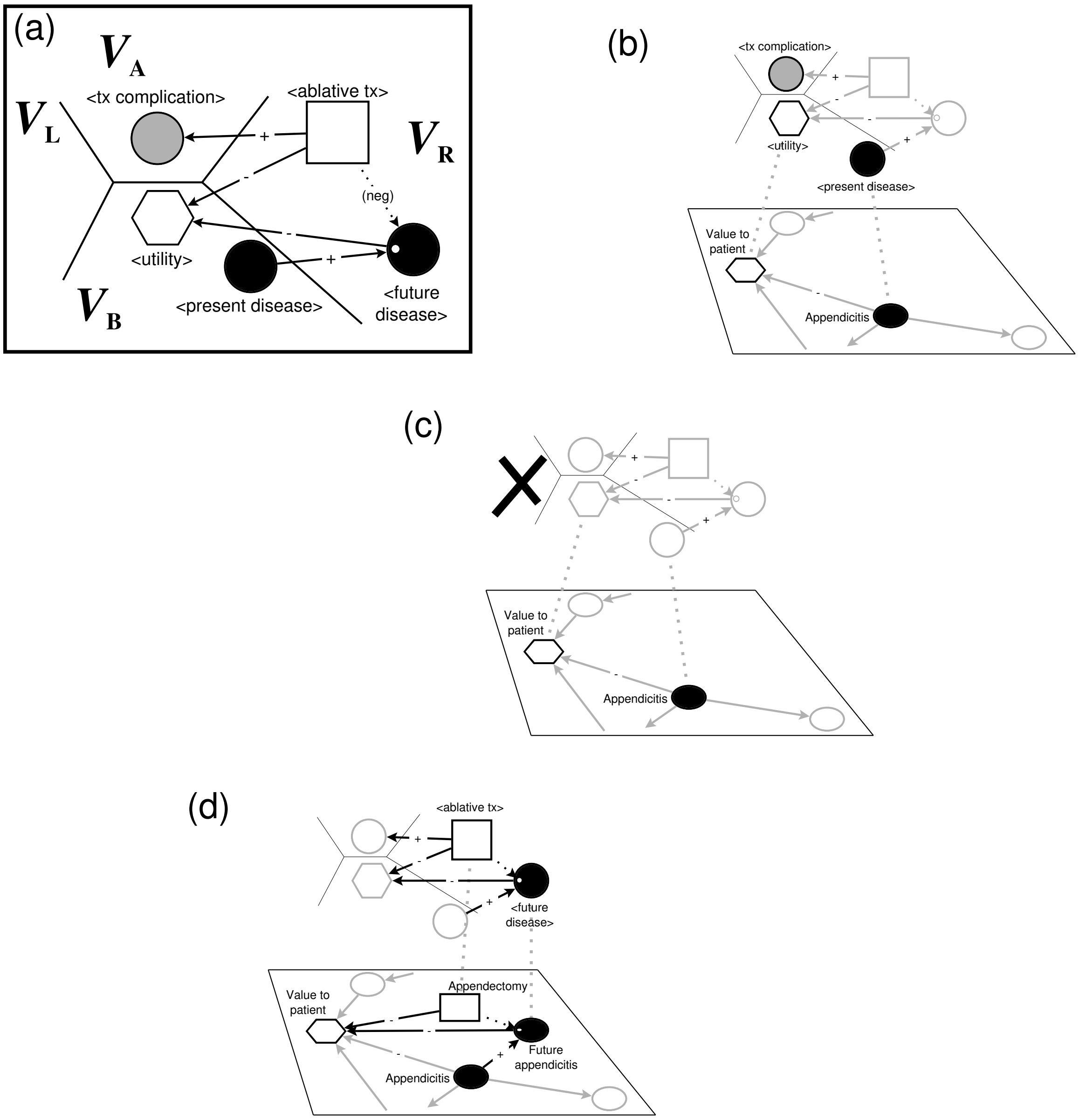

Figure 3.3: Sample application of the graph-grammar production rule. (a) This production rule describes how nodes of the type ⟨ablative tx⟩ can be added to the host graph. V_{L}, V_{A}, V_{B}, and V_{R} (left, above, below, and right) are the four regions of a graph-grammar production rule. (b) The first view of the host graph shows two nodes from the host diagram matching ⟨utility⟩ and ⟨present disease⟩ in the production. (c) If V_{L} contained vertices, a matching set of nodes would be removed. (d) Additional nodes Appendectomy and Future appendicitis are added to the QCID model. (tx = treatment; dotted arcs correspond to contingency arcs.)

1. Find a region of the host graph where the nodes and arcs match the vertices and edges of the determinate (V_{B}) and left (V_{L}) regions of the production.[2] Thus, in Figure 3.2(a), we search the evolving diagram for a node of type X and a node of type Y such that there is an arc from the former to the latter. In Figure 3.3(b), we perform a similar match for medical concepts.

2. Find zero or more host-graph arcs that match edges between the left (V_{L}) and indeterminate (V_{A}) regions of the production.[3] In Figure 3.2(a), we find an arc from E_1 to X and an arc from E_2 to X. In the host graph, the node X has no children of type F.

3. Remove from the host graph those nodes that matched vertices within V_{L}. This step corresponds to the deletion of node X and its incident arcs in Figure 3.2(b).

4. Add to the host graph new nodes and arcs that correspond to the vertices and edges within the right region (V_{R}) of the production, and add to the graph arcs that correspond to the edges connecting the embedding part (from V_{B} and the matched portion of V_{A}) of the production to the right region (V_{R}). In Figure 3.2(c), we add to the host graph nodes of type XX and type XY. Also, we add arcs from XX to XZ, from XZ to Y, from both E_1 and E_2 to XX, and from both E_1 and E_2 to XZ.

[2] If V_{L} and V_{B} match multiple subgraphs, the user must select the appropriate match.

[3] The user of our system must confirm or reject any potential matches to V_{A}. Edges between V_{L} and V_{R} or between V_{A} and V_{B} are not permitted.

Gramarye requires a set of such replacement rules. Gramarye's graph-grammar formalism differs somewhat from that described by Barthelmann. In particular, it requires the user to select the location of a production's application when more than one location is appropriate, and to select which nodes in the host graph should match vertices in V_{A} if there exist such matches. Nodes matched to V_{R} that already appear in the host graph are not duplicated.

Since the variety of node labels that appear in medical decision models is huge, whereas the number of distinct patterns is on the order of dozens, the graph grammar that I use to derive medical decision models employs labels for abstract types of nodes, rather than labels for specific nodes. Throughout this dissertation, I shall refer to the abstract types as **nonterminal symbols,** and to the specific node labels as **terminal symbols.** When a production describes simple node replacement with no constraints on the embedding environment, we can abbreviate that production with a string-grammar production. Since I restrict these string-grammar productions to replace nodes with general labels with nodes with more specific labels, the entire portion of the grammar that I can represent as string-grammar productions is, instead, represented as a single classification hierarchy, with the nonterminal symbol $\langle\mathsf{node}\rangle$ as the universal class; the symbols $\langle\mathsf{chance}\rangle$, $\langle\mathsf{decision}\rangle$, and $\langle\mathsf{utility}\rangle$ representing the first partition of nodes; and successive partitions eventually ending with terminal symbols, which are those specific considerations that a user might enter and that would have meaning as random variables in an actual influence diagram. Therefore, the vertex labels in my productions can match to any descendant terminal symbol in the node-label hierarchy. For example, in Figure 3.15, the terminal symbol Tb is classified under the nonterminal symbol $\langle$malady$\rangle$. Consequently, a production vertex labeled $\langle$malady$\rangle$ can match to a host graph node labeled Tb. Gramarye also requires that the derivation proceeds in stages, where all rules that can be applied and that would add considerations to the host graph are applied as a stage, prior to any search for additional rules that may be applicable. This latter convention allows the grammar to dictate the outcome of any derivation *without any explicit specification of the order in which production rules should be applied.* Section 4.3.1 describes this order independence in depth. Technical details of the graph-grammar formalism that I have implemented can be found in Appendix A.

3.4 Representation of Typical Graph Patterns

In trying to automate the assembly of models from small, reusable pieces of structural information, we are shifting the burden of design from the composer of the individual model to the developer of the graph grammar. The terse clarity of Göttler's operational formalism is particularly useful to the developer, and here is where the advantage of graph grammars over a typical programming language is most noticeable. The graphical representation of each rule gives the developer a declarative picture of the

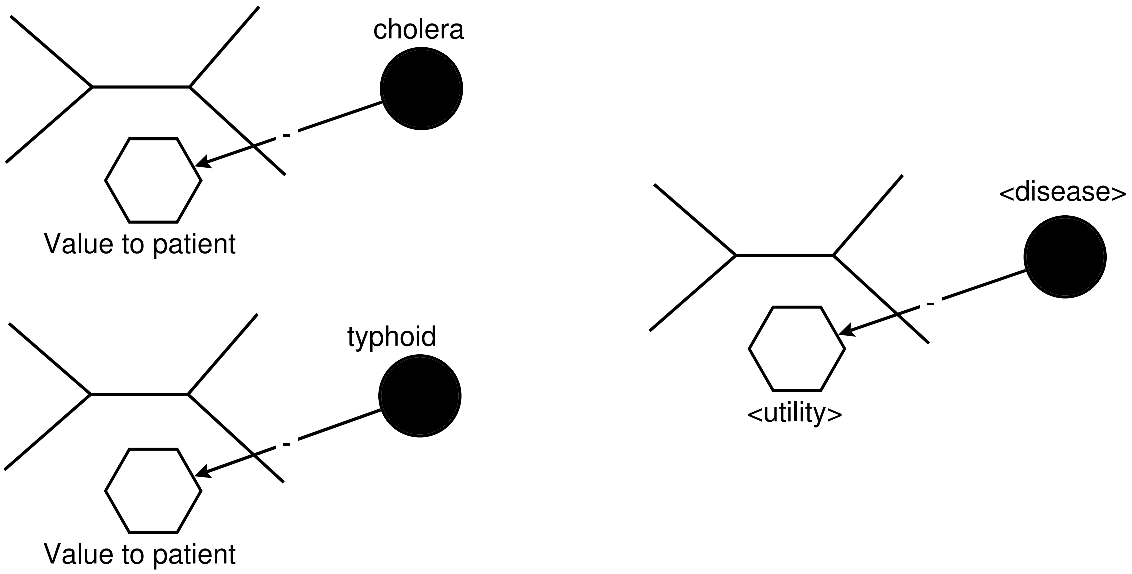

Figure 3.4: Abstraction of graph-grammar patterns. On the left, two rules that are syntactically equivalent add two different diseases. Since many other diseases may be added to a model in the same manner, the graph grammar uses the single rule on the right.

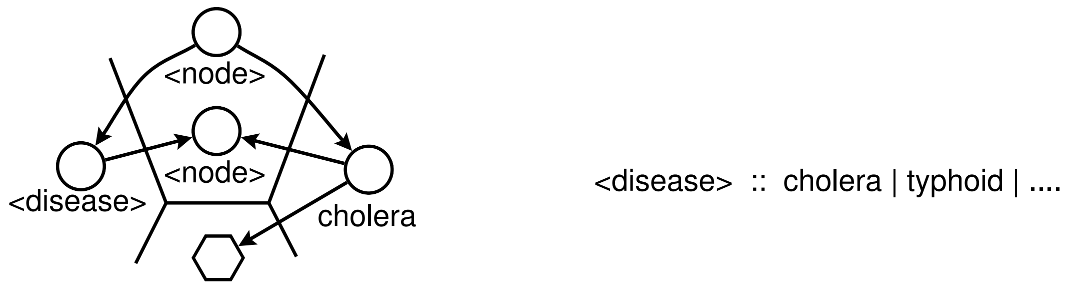

Figure 3.5: Alternative formalisms for expressing simple node-label replacement. On the left is a graph-grammar rule that changes a chance node's labels from ⟨disease⟩ to cholera without altering the rest of the graph. On the right is the equivalent rule in the format—Backus–Naur form—of a production rule in a context-free string grammar. The latter format is used in Gramarye for describing which terminal symbols can be substituted for a given nonterminal symbol.

complex interdependencies of the production rules. Exactly how the computer effects the instructions contained in each rule is irrelevant to the designer. It is important that the denotational semantics of the graphical representation are clear and unambiguous, so that the developer of the grammar can reason about the graph grammar and devise adjustments when the grammar derives models that appear faulty.

To construct medical decision models, Gramarye uses patterns that describe how to introduce certain types of influence-diagram nodes (e.g., nodes that represent the complications of a treatment procedure) into an evolving influence-diagram decision model. These patterns are described in Section 3.6.1. However, Gramarye's derivation system accepts any grammar in the format described in Section 3.3, including grammars outside the domain of medicine, and grammars that describe a space of models written in graphical representation languages other than the influence-diagram representation that I use in this dissertation.

3.5 Qualitative Contingent Influence Diagrams

In the **influence-diagram** notation, there are only four types of nodes: **chance nodes, decision nodes, deterministic nodes,** and the **utility node** (Figure 3.6). **Decision nodes** stand for a set of exclusive alternatives among which the decision maker must choose. A **chance node** represents a random variable that describes an uncertain outcome or an unknown factor. **Deterministic nodes** are a degenerate

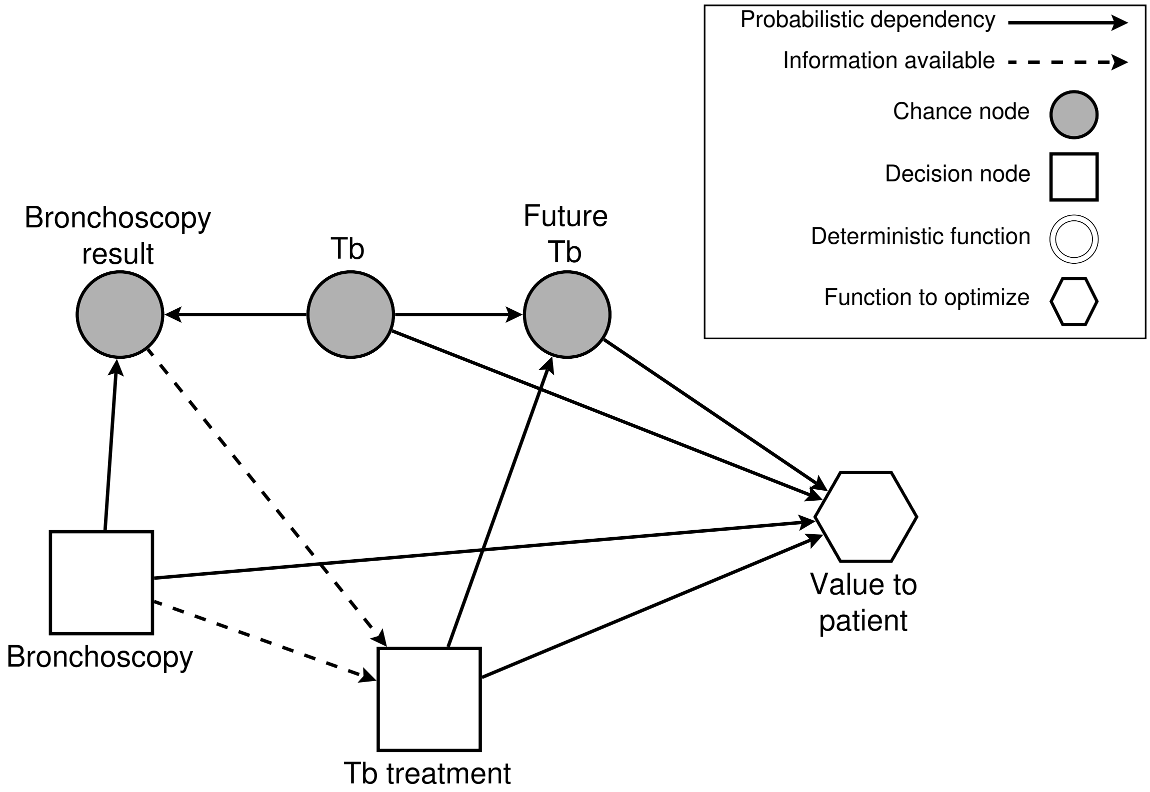

Figure 3.6: An influence-diagram model for the tuberculosis decision. The decision maker must decide whether testing, and whether treating, can be expected to do the patient more good than harm.

version of chance nodes: Deterministic nodes are fully computable as a function of the values of their parent nodes. The **utility node** is a special deterministic node, the value of which the decision maker wishes to maximize.

An **influence diagram** consists of a set of these nodes in a **directed acyclic graph** (**DAG**). The arcs in this DAG are solid when their target is a chance, deterministic, or utility node. Solid arcs denote a probabilistic dependency of one variable (indicated by the target node) on another variable (the source node for the directed arc). When their target is a decision node, the arcs are dashed; they represent the availability of information (about the source node's value) at the time when the decision maker must choose among the alternatives represented by the target node.

Influence diagrams offer an explicit representation of probabilistic independence, denoted by the lack of an arc between two nodes. Unlike decision trees, influence diagrams grow linearly with the number of factors being modeled.

As a simple example, Figure 3.6 shows an influence-diagram model for the decision problem first described in Section 1.1. Recall that our hypothetical physician (1) suspects that her patient might have tuberculosis (Tb), and (2) wishes to avoid any unnecessary testing or treatment. She has noted suspicious historical findings, such as blood-tinged sputum, that are summarized in her prior estimate of the probability that the patient has Tb. She is still fairly uncertain, and considers performing bronchoscopy—which involves inserting a long mechanical instrument down the patient's throat and into his lungs—to obtain a specimen that might establish the Tb diagnosis. The bronchoscopy is invasive, and it might not produce any additional information about the patient's disease. If she and her patient elect not to perform the bronchoscopy, they must still decide whether or not to begin antituberculosis treatment. If she and her patient elect to perform the test, they can use the additional information provided by the test to decide whether to begin therapy. Her goal, and presumably her patient's goal, is to improve the patient's health while causing as little harm, creating as little risk, and incurring as little expense as is possible. The goal will be to maximize the expected outcome utility for the patient, where **outcome utility** is some function of patient preferences based on risk, noxiousness, cost, or whatever the salient factors are for the decision maker.

From the diagram in Figure 3.6, we see that the physician's—or patient's—decision to perform or not to perform the test affects the test results (i.e., they will not be available if she does not perform the test). The decision to perform the test also directly affects the overall value to the patient, since the test is associated with some cost, some risk, and some disutility for the patient. The decision to treat or not to treat affects the probability that the patient will have Tb in the future (i.e., at the end of therapy). The treatment decision also directly affects the overall value to the patient, since treatment too is associated with some cost, risk, and disutility.

Performing the test costs a fixed amount of money, is somewhat noxious to the patient, and carries some risk. Presumably, the only reason to perform the test is that it may decrease the physician's uncertainty about whether the patient has tuberculosis, putting the physician in a better position to choose between treating and not treating her patient. The test result helps the patient, by making the physician less likely to treat nonexistent disease, by making the physician more likely to treat existing disease, or by doing both. This **value of information** can be calculated from an assessed[4] influence-diagram model (Howard and Matheson, 1984). If the utility

[4]Throughout this dissertation, I shall use the term **assessment** to refer to the assignment of

of treating is expected to be higher than that of not treating, regardless of the result of the test, then the test provides no value to the patient. Similarly, if the utility of treating is expected to be lower than that of not treating, regardless of the result of the test, then the test provides no value to the patient. These results, and more specific recommendations for a given situation, follow from the influence-diagram model, the assessed probabilities, and the assessed utility model. The influence diagram does not describe explicitly how each random variable is modeled. The decision maker may want to regard tuberculosis as being a **binary variable**—either present or absent. On the other hand, she may wish to distinguish different classes of severity, either because the different classes have a different effect on the patient's well-being, or because epidemiological data group cases in such a manner. The initial qualitative model does not commit the modeler to any particular level of precision for the variables in the model. However, the initial model does provide a framework for assessment format and subsequent modeling decisions.

Without assessing the model (i.e., assigning specific probability distributions and deterministic functions to each node), we can still provide more information about the physician's dilemma. We know that the patient is adversely affected by the existence and severity of present and future tuberculosis with which he might be afflicted. We know that the patient would prefer to avoid any unnecessary testing or treatment.

specific probability distributions to each chance node, specific alternatives to each decision node, and a specific utility function to the utility node.

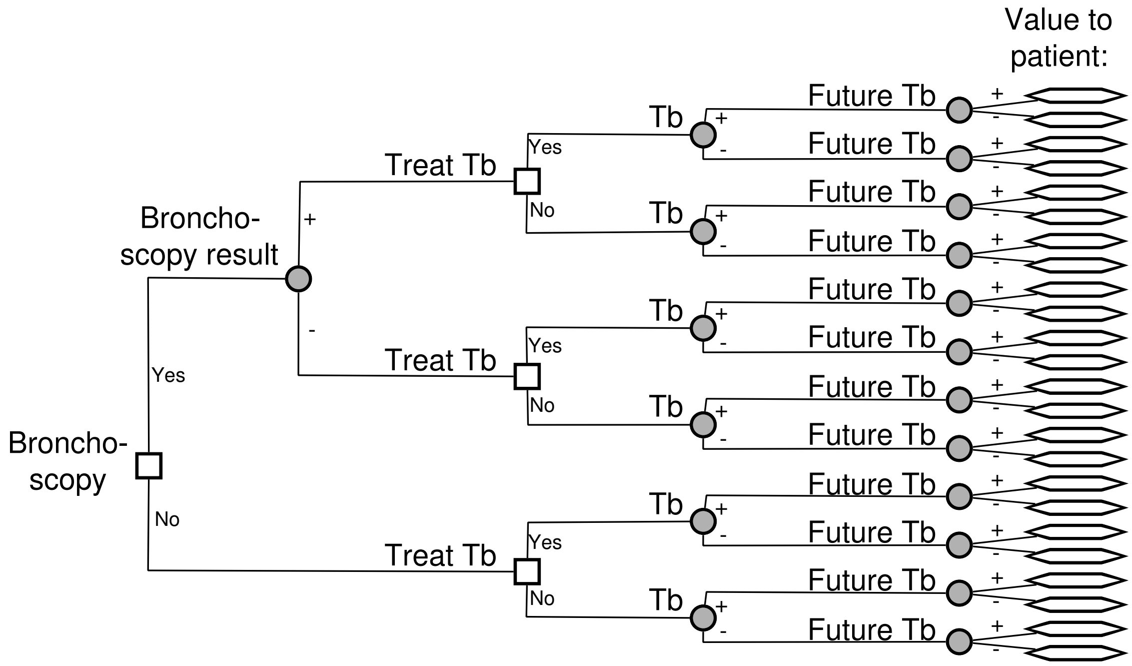

Figure 3.7: A model using the *decision-tree* notation, representing the tuberculosis decision problem. Note that the decision tree is *asymmetric:* The chance node Bronchoscopy result appears only above the first decision, which reflects the fact that Bronchoscopy result is meaningful in only those scenarios where the decision maker has decided to perform the bronchoscopy.

Most relationships between variables representing domain concepts have a typical qualitative value—either proportional or inversely proportional. To describe monotonic constraints on probabilistic and functional dependencies, we follow Wellman's (1990a) use of **qualitative** arc labels: A plus sign, "+," restricts the target node's distribution of values to vary in the same direction as do changes in the source node's distribution of values; a minus sign, "−," restricts the source and target nodes to vary in opposite directions; a question mark, "?," indicates an unclear or nonmonotonic relationship.

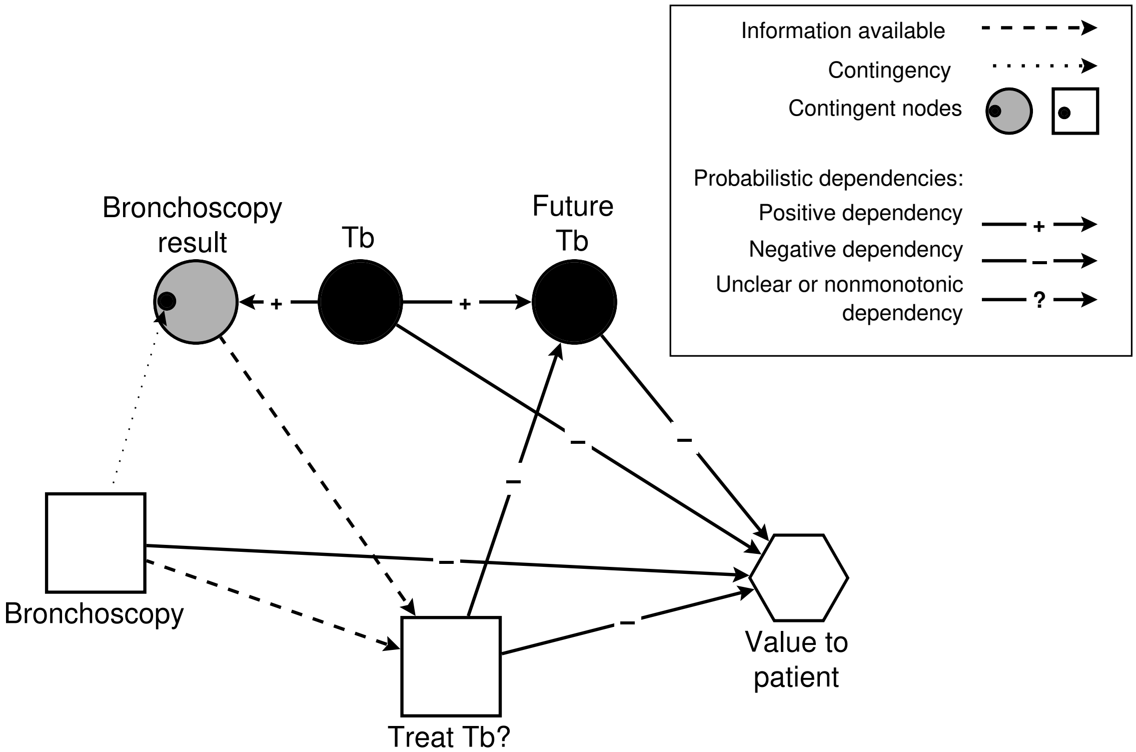

Figure 3.8: A model using the quantitative contingent influence-diagram (QCID) notation, representing the tuberculosis decision problem (see Section 3.5). In the QCID notation, **asymmetries**—such as the pertinence of test results only when the test has been ordered—are represented explicitly with contingent nodes, which have dots on their left sides. This use of contingent nodes delineates absolute dependencies—such as the dependence of Tb test results on the Tb test decision. Also, the solid arcs distinguish probabilistic relations that are proportional from those that are inversely proportional. The former are labeled with a plus sign, +; the latter are labeled with a minus sign, −. For visual clarity, I color disease nodes black and other chance nodes light gray.

Also, we know that there are **asymmetries** buried in the model (Figures 3.7 and 3.8). Here, I mean by **asymmetries** those relations—probabilistic or informational arcs—that are valid for only particular values of certain nodes. For example, the informational arc from test results to the treatment decision makes sense only if the physician decides to perform the test. Because these asymmetries are common, I use **contingency nodes,** a notation described by Fung and Shachter (1990). In this notation, a node is divided into several contingent nodes, each with exclusive conditions. Each contingent node is considered relevant to the rest of the diagram for only those scenarios in which its conditions are met. If a contingent node's conditions are not met, then the node and its incident arcs can be ignored. By introducing contingent nodes to the influence-diagram representation, I gain the ability to express common-sense principles that may be obvious to people modeling decision problems, but that would not otherwise be apparent in the influence diagram. In my grammar, I delineate these conditions with dotted contingency arcs. Computationally, I represent QCIDs as graphs, and contingencies are represented by dotted arcs, labeled with either a C or a CN.[5] The C label connotes that the target node is contingent on the source being affirmed or having its highest value according to some predefined ordered scale, whereas the CN connotes that the target node is contingent on the source being denied or having its lowest value. I chose this representation of asymmetry because of the

[5]Throughout this dissertation, I have omitted these C and CN labels on contingency edges. In several figures, I have omitted contingency arcs for the sake of clarity.

simplicity and linear growth of a single graph of dependency arcs. Also, Gramarye is designed for the generation of a single graph, rather than of a graph that is decorated by numerous decision-tree structures, as in the formalisms described by Smith and colleagues (1993) and by Call and Miller (1990).

In Figure 3.8, I show, for the physician's dilemma, a model that uses a **qualitative contingent influence diagram** (**QCID**). With QCIDs, we label each probabilistic—also called **conditioning**—arc with the sign of the qualitative dependence indicated by the arc. We also label each contingent node with one or more dots on its left side, each dot representing a constraint on some other variable (or set of variables) that must be satisfied for that node to be relevant to any of its successors. In the QCID models that Gramarye develops, each contingent node is associated with explicit conditions on other nodes through special contingency arcs. These attributes could be represented easily in the diagram as a third type of arc, but such a notation would obscure the visual simplicity of the diagrams. The qualitative arcs in Figure 3.8 express constraints on each outcome's probability distribution.

By adding qualitative labels to the arcs and contingencies to the nodes, we obtain a model that is closer to being an assessed influence diagram, yet the model is still general enough that we do not need to redesign it for each test, for each treatment, or for each patient. On the basis of our qualitative model alone, the physician and her patient cannot decide whether to begin treatment, or whether they should be willing to arrange bronchoscopy: If they are to arrive at such normative decisions, the patient must quantify his utility as a function of noxiousness, expense, risks, and the like, and the physician must assess the precise probability distributions for each outcome. Still, the construction of the QCID model is a major component of the overall modeling task.

3.6 Gramarye's Architecture

In Figure 3.9, the top-level input and output of the Gramarye system is described. A user enters a list of relevant terms, and Gramarye constructs a model with nodes that correspond to those terms. Technically, Gramarye is a derivation system that has no inherent ability to accept medical terms or to generate QCIDs. As mentioned in Section 1.5, Gramarye must be given four pieces of static information before it generates any models:

1. A collection of graph-grammar productions

2. An initial graph

3. A classification of node labels according to abstract symbols used in the productions

4. A visual notation for each subclass of node

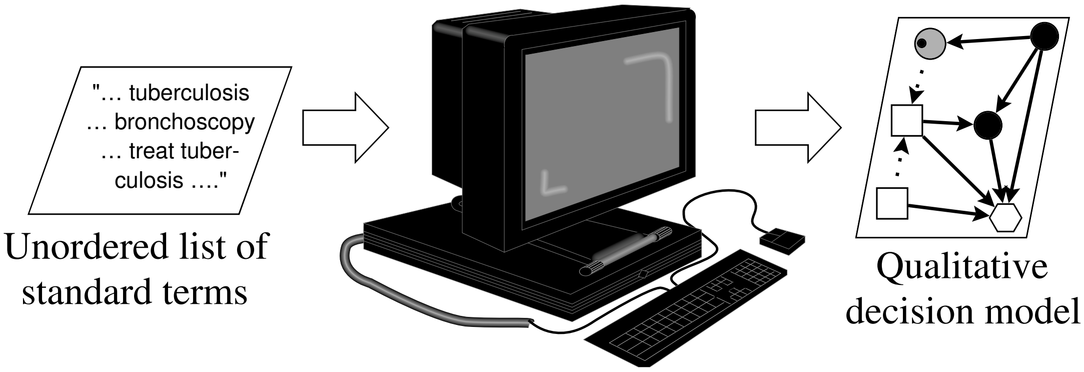

Figure 3.9: The basic function of Gramarye. A decision maker provides Gramarye with a list of decisions and considerations, such as tuberculosis, bronchoscopy, and tuberculosis treatment. Provided that the vocabulary and graph grammar installed in Gramarye are sufficient, a qualitative decision model is returned to the decision maker, who then can assess probabilities and utilities, and thereby arrive at a decision.

The particular application with which I am concerned in this work is that of generating medical decision models. Consequently, to construct influence diagrams from medical terms, I use the following static input:

1. A graph grammar for medical influence diagrams
2. The utility node, Value to patient
3. A classification tree for a medical lexicon
4. The shapes rectangle, circle, and hexagon for decision, chance, and utility nodes, respectively[6]

[6]I have treated deterministic nodes as degenerate chance nodes, where the probability distribution is peaked at a single value.

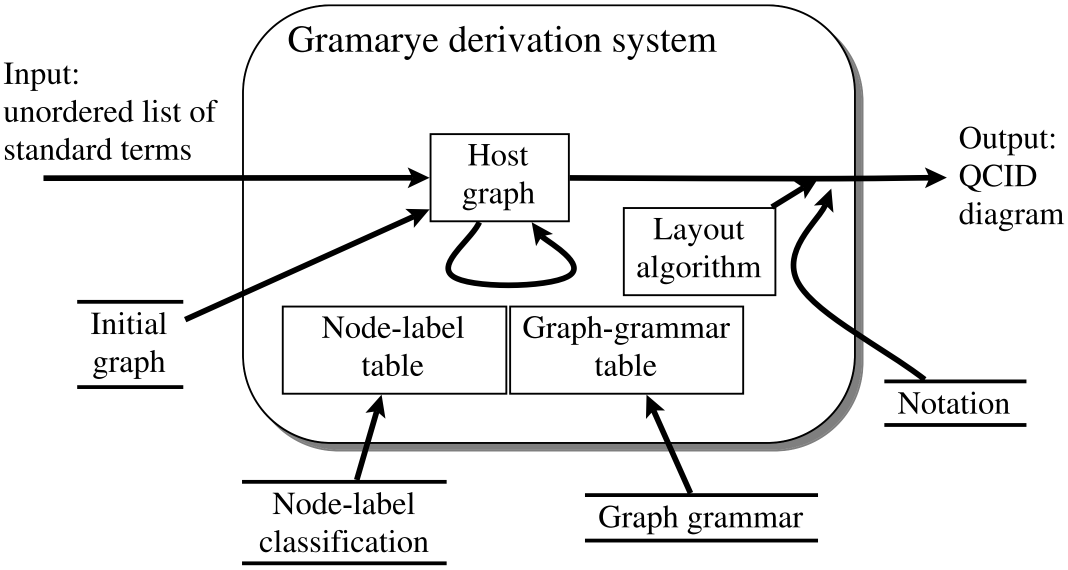

Figure 3.10: The basic structure of Gramarye. The four static inputs—the initial graph, the node-label classification, the graph grammar and the notation—are shown outside of the Gramarye derivation system. All information that is particular to medicine and to influence-diagram modeling is contained in these four static data structures.

Throughout this dissertation, I shall often ignore the distinction between Gramarye and its static input, and shall assume that these static components are part of Gramarye. In this section, however, I shall examine the details of Gramarye's architecture, and the process of constructing the graph grammar and the vocabulary of terms.

Figure 3.10 shows the static input and general structure of Gramarye. Currently, Gramarye consists of a user interface for loading the static input, a grammar checker for evaluating the coherence of the static input, a graphical browser for choosing classified concepts and for entering new concepts, a command-line interface for interacting with the user during the derivation process, the derivation system, a graph-layout algorithm designed for QCIDs, and a diagram generator. The medical lexicon—portions of which I have extracted from CPT (Finkel, 1990), SNOMED-III (Rothwell and Côté, 1990), and the QMR diagnostic system (Miller et al., 1986)—allows Gramarye to recognize specific entered terms, and to classify them according to the classes used in the graph-grammar productions.

3.6.1 A Graph Grammar for Prototypical Patterns

The current graph grammar appears in Figures 3.11 through 3.14. The graphical representation helps the developer of the grammar to identify structural motifs. Although neither Gramarye nor the QCID notation assigns any significance to node coloration, I have colored diagnostic nodes black and other chance nodes light gray to make the grammar's structural patterns more apparent. All the information in Gramarye regarding prototypical patterns in medical decision models can be viewed as 15 rules, each with no more than six nonterminal symbols. This fact attests to the terseness of the graph-grammar formalism.

The clarity of Göttler's graph-grammar formalism makes adjustments to a particular grammar easy for the grammmar's designer. One possible change would be to expand the grammar and the vocabulary to make strictly semantic distinctions. For example, the current grammar does not distinguish between kinds of tests and

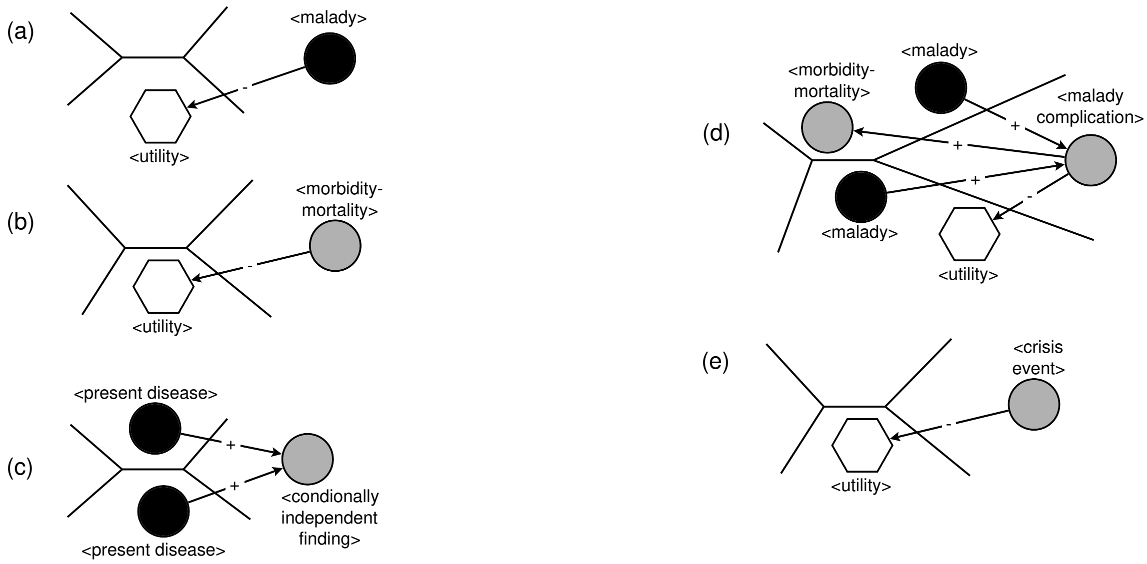

Figure 3.11: The current graph grammar used in Gramarye to derive medical QCID decision models: Part I. (a) The first rule permits any malady being considered by the decision maker to be added to the model with a negative arc to the utility node. (b) This rule allows the morbidities and mortalities of any adverse reaction to be included in a model. (c) This rule adds conditionally independent findings for any given disease. Note that the generality of this rule requires that the user specify which disease is correlated with which finding. (d) This rule allows the system to add a complication of one or more maladies. (e) The system follows the last rule when adding crisis events to a model. Although neither Gramarye nor the QCID notation assign any significance to node coloration, I have colored diagnostic nodes black and other chance nodes light gray to make the grammar's structural patterns more apparent. (The graph grammar is continued in Figures 3.12, 3.13, and 3.14.)

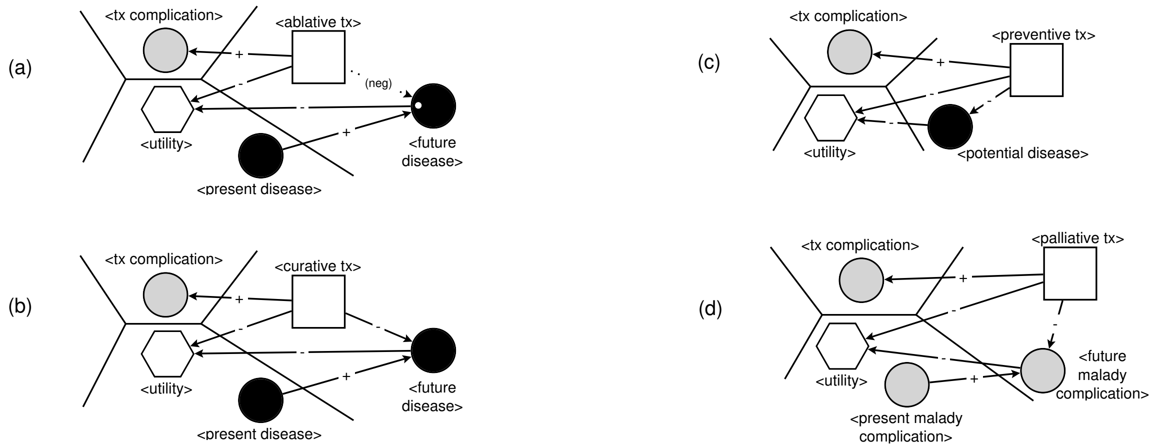

Figure 3.12: The current graph grammar used in Gramarye to derive medical QCID decision models: Part II. Here we see four types of treatment decisions, each distinguished by the structural connections surrounding the decision node. (a) Ablative treatments obviate the relevance of future diseases on the decision maker's utility. (b) Curative treatments reduce the probability of future disease states. (c) Preventive treatments reduce the chance of a potential disease, which the patient may have in the future but does not have currently. (d) Palliative treatments do not affect disease probabilities; they reduce the chance of future complications.

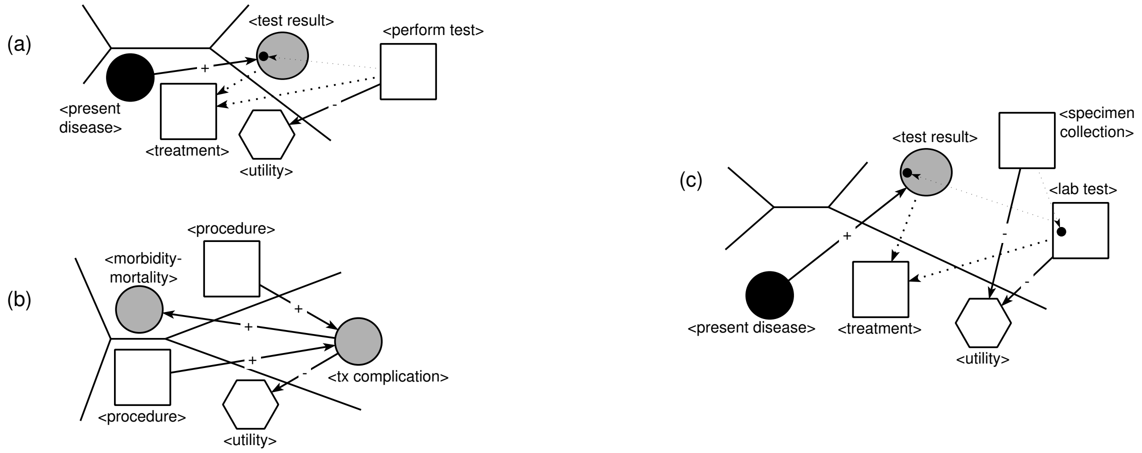

Figure 3.13: The current graph grammar used in Gramarye to derive medical QCID decision models: Part III. (a) The first rule corresponds to the clinical maxim that tests should not be ordered unless the test result will have a bearing on future treatment decisions. (b) A treatment complication is a particular type of adverse reaction: The same treatment complication may result from several different treatments.(c) Since laboratory tests generally require the collection of some specimen—such as amniotic fluid, in the case of amniocentesis—the addition of laboratory tests entails the presence of ⟨specimen collection⟩ and ⟨test result⟩ nodes as new or existing nodes in the decision model.

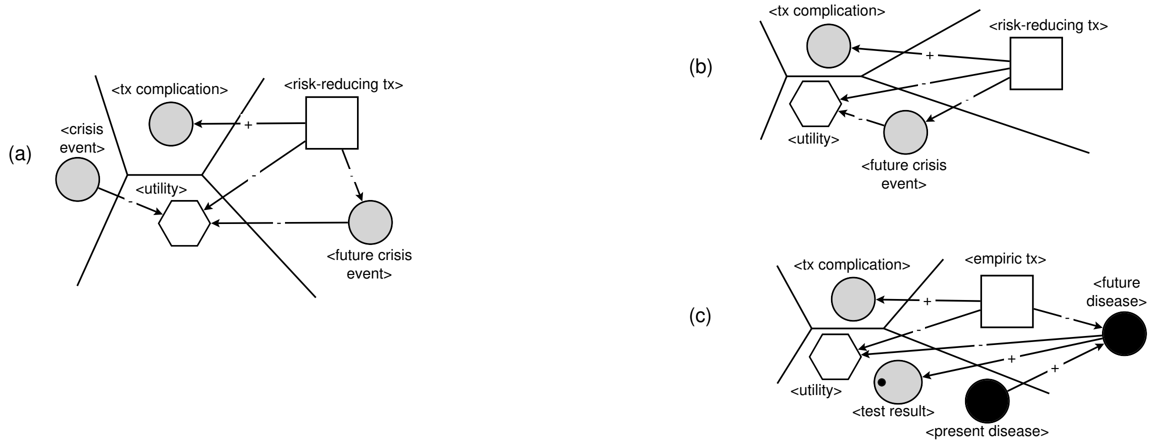

Figure 3.14: The current graph grammar used in Gramarye to derive medical QCID decision models: Part IV. (a) The decision on whether to perform risk-reducing treatment (such as endarterectomy) negates consideration of the present risk of a crisis (such as a stroke) in consideration of the effect of the treatment on reducing the likelihood of that crisis.(b) The second production rule adds a risk-reducing treatment when the appropriate ⟨future crisis event⟩ is already present in the model. (c) Empiric treatment is characterized by a subsequent test result that provides information on how effective the treatment is. Presumably, the test result is available to the decision maker *after* she decides for or against empiric treatment, but this constraint is not enforced by the current grammar.

diseases. A more elaborate classification hierarchy might group EEG and EKG tests in separate nonterminal categories, and a more elaborate graph grammar might then restrict EEG tests to CNS diseases, and EKG tests to cardiac diseases.

The current grammar concentrates on different structural patterns and does not make such purely semantic distinctions. However, the separation between syntax and semantics is not obvious. Typically, grammars define the **syntax** of a language (e.g., the linear sequence of parentheses, tokens, and semicolons in a PASCAL statement), but say nothing about the **semantics** (i.e., how tokens or groups of tokens affect the run-time behavior of a program). The basic syntax and semantics of the QCID semantic-network language are simple and well defined (see Appendix B). A graph grammar *over* the QCID language defines additional restrictions, and these restrictions may appear syntactic (e.g., decision nodes must have a negative arc to the utility node) or semantic (e.g., for a treatment to be an ⟨ablative treatment⟩, it must have a certain declarative relationships with ⟨utility⟩, ⟨disease⟩, and ⟨future disease⟩ nodes). We use **denotational semantics** to refer to how the symbols and expressions in a programming language dictate a final state, given the initial state and the input. Denotational semantics are distinct from **reference semantics,** which describe the relation among symbols in a program and those symbols' referred objects in the user's environment. The graph grammar maps clinical concepts—such as "palliative treatment"—to *syntactic* rules about how nodes are to be included in a decision model. However, it is incorrect to label the graph grammar as purely syntactic; the denotational semantics of the resulting language of graphs is designed to mimic the referential semantics of clinical parlance. For example, the clinical distinctions among palliative, preventive, curative, empiric, and ablative treatments arise from ordinary clinical decision making. It is not a coincidence that the same distinctions correspond to separate structural patterns in decision models. Consequently, the grammar maps clinical abstractions into syntactic rules. Perhaps if the target graphical knowledge representation were more expressive than the QCID language, then organ-system abstractions (such as CNS versus cardiac diseases) would correspond to visibly different patterns, and the grammar's distinction for these two sets of diseases would amount to more than a semantic, type-checking restriction. However, I have not yet applied Gramarye to representations that are more expressive than the QCID language.

3.6.2 A Domain Vocabulary

All node labels in the grammar are abstract classes for standard medical terms. I have grouped the medical terms that my grammar recognizes into a **node-label classification**—a classification tree with the capacity for generating simple syntactic variants (Figure 3.15). The terminal leaves of this classification hierarchy are those terminal symbols that the derivation system recognizes and assigns to nodes as labels. The classification hierarchy of medical terminology gives Gramarye the ability to assign each entered term to one or more structural patterns. If the user enters a term that is not already classified according to the nonterminals of the graph grammar, then she must add the term to the classification tree.

To facilitate communication between the user and the system and communication among different users, I prefer to use terminal symbols that are well defined and are commonly used in medicine. The **Systematized Nomenclature of Medicine, SNOMED III** (Rothwell and Côté, 1990), is a standardized coding system for medical findings and diagnoses. SNOMED III uses many of the same semantic types found in our grammar to group over 200,000 standardized terms. **UMLS,** the Unified Medical Language System (Tuttle et al., 1992), contains a compilation of 17 different standardized medical coding systems. The UMLS vocabulary contains over 240,000 terms; where synonyms exist, a single preferred term is associated with each synonym. The terms in UMLS are already grouped according to a detailed classification tree, and according to a network of semantic relations. Many of the abstractions in our node-label tree also appear in the UMLS classification tree. Although I haved considered incorporating both UMLS and SNOMED vocabularies into Gramarye, I am currently using a much smaller lexicon, which contains 5400 findings, 855 diseases, and 1200 procedures.

Clearly, as it now stands, Gramarye's vocabulary is not sufficiently comprehensive for general medical use. I see two possible—and by no means exclusive—solutions to

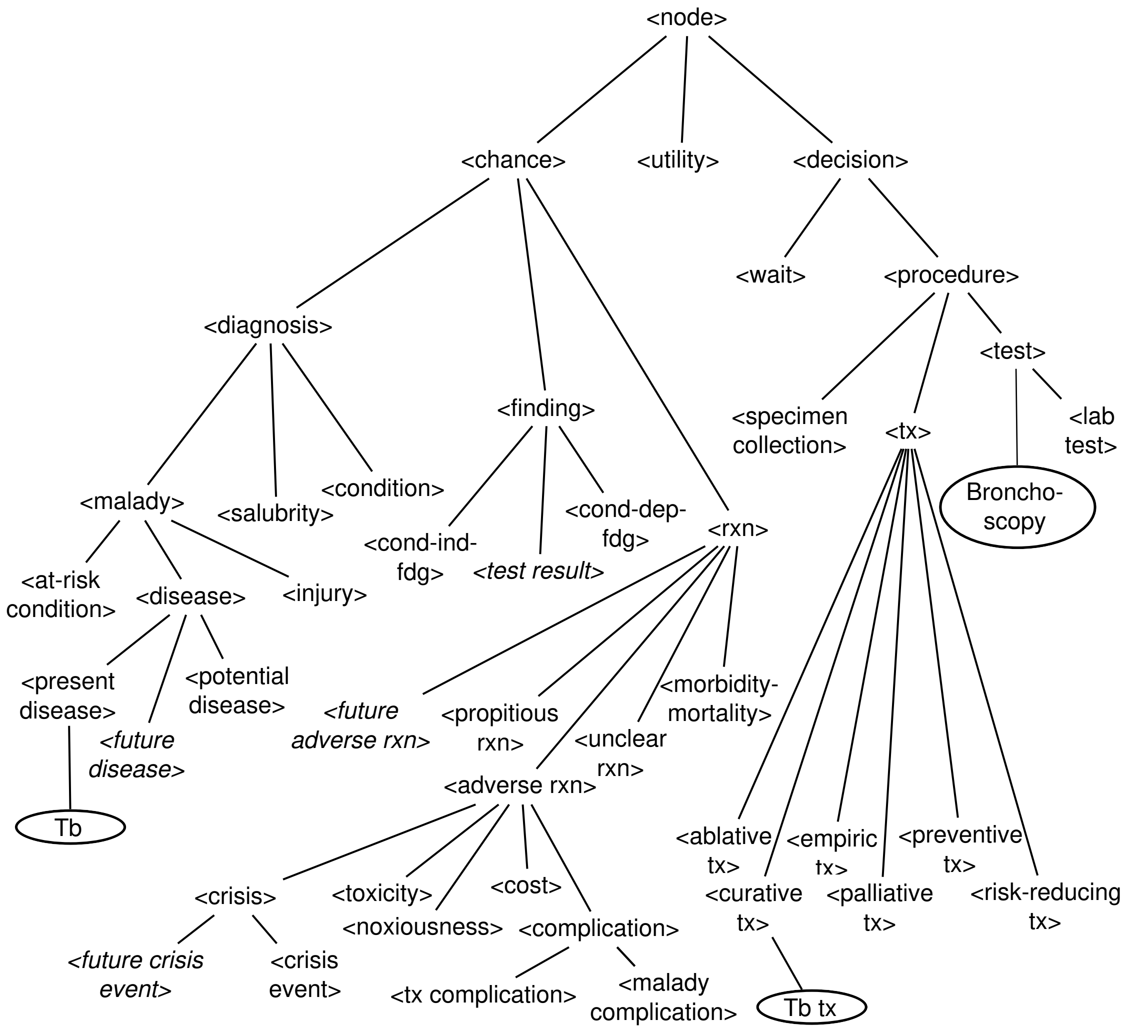

Figure 3.15: Abstract classes for terms in the node-label classification tree. These nonterminal symbols appear in the graph-grammar rules from Figures 3.11 through 3.14. Symbols in italics represent classes for lexical variants that are generated by Gramarye during the derivation process. Tb, Bronchoscopy, and Tb tx are the terminal concepts—rather than abstract classes—that were used as input for the derivation in Figures 3.17 through 3.19. (tx = treatment; rxn = reaction; Tb = tuberculosis.)

this problem: (1) I can increase the size of the vocabulary to cover most foreseeable entries, and (2) I can limit the user's entries to codes provided by the computer.

The first solution is feasible with only large, standardized, structured clinical lexicons, such as SNOMED III, or UMLS. Such a standard vocabulary provides the additional benefit of a well-defined referential semantics: Each term can be associated with a precise English explanation, so there is little confusion among users regarding what the term means in the real world. Such semantics might enhance the shareability of assessed probabilities and default utilities. However, lexical variants of words within an entered phrase may appear in dozens of SNOMED III terms, and the complete phrase may not match any standard term precisely. Also, many decision problems faced by physicians and patients include nonmedical considerations that cannot be pre-enumerated with present technology, so a predefined vocabulary almost certainly will not be sufficient for all clinical decision problems. Although several researchers are working to expand the scope of SNOMED III and to eliminate vagueness and redundancy in that vocabulary (Campbell et al., 1994), I have relied heavily on manual classification of many of the terms in the user's initial list of considerations for a specific decision problem.

The second solution to the restrictions of a single finite vocabulary—limiting entries to those from a computer—could use an electronic medical record that produces a coded representation from a friendly user interface. Campbell and Musen (1994) have developed one possible user interface, and a plan for a formal representation of the medical record. In the system that they envision, a restricted language based on SNOMED III and conceptual graphs (Sowa, 1984) encodes all progress-note information. This type of graphical representation might be used by a system such as Gramarye, not only for the unordered list of standardized terms, but also for additional context that might disambiguate how a decision model should be constructed. The resulting scenario would allow the user of an electronic medical record to begin decision analysis with an automatically generated influence-diagram structure. Subsequent assessment of probabilities and utilities might take advantage of stored clinical data and bibliographic retrieval.

Although most terms that I have considered fall under a single classification in my node-label tree, many terms have more than one role, depending on the context of the decision problem. For example, an excisional biopsy may be considered both as a ⟨test⟩ and as a potential ⟨treatment⟩ for a small skin lesion. Currently, when an entered term appears in multiple places in the classification tree, the derivation system requires the user to choose among the possible roles for that term. Concepts—such as excisional biopsy—that present features of more than one abstraction in a single decision problem may require new abstractions and new productions in my grammar. Where concepts appear to play different roles in different contexts, an enhanced derivation system might be able to use the other entered terms and the

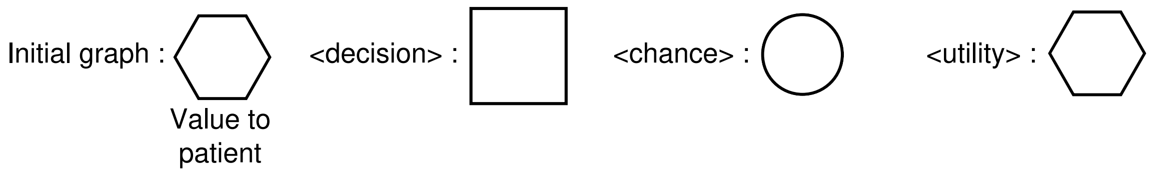

Figure 3.16: The initial graph and the visual notational conventions for the QCID diagramming language.

possible derivations to select the appropriate role.

3.6.3 A Graphical Knowledge Representation

The notational conventions of QCIDs are encoded in a single data structure that matches the immediate subtypes of the ⟨node⟩ class—the root class of the node-label classification hierarchy—to visual attributes available to the graphing module. This mapping instructs the graphing module to use the appropriate shapes, colors, and icons for the various types of node in the graphical knowledge-representation language. In the case of QCIDS, nodes of the ⟨decision⟩ class are assigned the square shape, nodes of the ⟨chance⟩ class are assigned the circular shape, and nodes of the ⟨utility⟩ class are assigned the hexagonal shape (see Figure 3.16).

Whether or not a node is contingent has no effect on whether vertices in a graph-grammar production can match to it; only an explicit occurrence of a corresponding contingency edge in the production has any bearing on whether the node is contingent on another. The absence of any edge in V_{BL} of a production rule does not prevent the application of that production to a subgraph where a corresponding arc does exist. Barthelmann has proposed an extension to Göttler's operational formalism that permits the specification of arcs that must be absent for that production to be applicable. However, I have not yet found this additional expressiveness to be particularly useful for generating medical QCIDs. Also, although Göttler's formalism is nodecentric and does not allow direct manipulation of edges without manipulating the source or target node, this limitation has not yet been a significant impediment in the modeling that I have done. For example, the production in Figure 3.14(a) has the effect of removing an arc between ⟨crisis event⟩ and ⟨at risk condition⟩, and both source and target nodes could remain unchanged by the production if we replaced the ⟨future crisis event⟩ vertex in V_{R} with a ⟨crisis event⟩ vertex.

3.7 Derivation of Graphs from Terms

To show how Gramarye derives a QCID model from a list of terms, I shall step through the derivation of a simple model for our hypothetical physician and her possibly tuberculous patient (see Sections 1.1 and 3.5). Her considerations are three:

1. How likely is it that the patient has Tb?
2. Is it worthwhile administering treatment?
3. Would further testing—in this case, bronchoscopy—be advised?

Consequently, she enters the terms Tb, bronchoscopy, and Tb treatment.

Gramarye begins by setting the host graph to the initial graph as declared in the static input file (see Section 3.6.3), which happens to be a single utility node, Value to patient. The system then finds the *most specific* graph-grammar production that might add nodes with the labels Tb, bronchoscopy, and Tb treatment. The most specific rule for a given term is that rule with the most specific nonterminal symbol for that term on the rule's V_R region. Of the three terms entered, only Tb can be added to the initial graph, since the other two terms require more of the host graph. At this point, Gramarye adds Tb to the graph, as shown in Figure 3.17.[7] Now that both disease and utility nodes are present in the host graph, Tb treatment can be added (Figure 3.18). The only remaining term, bronchoscopy, can now be added, as shown in Figure 3.19. Since no more terms need to be added, the derivation system returns the textual form of the QCID model and the corresponding diagram, as shown in Figure 3.20.

[7]If there were other diseases listed, they would also be added now, since, at a particular stage in the derivation, the system applies all rules that *can* be applied before examining the resulting graph to see what new terms can then be added. This strategy depends on the grammar to ensure that, if there is no determinism to the ordering of a set of graph manipulations, then that set of manipulations must have the same result, regardless of the order of application. Ehrig and Kreowski (1980) show a simple way to examine graph-grammar rules and to determine whether the latter are sequentially independent and, by Church–Rosser's theorem, parallel independent. In Section 4.3.1, I discuss how I use this result to avoid ambiguity in the grammar.

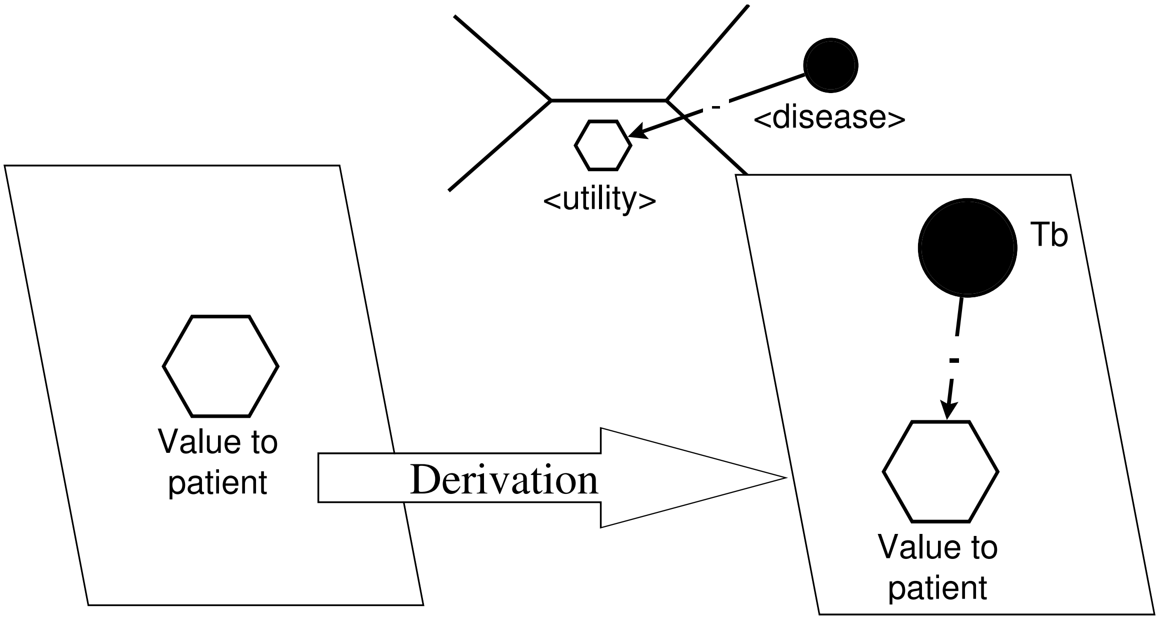

Figure 3.17: Addition of a node labeled Tb to the initial graph. According to the grammar rule shown on top, the only requirement for the incorporation of a disease node is the existence of a utility node. Gramarye adds the appropriate chance node and negative qualitative arc.

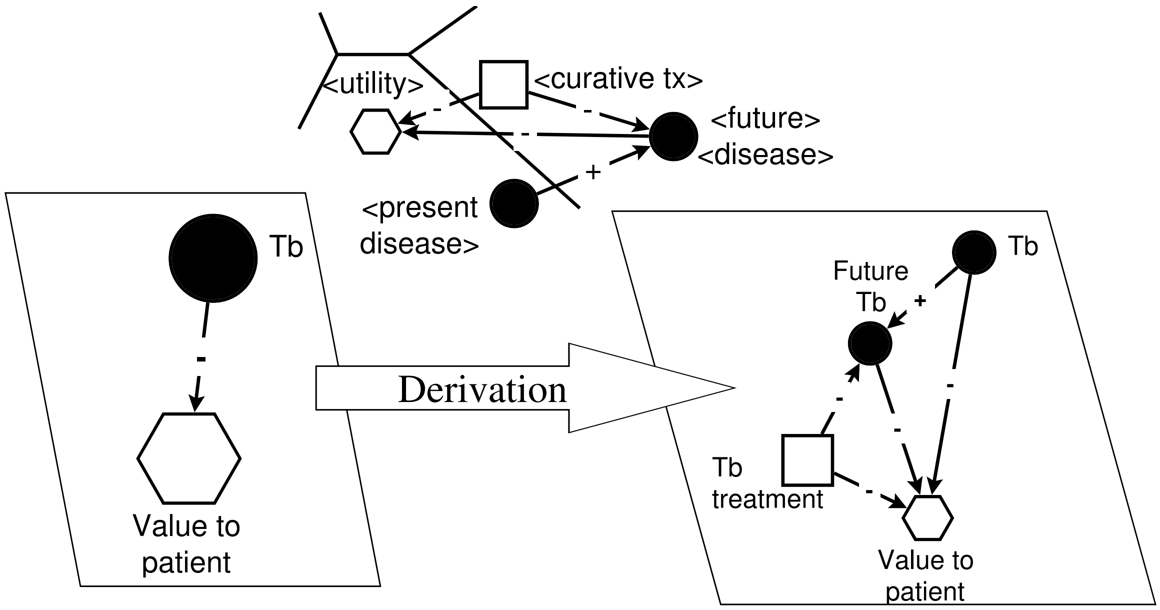

Figure 3.18: Addition of a node labeled Tb treatment to the host graph at the first intermediate stage of the development. The requirements of the grammar rule shown on top are that both utility and disease nodes are present in the host graph. In addition to the treatment decision, a chance node labeled Future Tb is added to the host graph in accordance with the grammar rule's right-hand region.

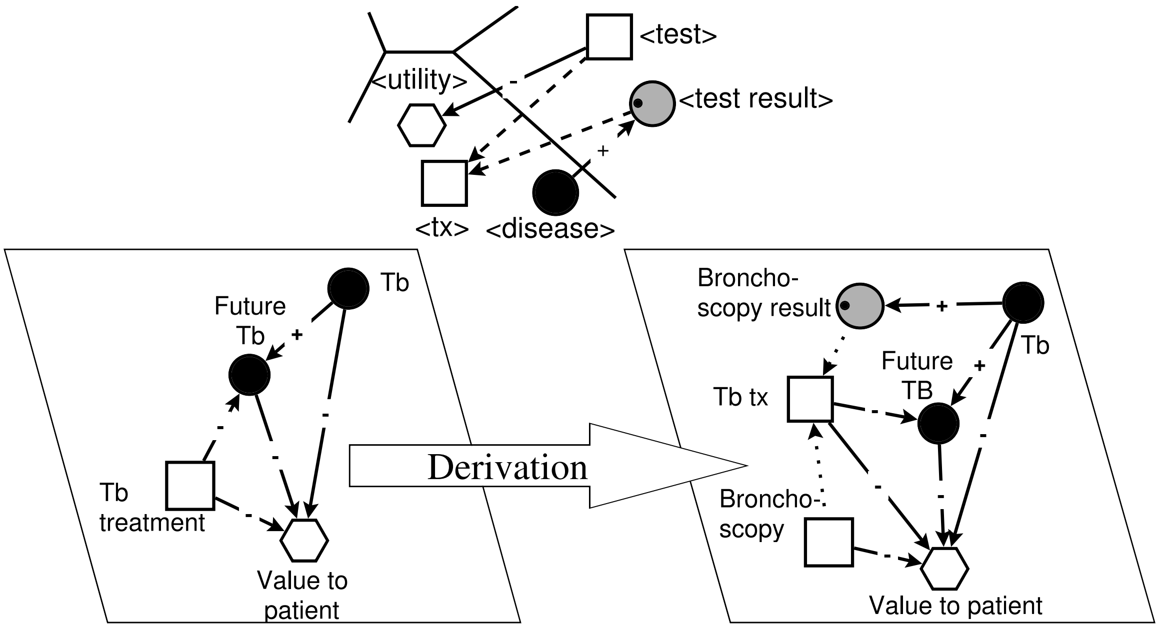

Figure 3.19: Addition of a node labeled Bronchoscopy to the host graph at the second intermediate stage. For simplicity, the contingent arcs from the testing decision to the test result is represented as a solid dot on the left side of the ⟨test result⟩ and the Bronchoscopy result nodes.

3.8 Gramarye's User Interface

I have designed Gramarye's current user interface with the developer in mind. The command-line interface is essentially a Lisp listener, a term browser, and a panel of buttons for routine commands used by Gramarye (Figure 3.20). The system traps any generated diagrams in a separate window, in which the user can edit the data structure. The Graph It button invokes the appropriate translation to a textual format used by the graph editor, and opens a graph-editor window with the diagram inside. The current graph editor is a commercially available tool.[8]

3.9 Assessment and Refinement of the Decision Model

As noted earlier (Sections 1.3 and 3.5), the QCID model returned is insufficient to allow a decision maker to arrive at the optimal plan for a given decision problem. Utilities and probabilities must be assessed. In most cases, after the initial model has been assessed, it will still require refinements and testing before any firm conclusions are warranted. An obvious extension to Gramarye would be the completion of the decision-analysis cycle with other tools for assessment and testing. Currently, Gramarye can store the edited model in HUGIN (Andersen et al.,

[8]Diagram!, from Lighthouse Design, Inc., San Mateo, CA.

1989), Ergo (Beinlich and Herskovits, 1990), CABeN (Cousins et al., 1992), Demos,[9] INFER,[10] SPI (D'Ambrosio, 1989), or IDEAL (Srinivas and Breese, 1990) format.[11] The user can then perform the subsequent assessment, testing, refinement, and inferential queries on the decision model. As new salient considerations arise, and as other considerations—for which the choice of the optimal plan is insensitive—are removed, the user may repeat this development cycle by entering the updated list of considerations into Gramarye, and by making the indicated changes to the assessed model. Current limitations of the grammar and the derivation system are discussed in Chapter 5.

3.10 Summary

Gramarye is a graph-grammar–based derivation system that constructs QCID models from a list of medical concerns. It begins with a single utility node, and adds nodes and arcs in accordance with the graph grammar until either the entire list of concerns has been included, or the system cannot add any more nodes. Gramarye recognizes only those terms that are listed in its node-label classification hierarchy; any other terms must be added to the classification by the user. The resulting model

[9] Lumina Decision Systems, Palo Alto, CA.

[10] Adam Galper. Personal communication.

[11] Although HUGIN, Ergo, CABeN, and INFER are designed for belief-network inference, such routines can be used to solve influence-diagram queries as well (Cooper, 1988; Shachter and Peot, 1992).

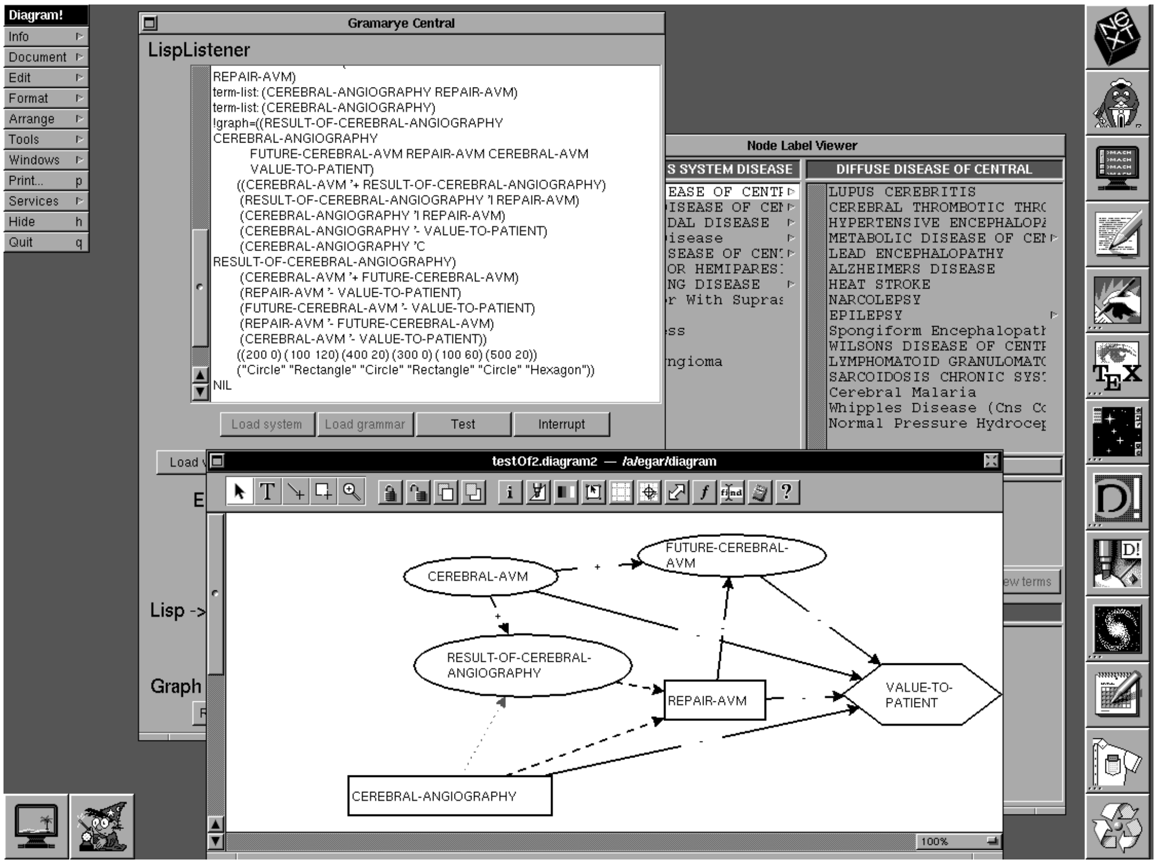

Figure 3.20: The current user interface to Gramarye. In the top line of the Lisp listener—a line which is no longer visible in the scrolled view, the user has typed "(derive '(tb t-test tb-treatment))." The derivation system traces its progress as it adds nodes for each of these three terms. Then, the system returns both the textual representation of a QCID model and the actual diagram, which appears in a separate diagram-editor window.

is qualitative; the user must assess quantitative probabilities and utilities to arrive at a normative decision. Furthermore, decision analysis generally requires testing and revision of an initial, putative decision model, and the model produced by Gramarye is no exception to this rule.

Chapter 4

Properties Desired of the Grammar

> Figures are not only the object of geometric problems, but also an important help for all sorts of problems in which there is nothing geometric at the outset.
>
> —*George Polya, How to Solve It*

Although much of the information conveyed by a decision model may be specific to the given domain—here, medicine—there are several structural features of influence-diagram models of dilemmas that, if violated, would be seen as errors by a decision analyst—even one who is not familiar with the domain. Although it may be simple enough to check that any one of these properties holds in an individual model, such a check does not help us to correct the fault, and so the user must do the initial modeling when the system yields an unreasonable influence diagram. The graph grammar used to derive models ought to maintain these minimal requirements for credibility. Also, as each desirable property becomes equated with a general constraint on graph grammars, the job of designing high-quality graph grammars becomes easier, and our understanding of decision-model topologies is furthered.

Prior to entering into Gramarye any specific considerations that are to be included in a QCID model, the user must install the **static input elements**—the graph-grammar productions, the node-label hierarchy, and the initial graph—which together define the domain. This collection of syntactical manipulations, hereafter referred to as the **grammar,** may enforce certain properties on all resulting derived graphs. Gramarye has a separate module that can be used by the designer of a grammar to check whether the grammar maintains these properties.

Most of the checking algorithms that Gramarye uses are sufficient, but they are not always necessary. That is, there may be grammars that fail my checks but that still enforce these properties for all derived graphs. However, even though the checks on a grammar are stronger than they need to be, this difference has not been a significant hindrance to the design process in my experience.

Throughout my description of these grammar checks, I shall need to refer to sets of node labels—either classes or instances—that are generalizations or specializations of a particular class in the node-label hierarchy. For a given node label, I shall refer to the labels above it in the hierarchy as **ancestors,** and to the labels below it—including terminal symbols—as **descendants.** Since these labels define classes of nodes in an influence diagram, and classes of vertices in a grammar, I shall use the same terms—**ancestors** and **descendants**—for sets of nodes and vertices. Since the node-label hierarchy is a tree, two production vertices may be instantiated by the same terminal node only if one vertex is the ancestor or descendant of the other. I shall denote this subclass–superclass relationship as **anc-desc-equivalence.** For two sets of vertices, the **anc-desc-intersection** consists of those vertices of the first set for which there exists an anc-desc-equivalent vertex in the second set.[1]

The properties for which Gramarye checks in a grammar can be divided into three groups: (1) basic computational requisites for computing an optimal decision from an assessed influence diagram, (2) domain-independent properties that I expect in any reasonable decision model, and (3) guidelines for perspicuous graph grammars. In Sections 4.1 through 4.3, I discuss what the properties in these three groups are, and how Gramarye checks a new grammar to ensure that the latter maintains such properties.

[1] Although I have arbitrarily chosen the vertices of the first set to be represented in the intersection, this choice has little importance for the grammar checking that Gramarye performs.

4.1 Basic Computational Requisites

So that a decision maker can evaluate the model and thereby decide what to do, I require an influence diagram to be **oriented** to a single utility model, and **regular,** as defined by Shachter (1986). These requirements correspond to the following properties, which should be maintained by any QCID-generating grammar used with Gramarye:

1. There should be exactly one overall utility node.

2. There should be no successors to the utility node.

3. The directed graph should be acyclic.

4. All decision nodes should be completely ordered by the directed graph.

In Sections 4.1.1 through 4.1.4, I discuss how Gramarye checks whether any QCID grammar maintains these computational properties for all graphs that are derived using that grammar.

4.1.1 Unified Utilities

When a decision maker evaluates an assessed decision model to find a single optimal plan, she requires a function that measures the expected utility of each possible scenario of outcomes. This restriction can be relaxed if the goal is only to view the expected utility of each decision alternative, and if the user does not mind viewing utilities as multidimensional vectors. However, the basic precepts of decision theory require that the decision maker be able to order completely her preferences for the various outcomes. It is unclear how this preference ordering should be done when the end utilities are left as vectors, so I would like to restrict the models that Gramarye generates such that they contain exactly one utility node.

To see that a grammar maintains this property, Gramarye needs only to note that there does not exist a graph-grammar production that adds or deletes a utility node into the emerging model, and that the initial host graph consists of a single utility node.

4.1.2 No Successors to Utility

The requirement that there be no successor to the utility node is a convenience for assessment. As long as the sets of immediate predecessors for two adjacent nodes are identical, then the arc between them can be reversed. A decision analyst *could* assess factors that contribute to the overall utility of the decision maker by estimating the probability distribution of a contributing factor for each value of the overall utility, but this approach to utility assessment is generally awkward. Consequently, I prefer grammars that restrict the models derived to those in which the utility node has no outgoing arcs.

Gramarye checks that this criterion is maintained in all derived models of a grammar by checking that the initial graph contains no arcs that leave a utility node, and that there does not exist a rule where an outgoing arc is added to a utility node.

4.1.3 Acyclicity

The requirement that the arcs in influence diagrams do not form a directed cycle distinguishes the influence-diagram notation from earlier graphical notations of probabilistic dependence (Rousseau, 1968). The reasons for the restriction are as follows:[2]

- If two or more decision nodes are involved in a cycle (Figure 4.1a), then the informational arcs imply that each decision is made *before* all the others in the cycle. This conundrum prevents us from finding any temporal sequence that would allow alternatives from both decision nodes to be chosen freely and nonsimultaneously.

- If chance and decision nodes are involved in a cycle (Figure 4.1b), then we lose the presumed relevance or influence that the decision node has on the chance node's outcome, since some information about the outcome is known at the time of that decision, according to the informational arc whose target is that decision node.

[2]Ross D. Shachter. Personal communication.

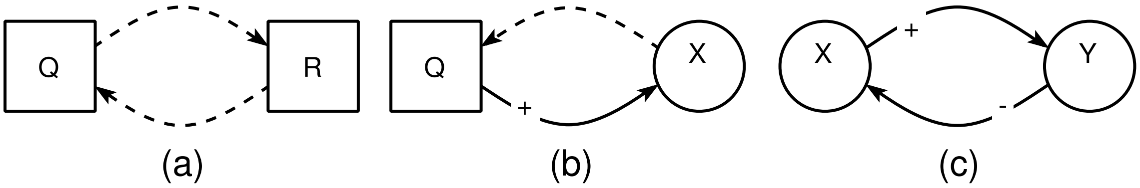

Figure 4.1: Cycles prohibited in the influence-diagram representation. (a) Two decision nodes in a cycle. Such an arrangement prevents a total ordering of decisions. (b) A cycle between decision and chance nodes. Here, the decision is fated, since information regarding the decision's effect is available prior to the decision. (c) A cycle involving two chance nodes. In this example, suppose X and Y are Boolean variables, and suppose that the conditional probabilities are assessed to be $\mathrm{P}(X|Y) = 0$, $\mathrm{P}(X|\bar{Y}) = 1$, $\mathrm{P}(Y|X) = 1$, and $\mathrm{P}(Y|\bar{X}) = 0$. Although the conditional probabilities meet the qualitative constraints of the influence diagram, they do not correspond to a consistent joint probability, because the conditional probabilities are contradictory.

- If two or more chance nodes are involved in a cycle (Figure 4.1c), then there is the possibility that independently assigned conditional probabilities for the two nodes will result in an indeterminate or inconsistent joint probability distribution.

The procedure that Gramarye uses to check that a grammar will never generate a cycle relies on the previous two properties: If there is exactly one utility node, and if that utility node can never have an outgoing arc, then there can be no directed cycles that include the utility node. Since, in Göttler's nodecentric formalism, no arc additions are ever made without the addition of the target or source node, the derivation system can create a new cycle only by adding one or more nodes with both incoming arcs and outgoing arcs. Consequently, Gramarye begins the procedure by creating a set $\mathcal{S}$ of sinks—node-label classes that can never be involved in a directed cycle. Initially, the utility node is the sole member of $\mathcal{S}$. Gramarye then repeats the following routine until no more classes are added to $\mathcal{S}$:

For each class that appears in some V_{R},

if all rules that can add anc-desc-equivalent nodes of that class

can add no outgoing arcs to any node outside of $\mathcal{S}$,

and outgoing arcs of anc-desc-equivalent classes in the initial graph

go to no node classes outside of $\mathcal{S}$,

then add that class to $\mathcal{S}$.

If the initial graph has no cycles, and if there are no anc-desc-equivalent classes of vertices in V_{R} with both outgoing and incoming edges for which the grammar can add an outgoing arc to a class outside of $\mathcal{S}$, then the grammar ensures that no derived graph will contain a cycle.

4.1.4 Complete Ordering of Decisions

The last of the five computational requisites—complete ordering of decisions—is one that I do not currently ensure in models generated from unordered lists of terms. For example, if the list contains multiple tests, then there is no a priori reason to decide to order one test before I order the others. Zhang and colleagues (1993) have suggested that there are situations in which decision models do not require complete ordering of the decision nodes. However, since the user may have domain-specific and situation-specific reasons for a particular ordering, I have not attempted to maintain this requisite in all graphs that Gramarye generates.

4.2 Domain-Independent Properties of Reasonable Decision Models

In addition to the aforementioned computational requirements, there are several properties that characterize reasonable QCID models of decision problems. Although they may be violated in a valid QCID model, I would prefer that Gramarye alert the user whenever her input considerations lead to such violations. These properties are domain independent, and rely on only the rationality of the decision maker, and the pertinence of the terms that she lists to the decisions she faces. I accept that terms may be included in or omitted from the initial list by mistake, but I would prefer the derivation system to notify the user of such problems, rather than to derive a model that either is not suitable for remaining stages of decision analysis or does not require further analysis for all the decisions listed. In particular, I would like my grammar to maintain the following properties of reasonable decision models:

1. There are no qualitatively dominated decision nodes.

2. All nodes have at least one chain to the utility node.

3. All chance nodes have at least one chain to the utility node with no intervening decision nodes.

4. There exists at least one decision node.

5. There exists at least one uncertainty, either probabilistic or deterministic.

To test a grammar for several of these properties, Gramarye builds a **staging graph**—a graph of stages in any derivation that Gramarye might perform using that grammar. To create the staging graph, Gramarye begins with a node that represents the initial graph, and a node for each production in the grammar (Figure 4.2). Gramarye adds directed arcs from the node representing the initial graph to all nodes whose productions are applicable to the initial graph. A node is regarded as **applicable** to the initial graph if all nodes and edges in the bottom and left region of the node's production can be matched to nodes and edges that are either in the initial graph or in the right region of productions belonging to an earlier stage (see Appendix A.2). Then, for each pair of productions in which there is a nonempty anc-desc-intersection of V_{R} of the first production and V_{BL} of the second, Gramarye adds a directed arc from the first node to the second. All immediate successors to the initial graph's node are grouped as the first stage. All successor nodes of the first stage that may be applicable after the first stage are grouped in the second stage, and so on, until a stage contains no successors, or the following stage duplicates a previous stage precisely. Nodes in this procedure can belong to more than one stage.

If cycles occur in a staging graph, and if these cycles occur prior to the inclusion of all production rules into at least one stage, then Gramarye rejects the grammar. Since, for any finite grammar, only a finite number of distinct stages—i.e., combinations of rules—are possible, Gramarye's construction of a staging graph is guaranteed to terminate. In practice, the staging procedure terminates long before this theoretical limit. For example, the grammar in Section 3.6.1 yields a staging graph with only five discrete stages. In Sections 4.2.1 through 4.2.5, I discuss these domain-independent properties.

4.2.1 Lack of Qualitative Dominance

I use the term **qualitative domination** to refer to a decision node that does not have both a positive and a negative qualitative chain to the utility node. A **positive qualitative chain** is a sequence of arcs, disregarding each arc's direction, where the overall direction of influence, according to sign algebra (Wellman, 1990b), is positive.[3] Informational arcs can be considered positive in this sign algebra. Contingency arcs can be positive or negative. For tests, the target node of the contingency arc—⟨test result⟩—relies on the test being performed, so the arc is positive. For ablative treatments, the target node of the contingency arc—⟨future disease⟩—is contingent on the treatment *not* being performed, so the arc is negative. Uncertain or nonmonotonic

[3]The sign of a chain is equal to the sign of the product of factors assigned to each arc along the chain, where the factor 1 is assigned to each positive arc, and the factor -1 is assigned to each negative arc.

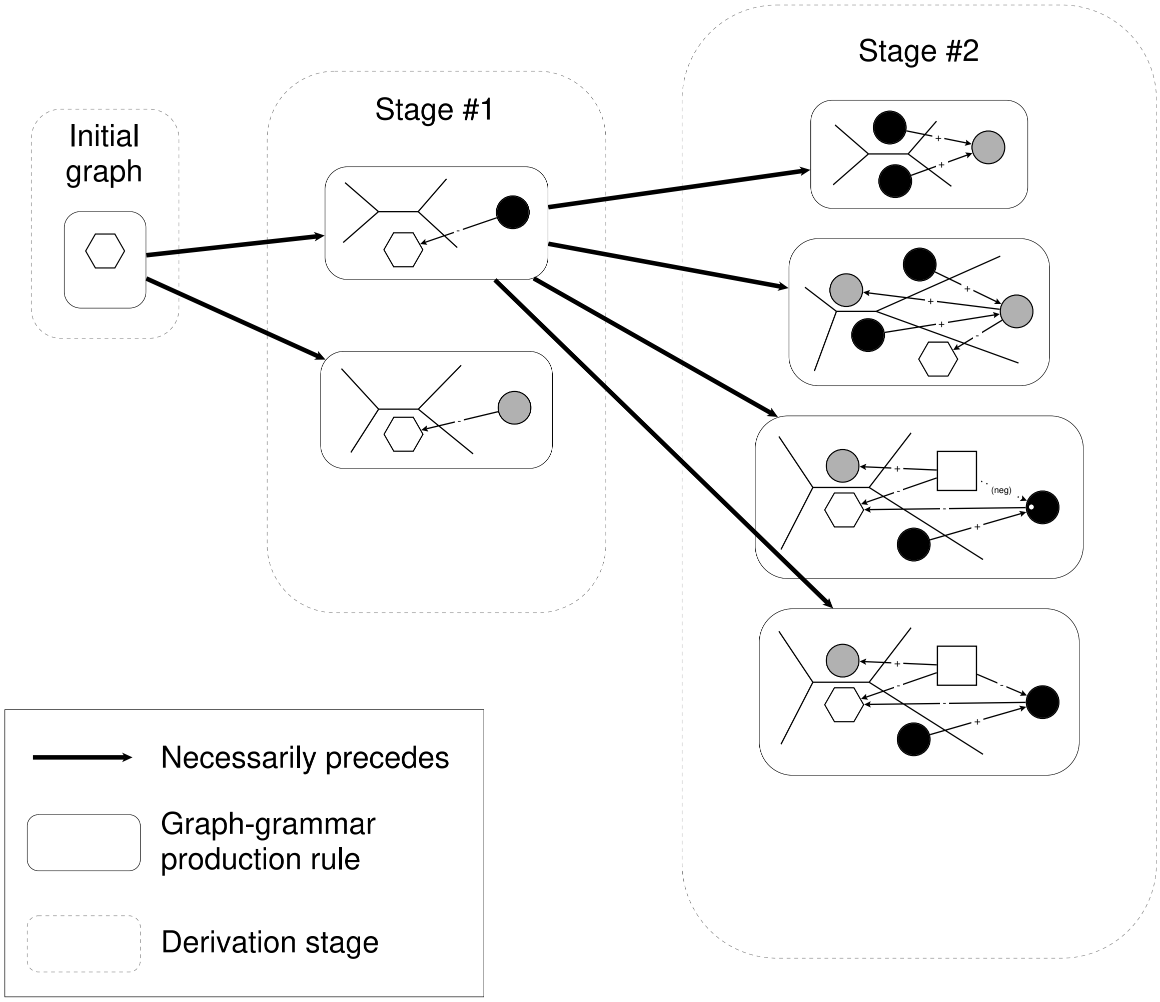

Figure 4.2: A staging graph for a graph grammar. Nodes represent graph-grammar production rules in the grammar. Arcs denote the precedence ordering that is dictated by the left and bottom regions of the production rules. Stages group those production rules that Gramarye may apply simultaneously during a derivation. Gramarye generates such staging graphs to test a grammar for various properties. For example, if the node representing a production rule is not connected to the initial graph by any path, then that production can never be applied (Section 4.3.3.

arcs can assume both positive and negative signs in any combination that might avoid qualitative dominance.

To determine whether a grammar prevents qualitative dominance, Gramarye creates the set $\mathcal{D}$ of deletable classes—the anc-desc-equivalent set of classes appearing in V_{R} of at least one production. Gramarye also creates four empty sets, $\mathcal{P}$, $\mathcal{N}$, $\mathcal{B}$, and $\mathcal{E}$, which are those classes that have been shown to have qualitative chains between such nodes and the utility node that are, respectively, *positive, negative, both* positive and negative, and *either* positive or negative, depending on the input list and resulting derivation. Gramarye then performs the following routine for each stage of the staging graph:

For each class that appears in some V_{R} at that stage,

  if that class is added at no subsequent stage,

    and,

    for each production at that or prior stage that adds it,

      the class is added with a $+$ chain—outside of $\mathcal{D}$—to either

        a utility vertex, or a class in $\mathcal{P}$ or $\mathcal{B}$,

      or

      the class is added with a $-$ chain—outside of $\mathcal{D}$—to

        a class in $\mathcal{N}$ or $\mathcal{B}$,

  then set $P(\text{class}, \text{stage}) \leftarrow \top$.

If that class is added at no subsequent stage,

and,

for each production at that or prior stage that adds it,

the class is added with a $-$ chain—outside of $\mathcal{D}$—to either

a utility vertex, or a class in $\mathcal{P}$ or $\mathcal{B}$,

or

the class is added with a $+$ chain—outside of $\mathcal{D}$—to

a class in $\mathcal{N}$ or $\mathcal{B}$,

then set $N(\text{class}, \text{stage}) \leftarrow \top$.

If P and not N, then add that class to $\mathcal{P}$.

If not P and N, then add that class to $\mathcal{N}$.

If P and N, then add that class to $\mathcal{B}$.

If not P and not N, then add that class to $\mathcal{E}$.

If all vertices of the decision-node class that appear in right side of some production are in the set $\mathcal{B}$, then the grammar ensures that no qualitative dominance will appear in any derivation.

4.2.2 Relevance to Utility

Unlike decision trees, influence diagrams represent explicitly which pairs of variables are directly dependent on each other. Requiring that all nodes have some chain to the utility node is equivalent to requiring that the diagram correspond to a connected graph. Assumptions of conditional independence are made explicit by absent arcs. The criterion that all nodes be connected ensures that the manner in which each pertinent variable affects the dilemma is stated clearly.

Gramarye checks that all graphs derived using a grammar will be connected by first fixing the set $\mathcal{D}$ of deletables to those classes whose anc-desc-equivalents appear in V_{L} of some production, and then checking that the initial graph contains a utility node, that no utility nodes are deleted, and that the initial graph is **permanently connected**—that is, all other nodes in the initial graph have some chain to the utility node with no intervening nodes of that chain in $\mathcal{D}$. Gramarye initializes the set $\mathcal{R}$ of rooted classes with the utility class. Then, Gramarye repeats the following procedure until no new classes are added to $\mathcal{R}$:

For each class c that appears in some V_{R},

  if all productions that have an anc-desc-equivalent vertex

    of class c in V_{R} include,

    with each such vertex,

      an edge to or from a vertex in V_{B} that is of a class that is anc-desc-equivalent to a class in $\mathcal{R}$

and

is not anc-desc-equivalent to a class in $\mathcal{D}$,

then add class c to $\mathcal{R}$.

If all classes that appear in some V_{R} of the grammar are members of $\mathcal{R}$, then the grammar ensures that all graphs derived will be connected, with all nodes having some chain to the utility node.

4.2.3 Probabilistic Chain from Chance Nodes to the Utility Node

A chance event that has no bearing—direct or indirect—on the decision maker's utility function should not be considered. Consequently, the grammar should not create any chance nodes that do not have a **relevance chain**—a sequence of contingency and relevance arcs with no restriction on arc direction—to the utility node.

To check that a grammar ensures that all chance nodes will have at least one decision-free chain to the utility node, Gramarye first sets $\mathcal{D}$ to those deletable classes whose anc-desc-equivalents appear in V_{L} of some production. It then checks that the initial class contains a utility node, that no utility classes are in $\mathcal{D}$, and that all chance nodes in the initial graph have a chain to the utility node, where the chain contains no decision nodes and no nodes of classes in $\mathcal{D}$. Then, Gramarye initializes the set of classes rooted by nondecisional chains, $\mathcal{C}$, to include the utility node, and repeats the following routine until no new classes are added to $\mathcal{C}$:

For each class c of type Chance that appears in some V_{R},

  if all productions that have a

    vertex that is anc-desc-equivalent

      to c in V_{R} include,

    with each such vertex,

        an edge to or from a vertex in V_{B} that is

          of a class that is anc-desc-equivalent to a class in $\mathcal{C}$

          and

          is not chain to a class in $\mathcal{D}$,

  then add class c to $\mathcal{C}$.

If all chance vertices that appear in some V_{R} of the grammar are also members of $\mathcal{C}$, then the grammar ensures there will always be a probabilistic chain from each chance node to the utility node.

4.2.4 Existence of Decision Nodes

For a decision problem to exist, there must be at least one decision to be made. Gramarye does not check grammars for this property, and the current grammar does not enforce this property. Because the input list of terms is unordered, the current derivation scheme cannot rely on decision nodes being added at a later stage of a derivation. I would prefer that the grammar ensure that such properties hold at every intermediate step in the derivation, but it does not do so currently.[4] However, it is a trivial operation to check that entered lists of considerations contain at least one term that corresponds to a decision node, thus ensuring that all derived models contain decision nodes.

4.2.5 Existence of Chance Nodes

The requirement that there be at least one chance or deterministic node merely amounts to a requirement that I *not* be able to solve the decision by simply ordering the utilities of each decision alternative (or scenario of decision node choices). Currently, Gramarye does not check a grammar to ensure that this property is maintained. However, since all rules in the initial stage of the grammar given in Figures 3.11 through 3.14 add chance nodes that cannot be deleted by any production

[4] An odious kludge to ensure that every model has at least one decision would involve setting the initial graph—currently just a utility node—to include a default decision, and deleting any such default in every production that might add the initial decision to the diagram.

in the grammar, I see that all derivations that are different from the initial graph will have at least one chance node.

4.3 Guidelines for Perspicuous Graph Grammars

Several of the properties for which Gramarye checks in a new grammar are intended to support the clarity of the grammar itself. Derivations should not surprise the designer of a grammar, and should be predictable in the absence of any automated derivation system. Just as a computer program should not have unused variables and inaccessible routines, a grammar's node-label hierarchy and the graph-grammar productions should reflect a single coherent design. Consequently, Gramarye checks a grammar for the following properties:

1. Other than the derivation staging that is inherent in a grammar, the order of production application has no effect on the graph generated.

2. All productions can be understood by a designer without resort to staging analysis.

3. Each production in the grammar is potentially applicable for some input.

4. Each class has the potential to be represented in some derivation.

In Sections 4.3.1 through 4.3.4, I discuss what these properties are and how Gramarye checks to see that a grammar maintains them.

4.3.1 Order Independence

A partial ordering of production application is inherent in the grammar and in the stipulation that any derivation system should attempt to apply all rules that are applicable to the current host graph before looking at new rules that are now applicable with these additions. As in Section 4.2, I refer to this ordering as **staging,** and I use the term **stage** for the collection of rules that introduce new terms and are applicable to the same host graph. The order in which rules are applied *within the same stage* should not concern the designer or user of a grammar, since such knowledge would depend on a detailed understanding of the derivation system's implementation. Consequently, such order independence is a prerequisite for implementation independence, where a given grammar and a given set of user input will result in the same generated model for any implementation of a staged derivation system, whether that implementation is Gramarye or some other program.

One advantage of the graph-grammar formalism is that it provides a simple method for ensuring that the order in which two productions are applied has no effect on the result of their application: If (1) each vertex in the anc-desc-intersection of productions p_1 and p_2 is in the embedding environment, and (2) there is no common edge between p_1 and p_2, where the edge has a source or target outside the embedding environment in at least one of the productions, then the two productions maintain both parallel and sequential order independence (Ehrig and Kreowski, 1980). Hereafter, I shall refer to condition 1 as **vertex disjunction,** and to condition 2 as **edge disjunction.** These two conditions are sufficient, although not necessary, to show that order independence holds for any pair of graph-grammar productions.

To check that a grammar maintains vertex disjunction for all productions within each stage of any derivation, Gramarye first checks that each of the following holds:

1. There are no deletions (V_{L}) that disallow productions (V_{L},V_{B}) in the same stage.

2. There are no deletions (V_{L}) that appear in the indeterminate region (V_{A}) of rules in the same stage.

3. There are no additions (V_{R}) that are repeated by different and equally specific productions in the same stage.

These three conditions are entailed by two checks on the rules of each stage: There is no anc-desc-intersection between V_{L} of one rule and V_{L}, V_{A}, or V_{B} of another rule, and there is no anc-desc-intersection between V_{R} of one rule and V_{R} of another rule. With these checks, Gramarye shows that there is no anc-desc-intersection of any V_{L} with the vertices of other productions in the same stage, and that there is no anc-desc-intersection of any two V_{R} regions of productions in the same stage.

Gramarye then checks within the each stage of the staging graph to see that there are no additions (V_{R}) that alter optional arc additions (to or from V_{A}), and that there are no additions (V_{R}) that alter possible rule application areas. It does so by checking that the anc-desc-intersection of the V_{R} vertices and the V_{L}, V_{A}, and V_{B} vertices is empty. If there is a nonempty V_{R}–V_{LAB} anc-desc-intersection, then Gramarye cannot show *strict* vertex disjunction. However, Gramarye then checks whether no optional edges—between V_{R} and V_{A} or between V_{L} and V_{A}—exist in a stage that adds nodes of the type of the vertex in V_{A}, unless that same arc can be added *identically* by the productions with the V_{A} vertex in their V_{R} region. It performs this check by checking that any V_{A}–V_{R} anc-desc-intersection is accompanied by a symmetrical V_{R}–V_{A} anc-desc-intersection. This latter criterion is weaker than is the criterion for *strict* vertex disjunction. It relies on consistent user assistance to prevent order dependence. Gramarye's current implementation obviates the need to worry about V_{R}–V_{ABL} intersection causing intrastage order dependence, since the matching monomorphism is instantiated for all applicable rules prior to the application of any one of the rules during a stage. So, for example, my current grammar checker would permit the productions in Figure 3.11(b) and (d) to occur in the same stage.[5] However, for *strict* implementation independence, the former criterion—an empty V_{R}–V_{LAB} anc-desc-intersection—should be maintained.

To check that a grammar maintains edge disjunction for all productions within each stage of any derivation, Gramarye checks that no production has an edge that appears as an anc-desc-equivalent edge in another production with either the source

[5]In fact, these two productions do not occur in the same stage.

or target vertex in V_{R} or in V_{L}. If this condition does not hold, then Gramarye checks that no production has an edge that appears as an anc-desc-equivalent edge in another production with either the source or target vertex in V_{L}. The latter criterion relies on Gramarye performing a complete matching for the application of a rule prior to the actual application process for any rule at the same stage. Since, for Gramarye, no arcs that are added during a stage can be used in the imbedding environment of another rule to be applied in the same stage, intersections between edges with one vertex in V_{R} and edges with both vertices in V_{LAB} of another production do not result in intrastage order dependence for Gramarye. Intersections between edges with one vertex in V_{R} and edges with one or more vertices in V_{R} also do not result in intrastage order dependence for Gramarye, since the nonvariant vertices in V_{R} of two rules, or in V_{R} of two applications of the same rule, necessarily have as their labels two different considerations from the user's list. Variant vertices are wholly determined by the V_{LB} matching of a rule application in Gramarye. Once again, for *strict* implementation independence, the former criterion should be maintained, although the latter criterion is sufficient to show intrastage order independence when Gramarye is used as the derivation system.

4.3.2 Declarative Transparency

Several of the properties for which Gramarye checks in a grammar have more to do with the clarity of the grammar than with the quality of the derived graphs. Specifically, when a rule specifies some set of vertices and edges in the embedding environment, that embedding environment should represent *all* such nodes and arcs that might be included at any stage of the derivation of a model. Consequently, Gramarye checks that a grammar has the following characteristics:

1. There is no anc-desc-intersection between added edges in one production and edges contained in the embedding environment of another production at the same stage, unless the second production appears at a later stage.

2. There is no anc-desc-intersection between vertices in V_{A} and vertices in V_{R} of subsequent stages, unless the edges between V_{A} and V_{R} in the first production appear between V_{R} and V_{A} in the second production.

Also, so that inhibitory effects of node deletion are not hidden within the grammar, Gramarye requires that productions with a nonempty V_{L} allow whatever nonvariant nodes they introduce into a graph to be added by a related production at a later stage. To ensure this property, Gramarye checks that

1. There are no deletions that disallow other rules at subsequent stages.

2. There are no deletions that do not pair with a **compensatory** production that adds the same class of nonvariant nodes, and that has an empty V_{L}.

Here, a production p_2 is defined by Gramarye to be **compensatory** to a nonempty-V_{L} production p_1 if

1. The nonvariants in V_{R_1} are identical to the nonvariants in V_{R_2}.
2. At least one vertex is in both V_{R_1} and V_{B_2}.
3. No anc-desc-equivalent class of that vertex appears in a V_{R} of the same stage as V_{B_2} or of a previous stage.
4. The subgraph of vertices and edges in the right and bottom regions of p_1 is identical to the subgraph in the right and bottom regions of p_2.

For example, if the addition of a node of type X requires the deletion of a node of type Y, then there should be a subsequent compensatory rule. This compensatory rule adds X nodes in the same manner, just as though the Y node was present and the first rule was being used. This sort of compensation appears in the two productions shown in Figure 3.14(a) and (b). The first production, Figure 3.14(a), adds only the first node of the type ⟨risk-reducing tx⟩ that pertains to a particular crisis. All other risk-reducing treatments that apply to the same crisis are added by the production in Figure 3.14(b). The first production's unique creation of a ⟨future crisis event⟩ variant node ensures that the second production must follow the first, and must be used only when the crisis event is reduced by both treatments.

4.3.3 Applicability of Each Production

Clarity in a grammar is affected adversely when there are rules that cannot be used in *any* derivation. To prevent this situation, Gramarye checks that all rules appear in at least one stage of the staging graph (see Section 4.2).

4.3.4 Potential for Inclusion of Each Class

As a warning for grammar designers whose graph-grammar productions cannot incorporate concepts of known classes into a model, Gramarye checks whether all classes have some corresponding rule that includes those classes in a derivation. The check that Gramarye performs is to examine whether all leaf classes have an anc-desc-equivalent class in the V_{R} of at least one rule. Gramarye then warns the user of the least-specific classes that are not included in any V_{R}. Although my grammars often violate this property and I ignore the warning, I still believe that this check performs a valuable bookkeeping service for the designer of a grammar.

4.4 Properties for Further Investigation

The preceding properties, which my current grammar maintains, are not sufficient to guarantee that the decision models that Gramarye derives will be acceptable to the user. I also do not mean to imply that the preceding properties are the only ones that are desirable for decision models. As researchers find other traits that distinguish acceptable models from unacceptable ones, they may develop similar graph-grammar checks to ensure that those traits are always maintained by a derivation system.

4.5 Summary

As demonstrated by Gramarye's grammar-checking procedures, the graph grammar that Gramarye currently uses maintains the following properties in all derived QCID models:

1. There is exactly one overall utility node.

2. There is no successors to the utility node.

3. The directed graph is acyclic.

4. There are no qualitatively dominated decision nodes.

5. All nodes have at least one chain to the utility node.

6. All chance nodes have at least one chain to the utility node with no intervening decision nodes.

7. Other than the derivation staging that is inherent in a grammar, the order of production application has no effect on the graph generated.

8. All productions can be understood by a designer without resort to staging analysis.

9. Each production in the grammar is potentially applicable for some input.

10. Each class has the potential to be represented in some derivation.

The grammar does not ensure that there is at least one decision node and at least one uncertainty, or that decision nodes are completely ordered.

Also, the grammar guarantees that any set of input that derives a model will derive a unique model, regardless of the order in which rules are applied by the derivation system. Here, I use the term input to refer both to the unordered list of considerations and to the choices made by the user during the application of rules that have indeterminate environments. Thus, the current grammar is, in a sense, unambiguous.

Chapter 5

An Evaluation of Gramarye

Est modus in rebus. Sunt certi denique fines quos ultra citraque nequit consistere rectum.

Things have their due measure; there are ultimately fixed limits, beyond which, or short of which, something must be wrong.

—*Horace*

The central hypothesis of this dissertation—that graph grammars can be used to generate decision models—is based on my experience over the past 15 months with an implemented graph-grammar derivation system. Any single derivation provides proof that this hypothesis is true—to a degree and under qualified conditions. However, the successful application of this technology to improving medical decisions depends on degrees and qualifications: How much benefit can the technology provide to patients, clinicians, and other health-care workers? For what problems is this technology likely to be useful? How easily can the technology be integrated into other systems?

For Gramarye, the process that is being automated—the composition of QCID models—is performed only rarely, and then by only a few researchers. The subsequent steps of the decision-analysis cycle are also the subject of academic research, and the informational demands of formal decision analysis are beyond what current hospital information systems can provide. Although several professional consulting companies analyze corporate decisions, individual clinical decisions usually cannot justify the expense of decision analysis. Moreover, clinicians and patients lack the necessary familiarity with decision-theoretic tools and terminology.

Consequently, the primary goal of my evaluation of Gramarye has been to describe its strengths and weaknesses along five dimensions:

1. The scope of problems for which Gramarye is applicable
2. The interface between Gramarye and its user
3. The interoperability of Gramarye with other information systems
4. The accuracy with which the models reflect known probabilistic dependencies
5. The overall usefulness of the approach to decision modeling

In Sections 5.1 through 5.7, I discuss where along these dimensions Gramarye's strengths and limitations lie. It is my hope that other researchers will find this description helpful in deciding whether to use a graph-grammar derivation system for their decision-modeling needs.

5.1 Scope and Expressivity

In designing a grammar for medical decision modeling, I have deliberately avoided restricting the problem to any particular specialty field or problem within medicine, since, as I discussed in Section 2.1.2, influence-diagram templates may suffice for highly specific problems. Such a template-pruning approach is not likely to scale up to new, unanticipated problems, since templates must encompass all future models as subgraphs, and they do not employ any abstractions, under which the user could classify new considerations. The graph-grammar approach, on the other hand, can incorporate abstractions that are at any level of generality.

The abstractions used in the grammar described in Section 3.6.1 are general to medicine. As an example, Figure 5.1 shows two simple models derived by Gramarye. Gramarye derived the first model for the tuberculosis decision problem introduced in Section 1.1. Gramarye derived the second model to help a physician to decide whether an elderly woman should undergo testing and possible treatment for osteoporosis. These two models apply to disparate fields of medicine, but their topology is the same. The terms Tb and Osteoporosis are classified under ⟨disease⟩, both Bronchoscopy and Skeletal radiography are classified under ⟨test⟩, and both Treat Tb and

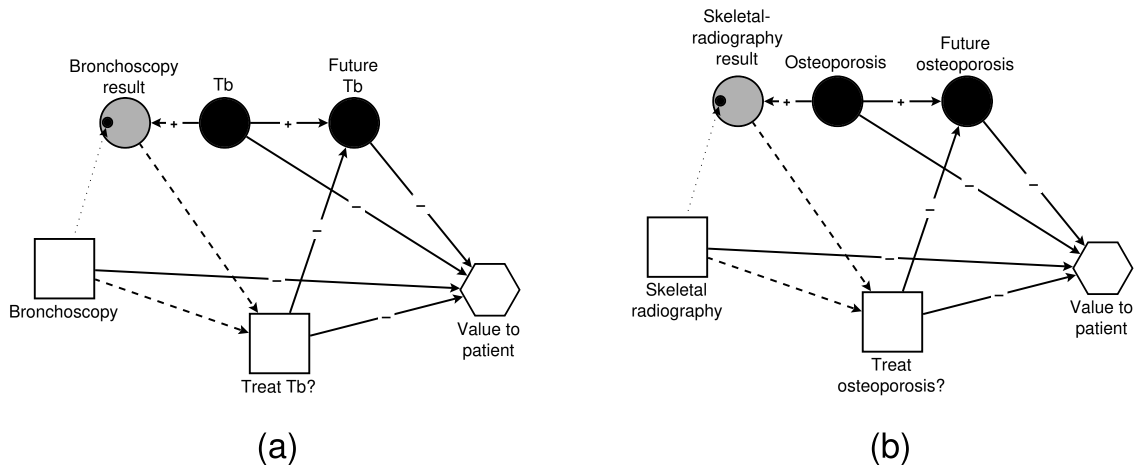

Figure 5.1: Two simple decision models from different medical domains. (a) The tuberculosis decision, modeled in Section 3.5. (b) An influence diagram that models the decision of whether a patient should have a radiographic bone scan to determine her degree of osteoporosis, and whether that patient should under treatment for the osteoporosis. Although the variables used in the two models may be defined, descretized, and assessed differently, topologically, the two models are isomorphic.

Treat osteoporosis are classified under ⟨curative treatment⟩. Due to this parallel classification of terms relating to tuberculosis and osteoporosis, Gramarye constructs topologically isomorphic models for these two decision problems, and any other decision problems that comprise three considerations of the types ⟨disease⟩, ⟨test⟩, and ⟨curative treatment⟩. The precise definition of variables—such as osteoporosis—and the way those random variables are quantized are difficult modeling issues, which I have assigned to subsequent phases of the decision-analysis cycle (Figure 2.1), and which I do not address in this dissertation.

At this point, a critic could argue that Treat osteoporosis should have been classified as a ⟨preventive treatment⟩, or as another type of treatment, rather than as a

⟨curative treatment⟩. Although a phrase more precise than Treat osteoporosis might have made the classification less debatable, there are many specific treatments that might be classified in different ways, depending on the context of the situation. For example, the surgical extraction of a tumor from a patient's abdomen may be a preventive, palliative, curative, or diagnostic procedure. Similarly, diseases—such as congestive heart failure—may be regarded as findings, depending on what else is known and on what is suspected.[1] The classifications that I have used should be familiar to physicians, and I hope that the user would be able to classify concepts appropriately for her particular decision problems. However, a more complex grammar might use abstractions that do not correspond to common clinical modes.

The designer of the vocabulary that Gramarye uses faces problems beyond the multiple classifications for the same terminal symbol. Gramarye does not model several types of considerations that a user might enter:

1. Considerations that are direct references to the context, such as "repeat therapy"

2. Considerations that are not medical

3. Medical considerations that do not fit into any of Gramarye's classifications

There is no a priori reason why the grammar that Gramarye uses could not incorporate

[1] Other researchers (Greenes et al., 1992) have suggested that the concepts "finding" and "diagnosis" represent two ends of a continuum.

patterns for nonmedical terms or contextually defined terms. Also, users may derive partial models from those terms that fit one of Gramarye's classifications, then add the other considerations manually. Later, the grammar's designer can associate these new categories of considerations with their corresponding QCID patterns, and can add these patterns and categories to the grammar. The graphical representation of graph-grammar productions makes encoding them a simple task, and Gramarye's grammar-checking routines can check that the properties discussed in Chapter 4 hold for the revised grammar. The primary disadvantage of adding additional patterns to a grammar is that users of the graph-grammar derivation system may have more trouble classifying terms under abstractions used in the grammar.

Gramarye also fails to derive decision models from close synonyms to terms that *are* classified in the node-label hierarchy. This latter problem could be addressed by an electronic thesaurus of medical terminology, or by a programmed series of questions to help the user classify terms that she has entered and that Gramarye has failed to recognize. Instead, I have built a graphical browser (Figure 5.2), which contains roughly 7500 terms derived from the QMR (Miller et al., 1986), SNOMED-III (Rothwell and Côté, 1990), and CPT (Finkel, 1990) clinical lexicons. This experimental vocabulary is far too small to cover the breadth of considerations that might occur in decision problems facing an internist. However, the user can add terms that are not present using the browser's graphical interface. Once added, these new terms are saved, along with their positions in the browser and in the node-label classification hierarchy. It is my hope that the examples and the additional abstractions—beyond those in Figure 3.15—used in the browser would help a naive user to classify new concepts; however, since I have been the sole user of Gramarye, I can only conjecture that other users would find the browser's vocabulary tree to be useful.

For the designer of a system such as Gramarye, the central problem posed by the vocabulary is that of devising abstract categories that have meaning to the user and that have distinct prototypical patterns in QCID models. This task of devising abstractions is essentially a modeling task, similar to the manual composition of QCIDs. However, by encoding these abstractions and patterns in a graph grammar, the designer of the grammar has endowed future decision makers with *reusable* patterns and the means for automated support for decision modeling. This same set of abstractions is what limits the expressivity of the derivation system, since, for considerations that are not classified uniquely, users must be able to understand how to classify considerations. Only when the vocabulary is known in advance can the grammar's designer incorporate patterns for a class of concepts where that class has no corresponding label—such as palliative treatment or preventive treatment—for the system's intended users.

Another feature of an expressive vocabulary is the ability to derive related concepts from the entered terms. The lexical variants that I found to be useful in Gramarye

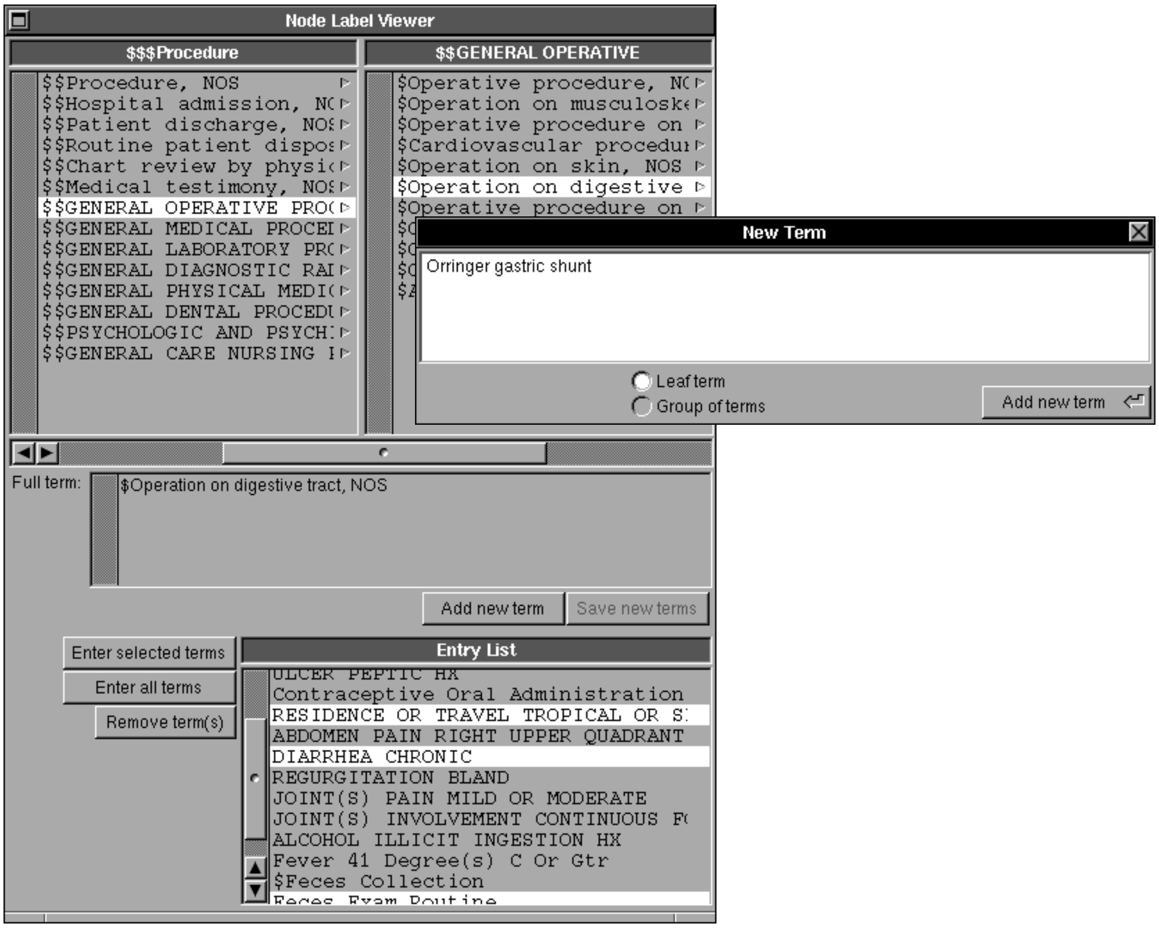

Figure 5.2: Gramarye's vocabulary browser. On the left, a graphical user interface allows the user to investigate abstract categories and terminal symbols in a classification tree that extends up to seven levels beyond the basic node-label classification shown in Figure 3.15. The user may add individual terms to her list of considerations by double-clicking on the term in the upper browser. In the window on the right, the user has begun to add an additional term—here, Orringer gastric shunt—to an existing abstract category.

were merely stem terms with either the prefix "future" or the suffix "result." For example, test results—observable chance nodes contingent on a decision to perform a test—are introduced into derivations by Gramarye, rather than by the user. This feature of the grammar's design enforces a particular style of modeling, and allows the designer to use patterns that introduce more than a single node for a single entered consideration. Such variants might be used by the designer to remind the user about typical associated considerations—such as the cost of a prescribed medicine—without requiring that those associated considerations be among considerations entered by the user. Lexical variants also can be used to introduce concepts with which the user may not be familiar. For example, suppose that the grammar's designer wishes to include a pattern corresponding to **hypeases**[2]—diseases that occur only as the result of the patient's perception that they are being tested for a potentially fatal disease (Figure 5.3). Suppose, also, that since the intended users of the system have never considered such diseases as a category, they have no term for such hypeases. The grammar's designer may introduce these hypeases as lexical variants of the potentially fatal diseases for which the user is considering testing. The lexical variant could consist of the prefix "iatrogenic fear of" followed by the name of the potentially fatal disease. For example, one hypease would be the concept iatrogenic fear of rabies. Thus, even though the user has no abstract category for hypeases, the grammar could

[2] I have invented the term "hypeases" for this hypothetical scenario. I am not proposing that the concept has any practical utility.

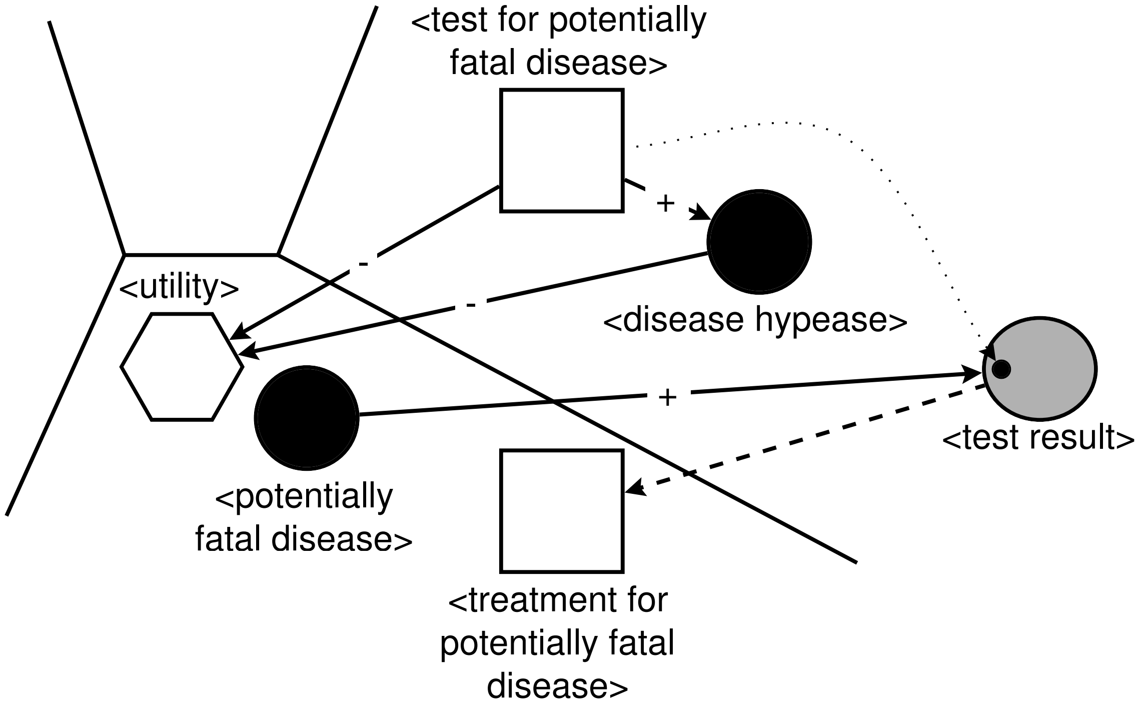

Figure 5.3: A production that incorporates hypeases into decision models. Here, **hypeases** represent diseases that do not belong to any category known to the users of the derivation system. However, the grammar's designer knows that these hypeases can result from tests for potentially fatal diseases. Gramarye can use this production to incorporate an unobservable chance node with a node label that is a lexical variant of the potentially fatal disease that matches to the chance vertex in V_{B} of this production.

introduce a hypease node into a derivation, and the the user could either accept or delete this hypease node from the final decision model.

Because I have been generating—rather than parsing—lexical variants, the simple addition of either a prefix or a suffix to some root term has sufficed. More complicated procedures for generating lexical variants are possible and may be useful, but the value of such techniques is independent of the value of graph-grammar derivation systems such as Gramarye.

In general, the scope and expressivity of a graph-grammar derivation system is limited primarily by the ability of the designer to associate prototypical patterns with specific terms, with general abstractions that have meaning to the user, or with lexical variants of either specific terms or general abstractions.

5.2 User Interface

One of the most critical determinants of the success of a decision-support system in the clinical environment is the system's ease of use. The particular **graphical user interface** (**GUI**) through which the user and Gramarye communicate relies on existing technology and platform-specific standards for user-interface design.[3] The QCID visual language is, at heart, a notation that has been used by many decision analysts for decades. Neither this GUI, nor the user-interface behavior of the graph editor,[4] nor the QCID representation is the subject of my research; my interest is in the amount of assistance that Gramarye requires from the user.

An unordered list of considerations is a simple input for the user to provide, but not all derivations can succeed with no other input from the user. In particular, there are three varieties of assistance that Gramarye requires from the user in certain derivations:

[3] NEXTSTEP, from NeXT, Inc., Redwood City, CA.
[4] Diagram!, from Lighthouse Design, Inc., San Mateo, CA.

1. Classification of new terms

2. Selection from among multiple matches to V_{L} or V_{B} of an applicable production

3. Selection from among one or more matches to V_{A} of an applicable production

The first variety of user assistance—classification of terms—currently relies on the user's familiarity with the grammar, as I discussed in Section 5.1. It is entirely possible that the context of the remaining terms in a list of considerations could help Gramarye to classify terms that are not in the current vocabulary. It is also possible that a more sophisticated vocabulary and node-label hierarchy could alleviate this demand on the user. Nonetheless, in my experience, I have not found Gramarye's need for term classification to be particularly troublesome, and the time spent assigning terms to classes has been dwarfed by the time and effort spent in constructing influence diagrams manually—that is, without the use of the prototypical patterns encoded in Gramarye. The task of classifying terms grows linearly with the number of considerations, whereas the complexity of QCID design grows exponentially with the number of considerations.[5]

The second type of user assistance—selection from among potential matches for a production—arises from the Gramarye's ignorance of probabilistic relationships

[5]More precisely, the number of possible directed acyclic graphs with n nodes is

$$\sum_{i=1}^{n}(-1)^{(i+1)}\frac{n!}{i!(n-i)!}2^{(i(n-1))}G_{n-1},$$

where G_{n-1} is the number of possible directed acyclic graphs with $n-1$ nodes (Robinson, 1976).

among specific terminal symbols. When each term that the user enters belongs to a different abstract class according to the relevant graph-grammar productions, then no assistance of this sort is required from the user. For example, consider the following decision problem:

> A 6-year-old boy, who has a no known history of seizures, experienced an event that, from his mother's description, sounds to the physician like a generalized tonic-clonic seizure. Since the child was ill, and had a high (reportedly 106°F) fever during the episode, the physician strongly doubts that the child is suffering from idiopathic epilepsy. However, to be certain, the physician considers recommending that the child undergo sleep-deprived electroencephalography (EEG). For children with idiopathic epilepsy, the physician recommends anticonvulsant treatment with phenobarbital, although both the physician and the child's mother are concerned about the possibility of central nervous system (CNS) depression that could result from the phenobarbital.

There are six considerations in this case:

1. The child's lack of known history of generalized tonic-clonic seizures
2. The reported single generalized tonic-clonic seizure
3. The chance and severity of idiopathic epilepsy

4. The decision to perform an EEG

5. The decision to administer phenobarbital

6. The possibility and severity of CNS depression from phenobarbital

From terms representing these six considerations, *and with no other input or assistance from the user,* Gramarye derived the QCID model in Figure 5.4. Since each of the considerations correspond to different abstract labels in the graph-grammar productions, the derivation system does not require any disambiguation from the user.

For other derivations, Gramarye relies more heavily on the user for assistance. For example, the following case used by Wellman (1990a) includes several considerations that the grammar described in Section 3.6.1 would classify under the same abstraction.

> A man with a known history of coronary-artery disease (CAD) and cerebrovascular disease (CVD) presents with a large abdominal aortic aneurysm (AAA). There is a surgical procedure to repair the AAA, but surgery carries known risks of operative mortality or disability. The operation is especially risky for this patient because his CAD increases the likelihood that he will have a myocardial infarction (MI) during AAA surgery, and his CVD enhances the probability of a stroke. The decision

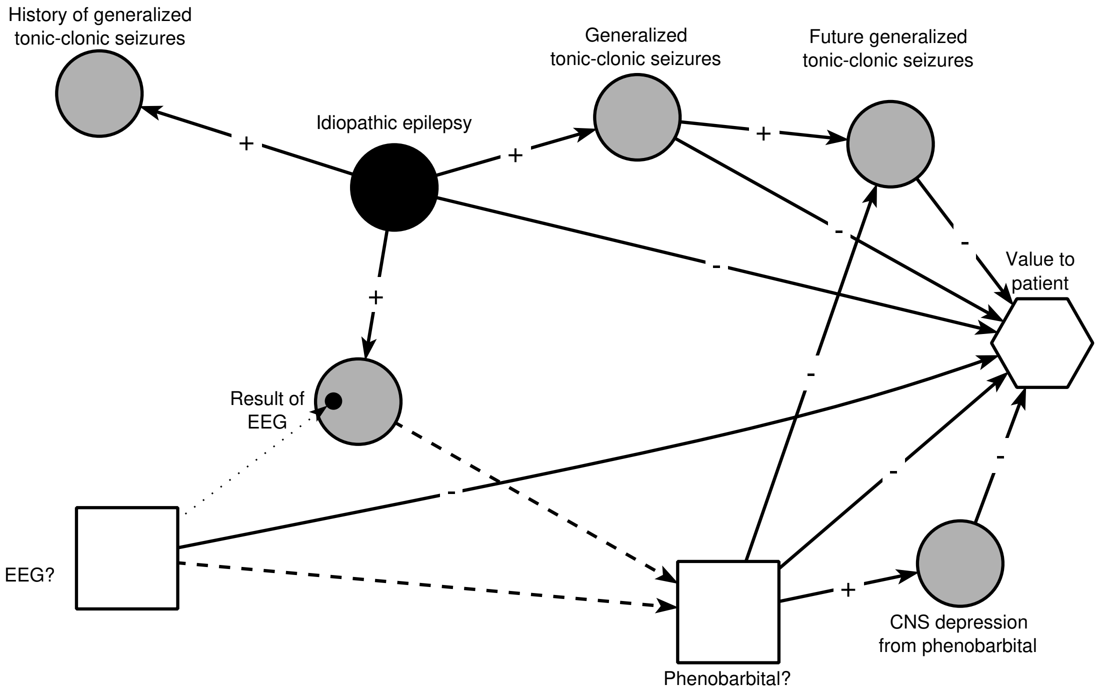

Figure 5.4: The qualitative decision model produced by Gramarye for the decision problem regarding childhood seizures. In this case, the physician is deciding whether to test for idiopathic epilepsy. Both the physician and the child's mother are concerned about the possibility that the favored treatment for idiopathic epilepsy might depress the child's central nervous system and retard his intellectual growth. (EEG = electroencephalography; CNS = central nervous system)

> also includes consideration of performing two tests: carotid arteriography, to assess the extent of the patient's CVD, and cardiac catheterization, to assess the extent of his CAD. In addition to the AAA repair, the physician has suggested two concomitant procedures: CABG surgery, to relieve cardiac ischemia caused by the CAD, and carotid endarterectomy, to reduce the risk of stroke from the CVD.

The decision considerations that I extracted from this description and used as input to Gramarye were

1. Coronary artery disease (CAD)
2. Cerebrovascular disease (CVD)
3. Abdominal aortic aneurysm (AAA)
4. CAD history
5. AAA width
6. Repair AAA
7. CABG
8. Endarterectomy
9. Cardiac catheterization

10. Carotid arteriography
11. MI
12. Stroke
13. AAA rupture
14. Anaesthesia complications
15. Patient morbidity
16. Patient mortality

Because several of these terms are classified similarly, the user must help Gramarye to select from among the possible matches to V_{A}, V_{B}, and V_{L} of rules used in the derivation. The amount of user assistance required in this derivation was 12 choices. I have recorded the actual dialogue between Gramarye and the user in Appendix C. The graphical model generated by Gramarye is shown in Figure 5.5.[6]

Of the dozens of models that I have derived using Gramarye, this decision model is the largest. It also is the model that required the greatest number of selections from the user. I expect that the vast majority of clinical decision problems could be modeled satisfactorily with fewer than a dozen nodes. The amount of assistance that

[6]Wellman's manually constructed decision model appears in Figure 2.3. Because I had seen this latter model before I designed the current grammar, I do not consider the similarities between Figures 2.3 and 5.5 to be evidence of the current grammar's validity.

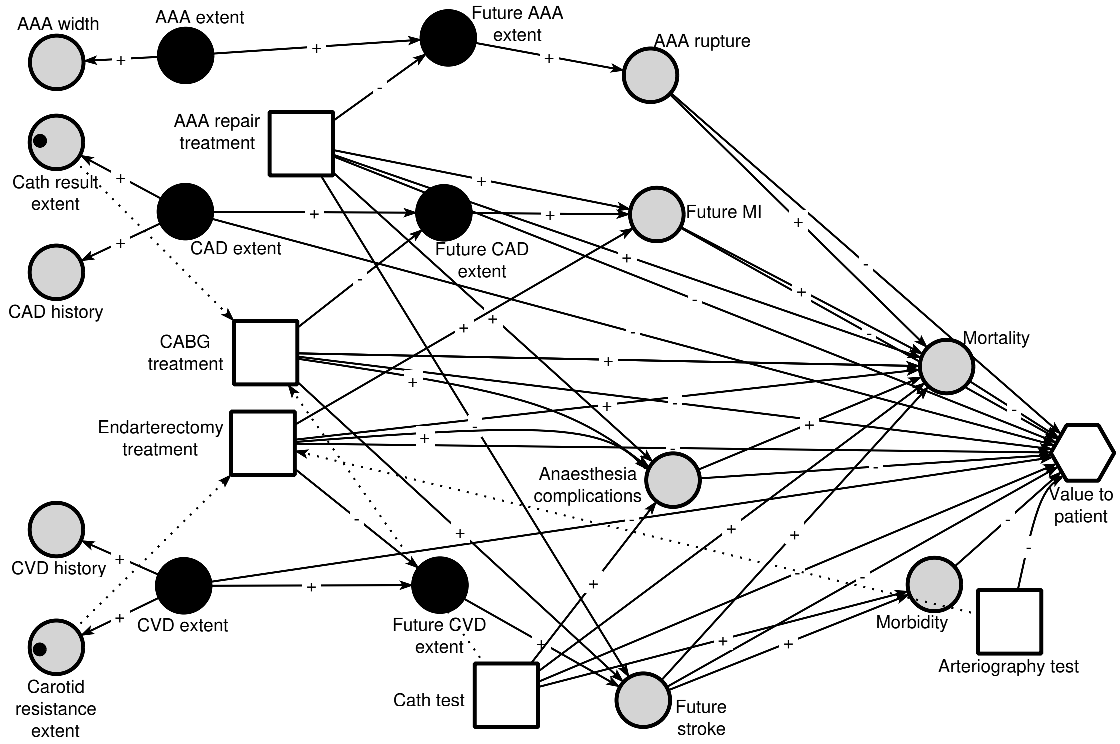

Figure 5.5: The qualitative decision model produced by Gramarye for the decision problem regarding coronary-artery bypass graft (Wellman, 1990a). The decision maker must devise a plan from two possible tests (cardiac catheterization and carotid arteriography) and three surgical treatments (abdominal aortic aneurysm repair, coronary-artery bypass graft, and carotid endarterectomy). The input concepts for this derivation were CAD, CVD, AAA, CAD history, CVD history, AAA extent, AAA repair, CABG, Endarterectomy, Cath, Arteriography, MI, Stroke, AAA rupture, Anaesthesia complications, Morbidity, and Mortality. The indeterminism due to multiple concepts filling the same role in the grammar of Section 3.6.1 forced the user to make 12 selections during the derivation of this model. The user interaction for this derivation appears in Appendix C. A similar decision model that Wellman constructed manually appears in Figure 2.3. (AAA = abdominal aortic aneurysm; CABG = coronary-artery bypass graft; Cath = cardiac catheterization; CAD = coronary-artery disease; CVD = cerebrovascular disease; MI = myocardial infarction.)

I expect such derivations would require, beyond classification of the input considerations, is fewer than five selections.

5.3 Interoperability

Since Gramarye is not designed to automate more than the initial, qualitative structuring of a decision model, the degree to which Gramarye can complement other decision-analysis tools and methods is important to the role of graph grammars in the decision-analysis cycle. In this section, I have divided the programmatic interface from Gramarye to other systems into three types of communication:

1. Input lists of considerations provided to Gramarye
2. Derivation hints provided to Gramarye to reduce the need for user assistance
3. Output models from Gramarye transmitted to tools used in subsequent phases of decision analysis

5.3.1 Input to Gramarye

The input list of considerations is an exceedingly simple data structure, which any other system might easily communicate to Gramarye. Devising that list of salient considerations is certainly *not* easy—a modeler must rely on common sense, medical training, and professional experience. Consequently, I do not envision automating the creation of this list. However, an electronic medical record (EMR) could have encoded already many—if not all—of these considerations. Users might select factors involved in a particular dilemma from an existing list of topics in the EMR. Alternatively, the EMR itself may have a representation of the patient's situation that is rich enough to propose specific clusters of concerns related to a planned test or therapeutic procedure that has been entered into the EMR. This latter possibility could help health-care workers to analyze the quality of medical decisions before those decisions are executed.

Another source of input considerations could be a previous iteration of the decision-analysis cycle (Figure 2.1). If, during the testing phase of decision analysis, the user finds that additional concerns should be included in the model, Gramarye can structure a new model from an extended list of considerations. Thus, both the initial structuring and the subsequent revisions of a decision model can be aided by a graph-grammar derivation system.

5.3.2 Derivation hints for Gramarye

Because, as shown in Section 5.2, syntactic distinctions are insufficient to distinguish between findings associated with one disease and findings associated with another, Gramarye's graph grammar cannot determine where to place a particular finding in a host graph with multiple diseases. A domain-specific knowledge base could reduce the number of such selections that the user is required to perform. Consequently, I performed an experiment in which I extended Gramarye such that it uses the medical knowledge contained in QMR-BN (Shwe et al., 1991) to decide which finding–disease links to include in a QCID model. The QMR-BN knowledge base contains over 55,000 explicit relationships among findings and diseases; where a finding and a disease are both present in QMR-BN, but no relationship appears in the knowledge base, Gramarye interprets this omission as an implicit statement of probabilistic independence.

Despite the wealth of implicit and explicit domain knowledge encoded in this knowledge base, I did not find this augmentation of the derivation machinery to reduce the amount of assistance required from the user. The reasons for this negative result are that disease–finding arcs account for only a fraction of the arcs in a typical decision model, that only a fraction of decision models that I derived included nodes for multiple findings and diseases, and that only a fraction of those findings and diseases were represented as terms in the QMR vocabulary. Gramarye could use known relationships from other knowledge bases and repositories of static models, but the variety of such specific relationships in medicine is so great that, for any new decision problem, there is little chance of Gramarye avoiding user assistance by relying on a known relationship.

If the input list of considerations is provided by an EMR, and if that EMR encodes semantic relationships between terms, those semantic relationships could provide Gramarye with the information it requires to avoid relying on the user for assistance. Campbell and Musen (1994) have proposed a principled vocabulary based on SNOMED III (Rothwell and Côté, 1990) and on conceptual graphs (Sowa, 1984) as the knowledge representation for an EMR. Unfortunately, since such EMRs are not currently available, I can only speculate about their interface to a system such as Gramarye.

Other sources of information that could be valuable to decision analysis include clinical databases, medical literature, practice guidelines, and clinical-trial protocols. Much of this information pertains to subsequent phases of the decision-analysis cycle. Therefore, I have not investigated how Gramarye might interact with these sources.

5.3.3 Output from Gramarye

Gramarye stores as editable diagrams the qualitative models that it generates. I have also implemented a compiler, which can translate these diagrams into Hugin (Andersen et al., 1989), Ergo (Beinlich and Herskovits, 1990), CABeN (Cousins et al., 1992), Demos,[7] Infer,[8] SPI (D'Ambrosio, 1989), or Ideal (Srinivas and Breese, 1990) format. Assessment, testing, and inferential reasoning over a decision model

[7]Lumina Decision Systems, Palo Alto, CA.
[8]Adam Galper. Personal communication.

can be performed by any of these belief-network and influence-diagram tools.

Many decision analysts use the decision-tree representation for their models. Although I have not generated decision trees using Gramarye, the decision-tree and influence-diagram notations are equivalent. Moreover, my experience in implementing a translator that converts decision trees into influence diagrams has convinced me that the reverse translation would be relatively simple.

Overall, the clearly defined input to and output from Gramarye enhance the latter's interoperability with other information tools.

5.4 Validity

Despite the model properties that Gramarye ensures through the grammar-checking procedures described in Chapter 4, there is no guarantee that the derived model reflects the user's best understanding of the situation. Although the user may easily edit a generated QCID model in the graph editor, the degree to which such editing is necessary determines, in part, the utility of using Gramarye for the initial modeling.

Since there is no metric of quality for arbitrary decision models, I have chosen to compare the models derived by Gramarye with models that practicing decision analysts have composed. Due to the dearth of medical influence-diagram models available to me, I have used 10 decision-tree models that a decision-analysis consulting team at the Tufts New England Medical Center[9] developed using their proprietary program, Decision Maker.[10]

So that my own modeling biases would be minimized, I implemented a compiler that translates these decision trees into equivalent QCID models. In building this translator, I found that, despite the theoretic equivalence of the influence-diagram and decision-tree notations, the actual conversion of Decision Maker models was rife with difficulties. In addition to the traditional decision nodes, chance nodes, and utility nodes (Section 3.5), Decision Maker relies on **nadir nodes, label nodes, Boolean nodes,** and **Markov nodes.** All prior and conditional probabilities are represented in Decision Maker as **bindings,** which are arbitrary assignment statements. Probabilistic dependencies are not represented explicitly, and emerge only as Decision Maker traverses specific nodes—of any of the seven types—which are associated with binding statements in a programming language. Also, multiple nodes in the decision tree may represent the same clinical event, but each of these different nodes has a unique, eight-letter identifier, and, in all but one case, the Decision Maker models and their Tufts New England Medical Center case descriptions provided no documented mapping from these identifiers to their common or separate clinical events. Often, the identifiers—when intelligible—referred to the parent node's state, rather than to the (possibly) associated random variables. Although such imprecise

[9]Brian Kan. Personal communication.

[10]Decision Maker, version 7.0, courtesy of Frank Sonnenberg and Stephen Pauker, New England Medical Center, Boston, MA.

labeling may have been understandable for the original human composers of these models, it impedes other researchers' ability to interpret these models. Nevertheless, I was able to implement a translator, and that translator was able to convert all 10 test models into influence diagrams, although I had to assist it by translating the eight-letter node identifiers into meaningful clinical phrases.

To avoid biasing the classification of terms with my understanding of the cases, I extracted all terms from all 10 models and alphabetized the list. Then, 1 month later, I classified them—under abstractions in the node-label hierarchy—as rapidly as possible. Then, for each case, I entered the appropriate list of terms, modifying the list only when necessary to achieve a successful derivation. I performed both the classification of terms and the subsequent derivation of all models in a single day. No structural features of the derived models have been altered since then. In Sections 5.4.1 through 5.4.10, I discuss how Gramarye behaved for each of the 10 decision problems.

5.4.1 Case 1

The first of the decision problems that I used for testing involves a 39-year-old woman with a carotid-body tumor (chordectoma). The tumor may metastasize, and it may enlarge and cause damage to adjacent cranial nerves. She and her physician must decide whether radiation therapy, surgical excision, or neither is the most propitious

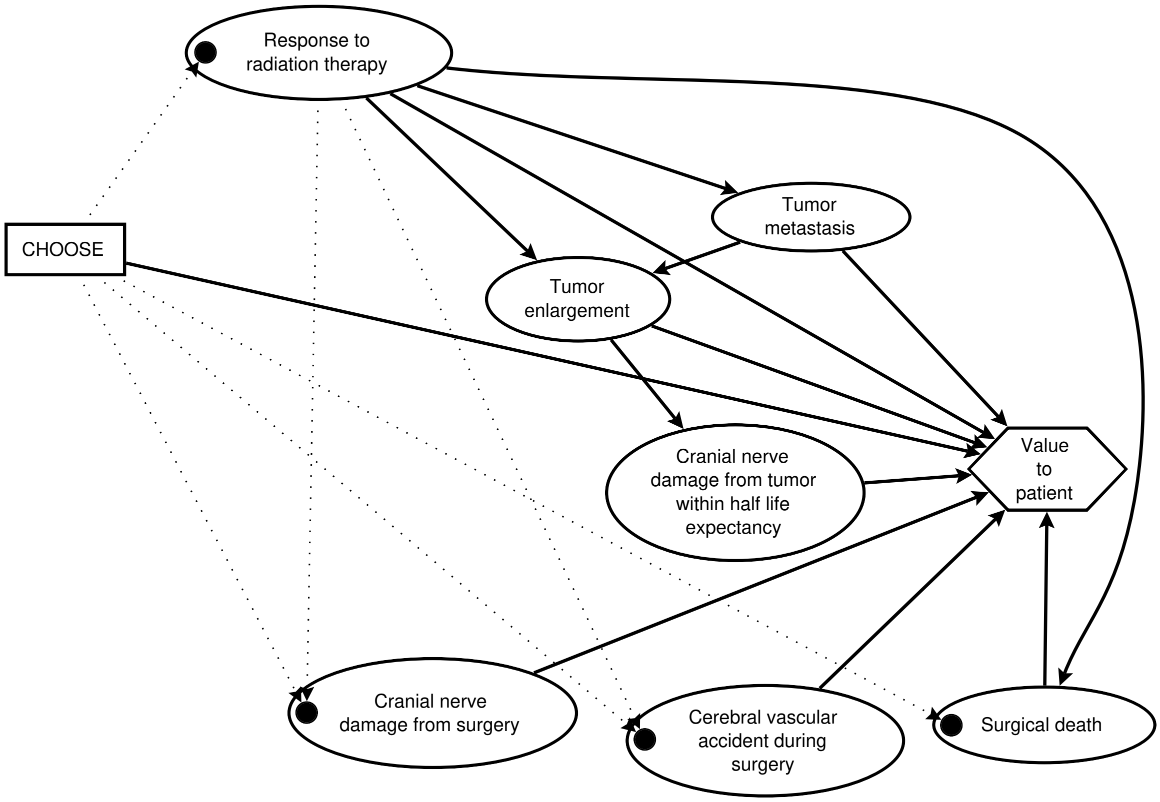

Figure 5.6: The manually composed model for case 1.

plan for her situation. Surgical treatment carries additional risks for this patient due to the fact that she has a cyanotic heart malformation.

The manually composed model for case 1 is shown in Figure 5.6. From this model, I extracted the nine considerations, listed with their respective classifications in Table 5.1. To achieve a complete derivation, I changed three of the classification choices that I made for terms while I was ignorant of their context in this particular case: I reclassified Surgical death as an instance of ⟨treatment complication⟩ rather than of ⟨mortality⟩, and I reclassified Tumor metastasis and Tumor enlargement as

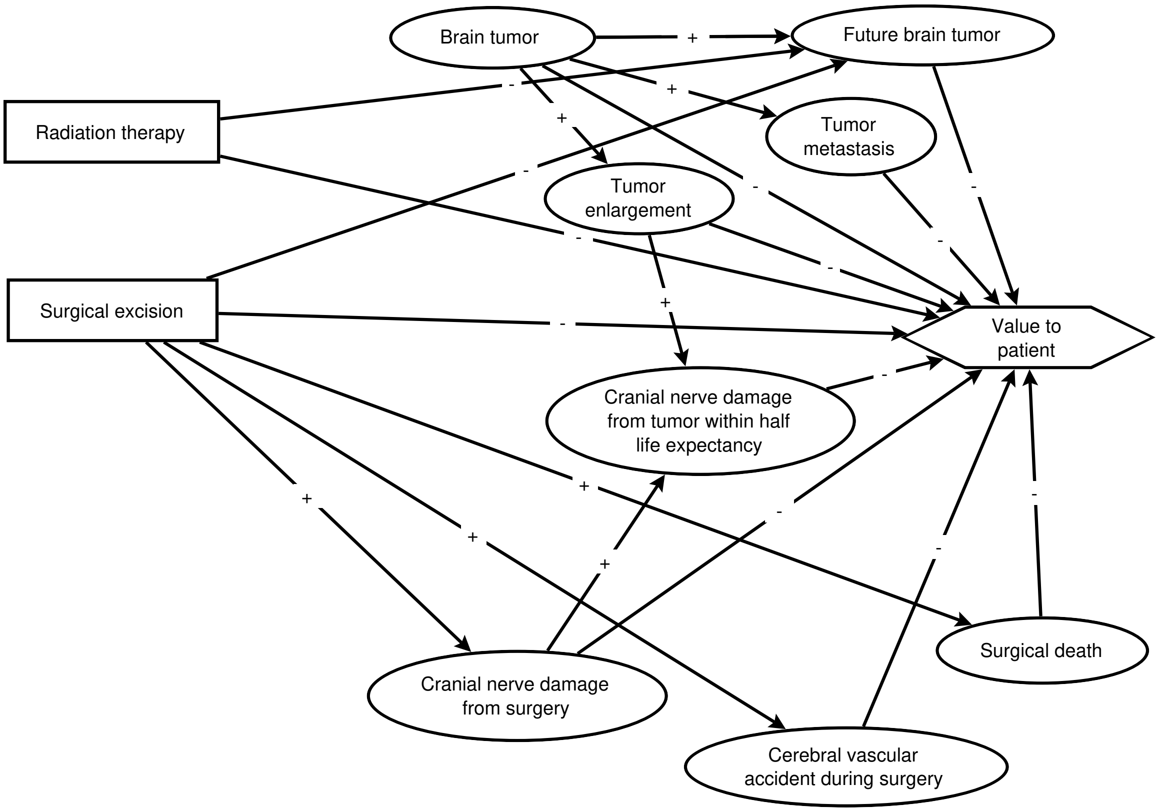

Figure 5.7: The model constructed automatically by Gramarye for case 1.

instances of ⟨present malady complication⟩, rather than of ⟨morbidity⟩. Also, although the consideration is not represented explicitly in the manually composed model, I added Brain tumor (i.e., the patient's carotid-body tumor) to the list that I entered into Gramarye.

Consideration	*Classification*
Cranial nerve damage from tumor within half life expectancy	⟨present malady complication⟩
Surgical death	⟨treatment complication⟩
Brain tumor	⟨present disease⟩
Tumor metastasis	⟨present malady complication⟩
Tumor enlargement	⟨present malady complication⟩
Cranial nerve damage from surgery	⟨treatment complication⟩
Cerebral vascular accident during surgery	⟨treatment complication⟩
Surgical excision	⟨curative treatment⟩
Radiation therapy	⟨curative treatment⟩

Table 5.1: Considerations entered into Gramarye for case 1. The column on the right lists the node-label–hierarchy abstractions under which I classified the considerations in the left column. The node-label hierarchy is described in Section 3.6.2.

Given this list of considerations, and given five rule-application selections provided by the user at the time of derivation, Gramarye constructed the model in Figure 5.7. The structural differences between this model and the manually composed model in Figure 5.6 can be divided into four general categories:

1. Stylistic differences in modeling disease progression

2. The difference between a contingency arc and the more general probabilistic arc

3. Gramarye's explicit representation of each decision alternative as a node

4. Individual probabilistic arcs that appear only in Gramarye's model or in the original model

In Sections 5.4.1.1 through 5.4.1.4, I discuss these differences between the two models.

5.4.1.1 Stylistic Differences for Disease Models

In the first category, the manually devised node Response to radiation therapy is comparable to Gramarye's pair of nodes Brain tumor and Future brain tumor. In both models, the parent node represents the decision to undergo radiation treatment, and in both models, the child nodes include Tumor enlargement, Tumor metastasis, and Value to patient. Two qualitative decision models that differed in only this respect could be assessed to behave identically.

5.4.1.2 Contingencies as Constrained Probabilities

The manual model contains four nodes—Response to radiation therapy, Cranial nerve damage from surgery, Cerebral vascular accident during surgery, and Surgical death—that are contingent on the decision node. The corresponding nodes in Gramarye's model are not contingent, but are probabilistically dependent on the decision nodes. Since the conditional probability table implied by Gramarye's probabilistic arcs may have the value 0.0 for scenarios that correspond to the decision to forego radiation therapy and surgery, this part of Gramarye's model may be assessed to behave identically to the manual model. However, Gramarye has not modeled the general constraint that treatment complications are contingent on the treatment being performed.

A simple revision to the grammar (shown in Figure 5.8) could rectify this situation, but additional consequences of such revisions should be considered. Can treatments result in complications that are identical to complications of a disease? Can the same complications occur in absence of any treatment? Should a single model contain a node that performs roles of more than one node-label–tree abstraction? These questions address issues of grammar design, and I have no definitive answer to any of them. Nonetheless, I am convinced that the revision in Figure 5.8 would be beneficial.

Another question that the model in Figure 5.6 evokes is how might contingencies be combined to reflect typical Boolean relationships? Currently, the contingencies are binary functions of binary variables, and the positive and negative labels of contingency arcs presume that the user understands positive and negative outcomes or alternatives for the parent nodes. The contingent node becomes relevant to the rest of the model if and only if *all* contingencies hold. However, we might prefer the contingent node to become relevant whenever *one or more* contingencies hold, as in the case of treatment complications that are contingent on one of a set of treatments being performed. More complex Boolean functions that reference nonbinary parents might have general applicability. Until these deterministic patterns of contingency are represented graphically, rather than being hidden in contingency dots, such constraints

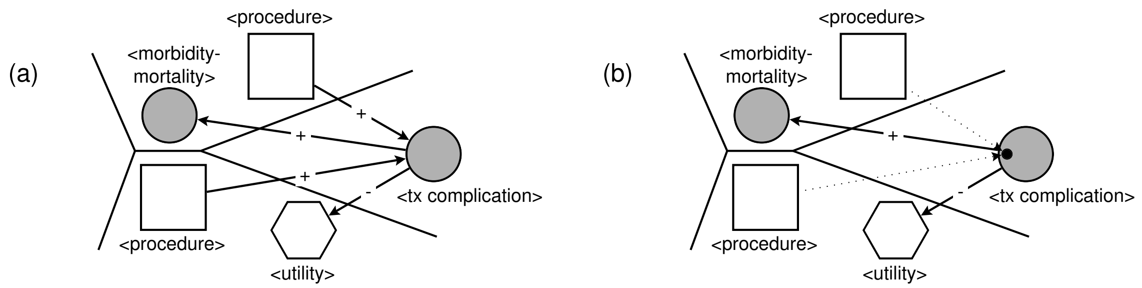

Figure 5.8: A possible revision to Gramarye's graph grammar. (a) On the left is the production rule from Gramarye (as introduced in Figure 3.13b) that adds treatment complications to a model. (b) On the right is a similar production for adding treatment complications. The latter production constrains treatment complications to be contingent on some treatment being performed.

can be introduced into a graph grammar only as attributes appended to individual productions, as in Göttler's programmed attributed graph grammars (1989).

5.4.1.3 Explicit Decisions

A requirement of the Decision Maker decision-tree shell is that all trees begin with a single decision node that is labeled CHOOSE. This feature of all Decision Maker models results in a corresponding CHOOSE node in each of the 10 manual models. In contrast, Gramarye represents explicitly as an individual node each procedure, test, or treatment that is being considered. In this case, the individual decision nodes must be sequenced manually (i.e., ordered completely using informational arcs) before the user can compute the expected utilities of the different alternatives (Chapter 4, Section 4.1.4).

If additional information becomes available to the decision maker after the first decision, but before the second decision, then it may be important for the decision maker to weigh that information so that she can arrive at a normative choice for the second decision. Models constructed with Decision Maker typically define the alternatives of the CHOOSE node to be a subset of all combinations of the various individual decisions. Although this latter approach has the computational advantage that certain unorthodox sequences of choices can be omitted from the model, and thereby ignored, the approach has the modeling disadvantages that individual decisions are not represented explicitly, and optimal sequences of decisions—particularly those which depend on latently available information—can be omitted by accident. Consequently, I regard Gramarye's explicit representation of decisions as a feature, rather than as a bug.

5.4.1.4 Arc Differences

Gramarye's model adds one arc that is not present in the manual model, and omits two that are in the manual model. The extra arc that Gramarye adds links the node labeled Cranial nerve damage from tumor within half life expectancy to the node labeled Cranial nerve damage from surgery. Presumably, if a cranial nerve is severely damaged by surgery, then the damage caused by tumor enlargement will have a different probabilistic distribution. A clearer model might merge these two nodes as a single node. A user who was familiar with Gramarye's grammar might decide to enter the single consideration Cranial nerve damage. Due to the original decision-tree representation, I cannot be certain whether the human modeler for this problem intended for these two nerve-damage events to be conditionally independent given ignorance of the utility node's value. However, the inclusion of the arc is noncommittal; the human modeler's exclusion of an arc represents the stronger statement of conditional independence. Consequently, I do not fault Gramarye's grammar for this difference between the two models.

The two arcs that Gramarye does not include link Response to radiation therapy to Surgical death and Tumor metastasis to Tumor enlargement. The latter omission is due to the lack of a ⟨malady complication⟩ in the V_{A} region of the production rule that adds ⟨malady complication⟩ nodes. This deficiency is deliberate; the current grammar does not model explicitly any aspect of human physiology. Consequently, chance nodes of the same type—such as ⟨malady complication⟩ or ⟨finding⟩—may or may not merit corresponding statements of conditional independence. The current grammar errs on the side of omission.

In the manual model, the probability of surgical death is doubled if the surgery follows unsuccessful radiation therapy. Since the current grammar does not dictate the proper sequence for therapies, Gramarye does not model any observations that can occur after a first therapy and before a second therapy. Consequently, Gramarye's model cannot have any such arc from a response outcome to the mortality of a second therapy. Even if Gramarye's user knew in advance not to consider surgical treatment followed by radiation therapy, she still could not have selected the node Future brain tumor as an intermediate observation—in this case, a ⟨future disease⟩ for Radiation therapy, and a ⟨present disease⟩ for Surgical excision; in the grammar used for this evaluation, the variant node labels for ⟨future disease⟩ require a ⟨present disease⟩ stem, and the two sets of terminal symbols that are classified under ⟨future disease⟩ and ⟨present disease⟩ are disjoint. Gramarye's grammar does not allow for a succession of disease states other than one state prior to any intervention, and a second state after all interventions.

Gramarye's omission of any dependence of Surgical death on Radiation therapy reflects the following selection on the part of the user during this derivation:

> ... while trying to add **Surgical death** ...
>
> The following labels could map to PROCEDURE:
>
> (Radiation therapy)
>
> Should Radiation therapy fill the PROCEDURE role? **n**

Given the decision-tree modeler's assumption that both radiation therapy and surgery would be performed only if radiation therapy was performed first and failed, Gramarye's user might have answered in the affirmative at this point in the derivation, and the assessed conditional-probability table for Surgical death could reflect a doubled risk of death when both treatments are performed. Thus, it is possible for

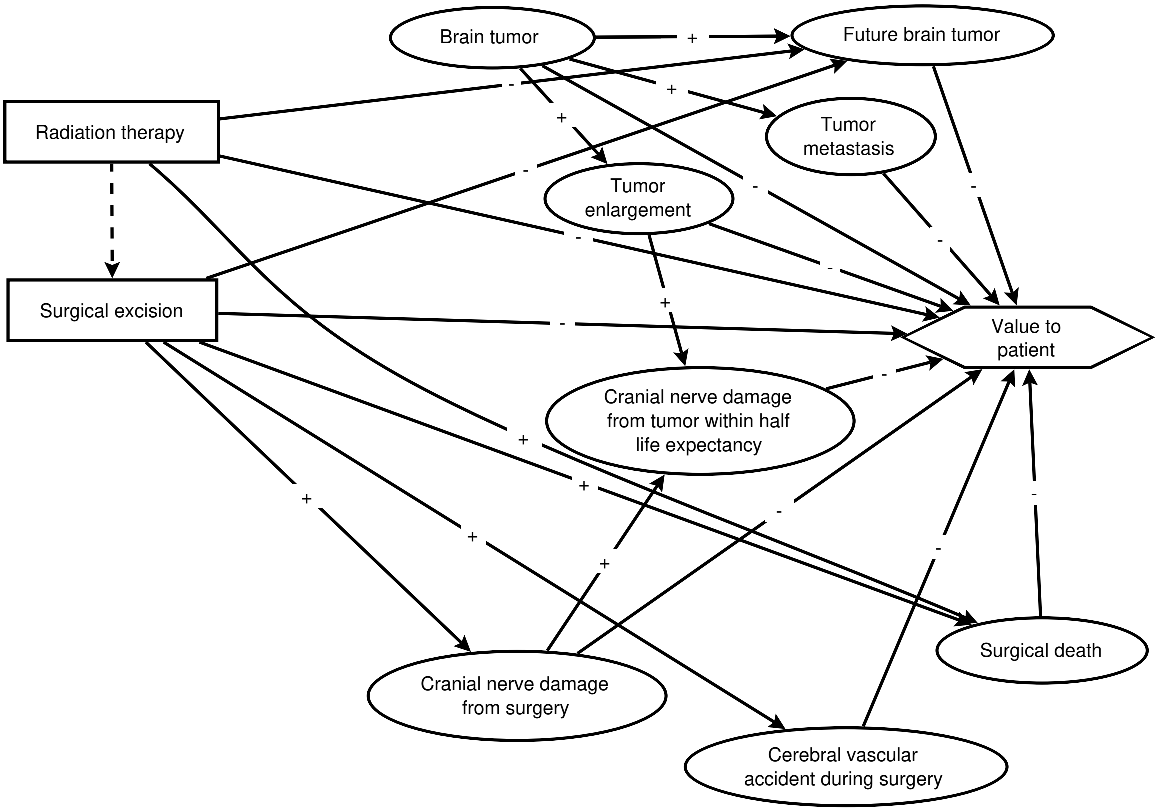

Figure 5.9: An adjusted model for case 1. This model assumes that radiation therapy should not follow surgical excision, and that surgery would not follow successful radiation therapy. From these assumptions, Gramarye's grammar could model the proposition that the risk of surgical death is doubled following unsuccessful radiation therapy.

Gramarye to derive a model for this problem where Surgical death is *implicitly* dependent on the patient's response to radiation therapy. However, the resulting graphical model (Figure 5.9) would not convey that it is the intermediate health status of the patient—rather than the mere combination of treatments—that affects the risk of surgical death.

In general, Gramarye would need to make a commitment to sequencing ⟨treatment⟩

decision nodes early in a derivation before it could introduce nodes that represent observations and states that occur between successive decisions. The current grammar has no information that it could use to sequence procedures—other than the grammar's requirement that specimen collections precede tests, which precede treatments. The grammar lacks also the abstractions for intermediate states and observations.

5.4.2 Case 2

The second of the decision problems that I used for testing involves a 31-year-old immunodeficient man who may have subacute bacterial endocarditis. The decision maker must decide whether to recommend additional testing (transesophageal echocardiography), and how long to continue the patient's current regimen of antibiotic therapy.

From the manually composed model in Figure 5.10, I entered the list of considerations shown in Table 5.2 into Gramarye. This derivation required no reclassification of terms, no additional concepts, and no assistance from the user during the derivation. The derivation resulted in the model shown in Figure 5.11.

The manual model's notion of cure, represented by the nodes Cure (given short treatment) and Cure (given recurrence), is represented also by Gramarye's equivalent—although stylistically different—Future subacute bacterial endocarditis. The decision

Consideration	*Classification*
Transesophageal echo	⟨test⟩
Subacute bacterial endocarditis	⟨present disease⟩
Mortality	⟨mortality⟩
Treat SBE for next 2 weeks	⟨curative treatment⟩

Table 5.2: Considerations entered into Gramarye for case 2. (echo = echocardiography; SBE = subacute bacterial endocarditis.)

alternatives in Gramarye's model are explicit and, in this case, are ordered by Gramarye. This explicit modeling of decision alternatives allows for more detailed modeling of test results, and allows these test results to be used by the decision maker in weighing subsequent decisions. The manual model maintains that, once a decision is made to perform the diagnostic test, any positive result from that test necessarily entails an additional 2 weeks of treatment. This assumption may be appropriate in this situation, but the style of modeling may also lead the decision maker to disregard prior probabilities and test characteristics (i.e., the sensitivity and specificity of a test for a particular situation).

Gramarye's model lacks three probabilistic dependency arcs that are present in the manual model. Two of these arcs provide reasonable disease and treatment conditioning for the probabilistic distribution of the Mortality variable. This omission in Gramarye's model appears to be the result of the grammar's method for adding nodes of types ⟨malady⟩, ⟨procedure⟩ and ⟨morbidity-mortality⟩. Since none of the rules that add nodes of these classes include optional arcs from ⟨malady⟩ and ⟨procedure⟩

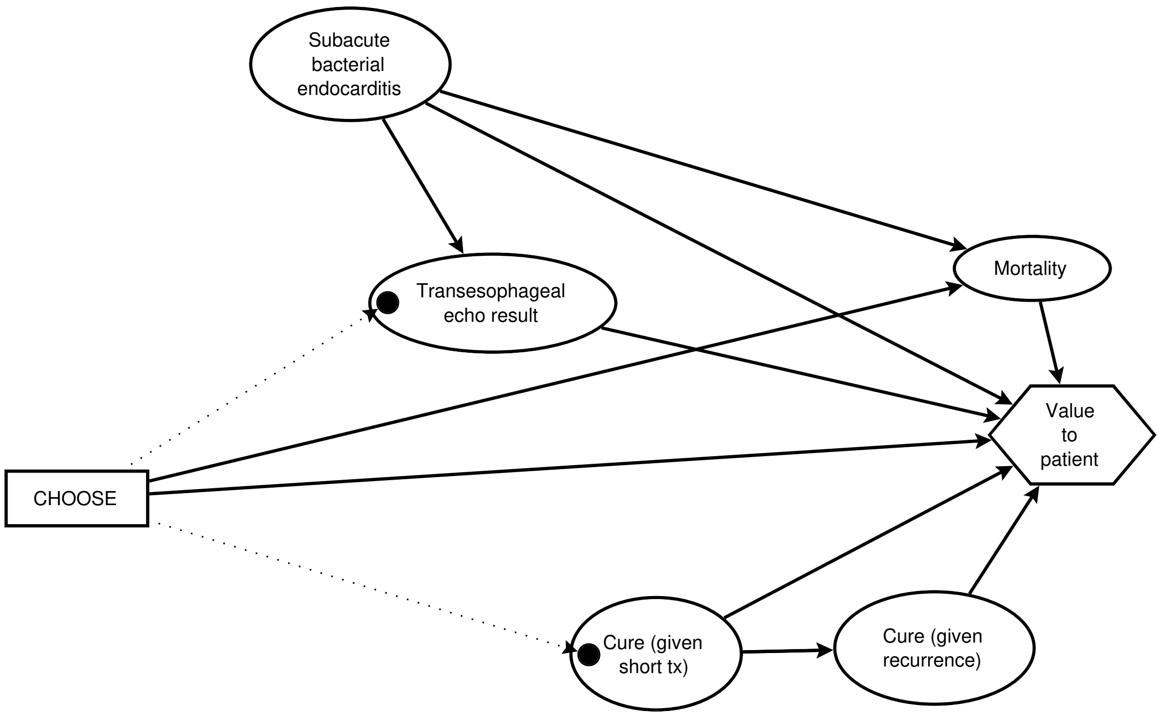

Figure 5.10: The manually composed model for case 2. (echo = echocardiography; tx = treatment.)

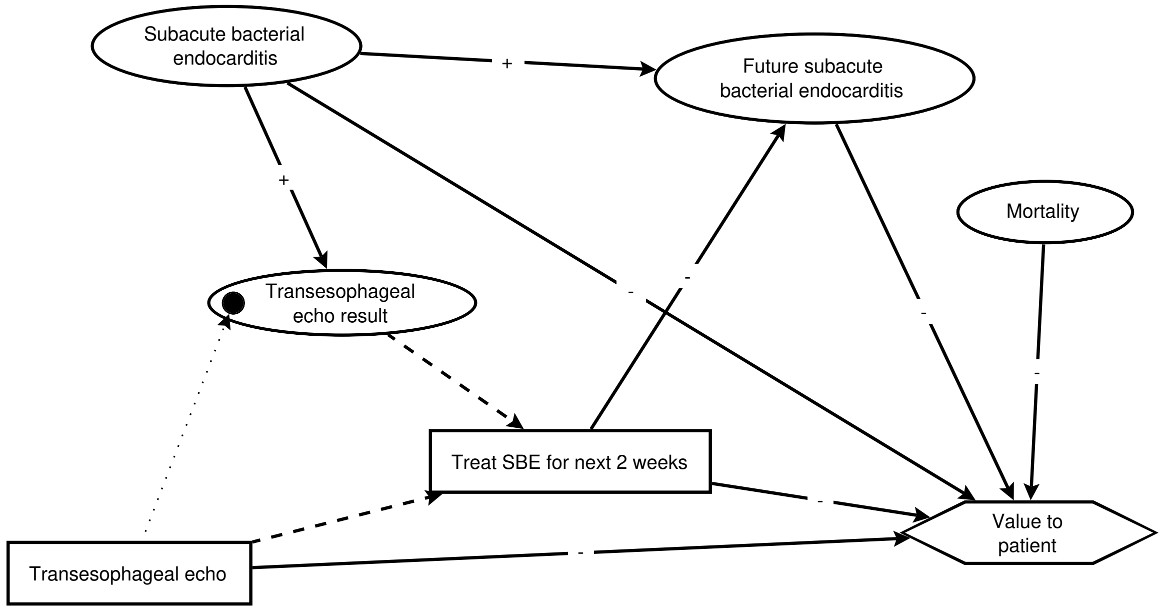

Figure 5.11: The model constructed automatically by Gramarye for case 2. (echo = echocardiography; SBE = subacute bacterial endocarditis.)

nodes to ⟨morbidity-mortality⟩ nodes, Gramarye does not attempt to include any such arcs in this derivation. Had the user included one or more more immediate causes of mortality—causes that were themselves complications of the disease or treatment—then Gramarye would have modeled these two arcs as arcs from the ⟨malady⟩ and ⟨procedure⟩ nodes to ⟨complication⟩ nodes, and arcs from those ⟨complication⟩ nodes to Mortality. Alternatively, the user might have classified Mortality as an instance of ⟨complication⟩, rather than as one of ⟨mortality⟩. Although this latter remedy would duplicate the probabilistic dependencies that are in the manual model, this obtuse classification of the consideration Mortality suggests that the abstraction ⟨mortality⟩ is not well incorporated into the current grammar's design.

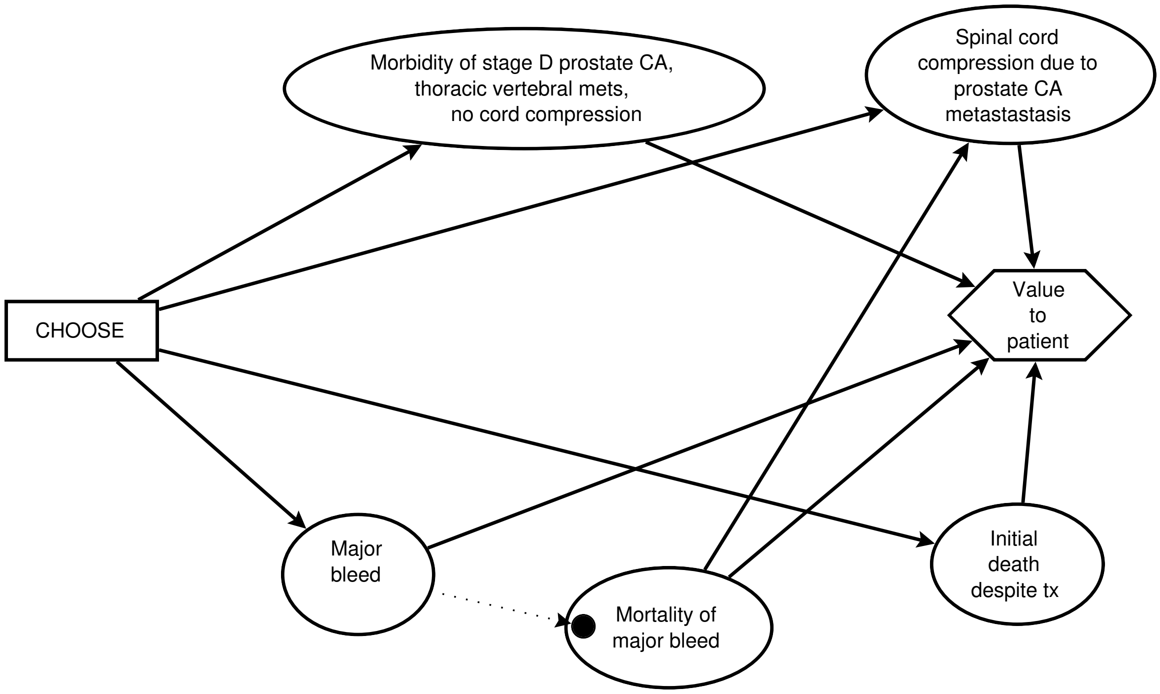

Figure 5.12: The manually composed model for case 3. (CA = cancer; mets = metastases; tx = treatment.)

The remaining difference between the two models is an arc from Transesophageal echo result to Value to patient that appears in the manual model. It is unclear to me whether the patient's well-being depended on a test result—rather than the disease state or the performance of the test—or the modeler intended the Transesophageal echo result node to represent a best estimate of the future state of the disease. In either event, this particular lack of an arc in Gramarye's model corresponds to a reasonable assumption of conditional independence.

5.4.3 Case 3

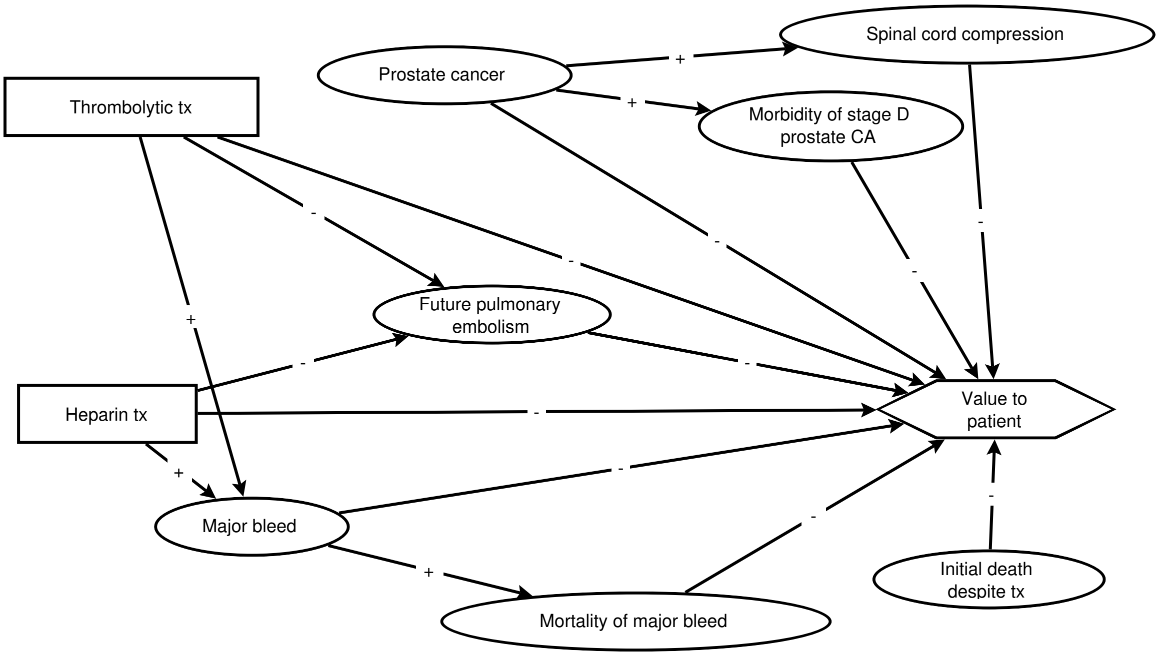

Figure 5.13: The model constructed automatically by Gramarye for case 3. (CA = cancer; tx = treatment.)

The third decision problem involves a 63-year-old Jehovah's Witness with metastatic prostate cancer and probable sepsis who has already been acutely heparinized for a suspected massive pulmonary embolus. The physician must decide whether to recommend heparin or thrombolytic therapy for this individual.

The manually composed model for case 3 appears in Figure 5.12; Gramarye's model—constructed from considerations in Table 5.3—appears in Figure 5.13. To obtain a successful derivation, I reclassified Morbidity of stage D prostate CA and Major bleed as ⟨present malady complication⟩ and ⟨treatment complication⟩, respectively, instead of their original classifications as ⟨morbidity⟩ (a subclass of ⟨morbidity-mortality⟩)

and ⟨crisis event⟩. I also introduced the additional concepts Prostate cancer and Pulmonary embolism, which are not explicitly represented in the manual model. The derivation required four selections from the user.

Consideration	*Classification*
Morbidity of stage D prostate CA	⟨present malady complication⟩
Prostate cancer	⟨present disease⟩
Major bleed	treatment ⟨complication⟩
Mortality of major bleed	⟨mortality⟩
Initial death despite treatment	⟨mortality⟩
Spinal cord compression	⟨present malady complication⟩
Pulmonary embolism	⟨crisis event⟩
Thrombolytic treatment	⟨risk-reducing treatment⟩
Heparin treatment	⟨risk-reducing treatment⟩

Table 5.3: Considerations entered into Gramarye for case 3. (CA = cancer.)

Gramarye modeled Initial death despite treatment as being conditionally independent (given ignorance of the value of the utility function, Value to patient) of the treatment selected. This statement of independence is in accordance with my understanding of the problem; the patient was at risk of dying from sepsis, and the treatment under deliberation was directed at future emboli, which might occur after the patient has recovered from the current crisis. Consequently, I do not regard Gramarye's model of dependencies here to be in error.

Gramarye's model does not include arcs from the decision nodes to the nodes labeled Morbidity of stage D prostate CA and Spinal cord compression. Here also, I do not regard this difference to be an error.

5.4.4 Case 4

The fourth of the decision problems that I used for testing involves a 34-year-old woman with acute myelocytic leukemia who presents evidence of having either appendicitis or neutropenic enterocolitis (typhlitis). The physician must decide whether to recommend surgical or medical management of the patient's current abdominal inflammation.

The manually composed model for case 4 is shown in Figure 5.14. Although the decision tree modeled for this problem was fairly complex, containing many more nodes than the three shown in Figure 5.14, 12 of the decision-tree nodes modeled had no connection to the root CHOOSE node, and, therefore, were computationally irrelevant to the decision. From the treatments and possible diseases under consideration in this problem, I used the list in Table 5.4 as my input into Gramarye.

Consideration	*Classification*
Appendicitis	⟨present disease⟩
Typhlitis	⟨present disease⟩
Appendectomy	⟨ablative treatment⟩
Hemicolectomy	⟨ablative treatment⟩

Table 5.4: Considerations entered into Gramarye for case 4.

Given this list of considerations used in the manual model, and given two clarifying selections from the user, Gramarye constructed the model in Figure 5.15. As discussed in Section 5.4.1, Gramarye's explicit representation of decision alternatives and of future disease states is a stylistic preference that does not affect the resulting behavior

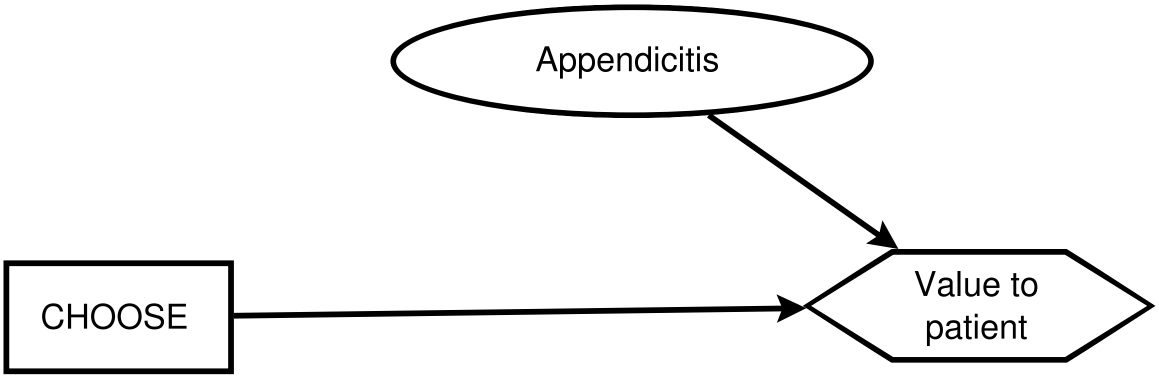

Figure 5.14: The manually composed model for case 4.

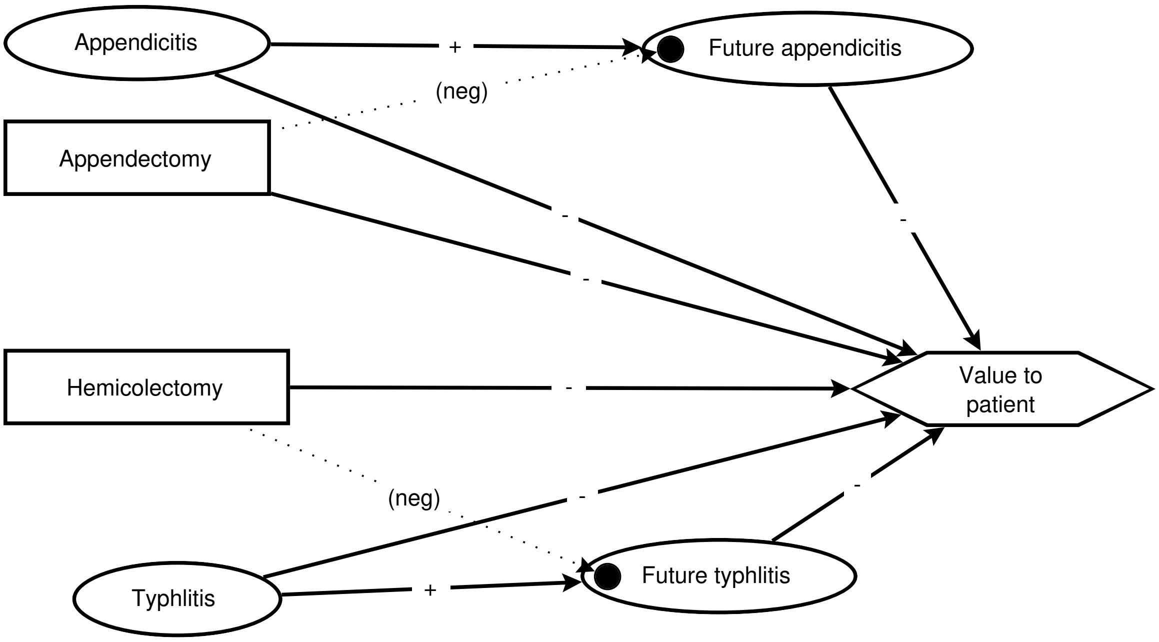

Figure 5.15: The model constructed automatically by Gramarye for case 4.

or recommendations of the assessed model. Consequently, I regard Gramarye's model to be equivalent to the manual model, with two exceptions: The manual model assumes (1) that the two diseases are mutually exclusive and exhaustive, and (2) that the decision alternatives are mutually exclusive and exhaustive. If a decision maker knows such a constraint before she begins to model the decision, then she might argue that the structuring of the model should follow this constraint on the assignment of variable outcomes. A fundamental assumption behind Gramarye is that all assessment follows the structuring of probabilistic dependence, and this sequence of steps may not be optimal for all aspects of a model.

5.4.5 Case 5

The fifth decision problem involves a 66-year-old man with colonic cancer and significant aortic stenosis (with ejection fraction of 0.65). The physician must decide whether to propose aortic-valve replacement or aortic valvuloplasty for the patient's stenosis, and whether to perform this procedure before or after a curative right hemicolectomy for the man's colon cancer. The physician is particularly concerned that, if colonic resection follows aortic-valve replacement, then the resection may result in subacute bacterial endocarditis. However, valvuloplasty could lead to restenosis, and need for subsequent replacement of the aortic valve.

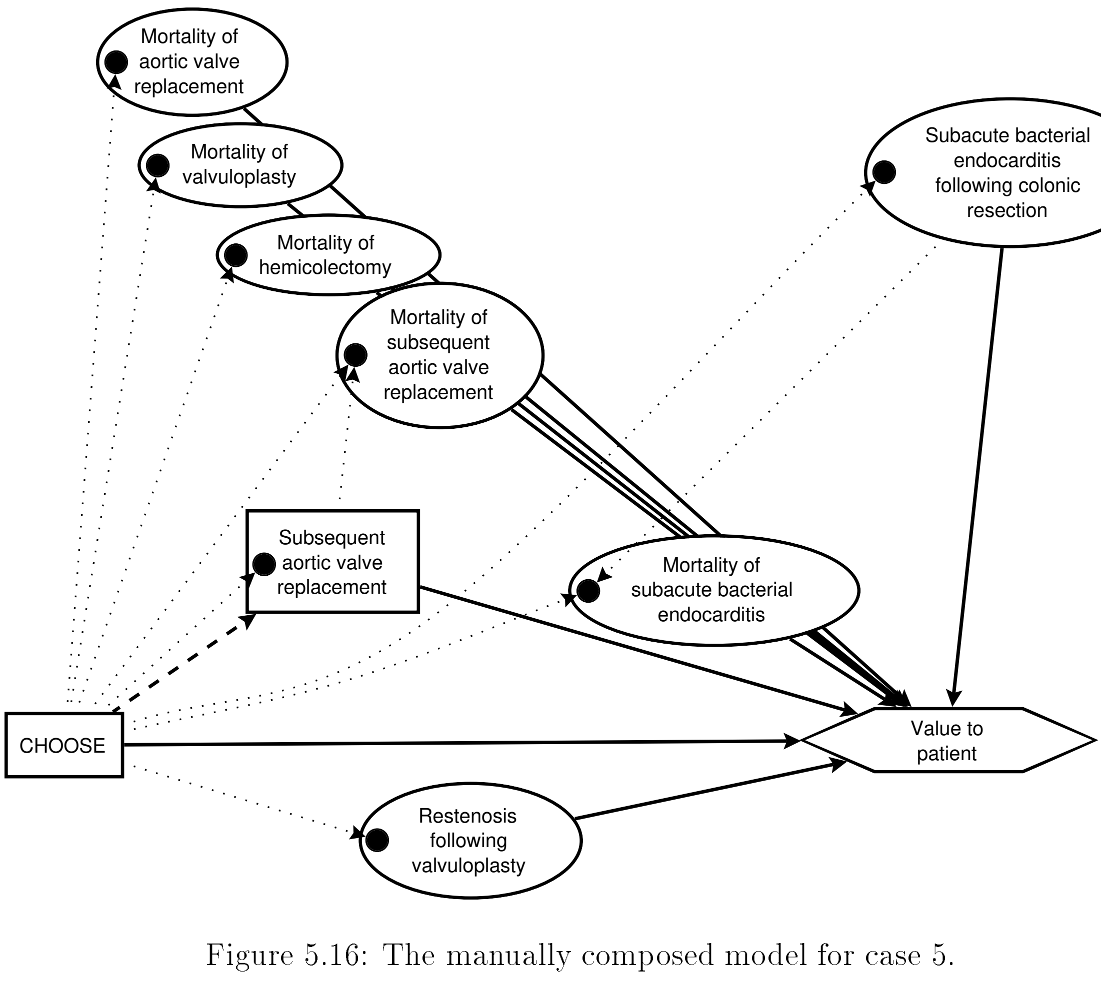

Figure 5.16: The manually composed model for case 5.

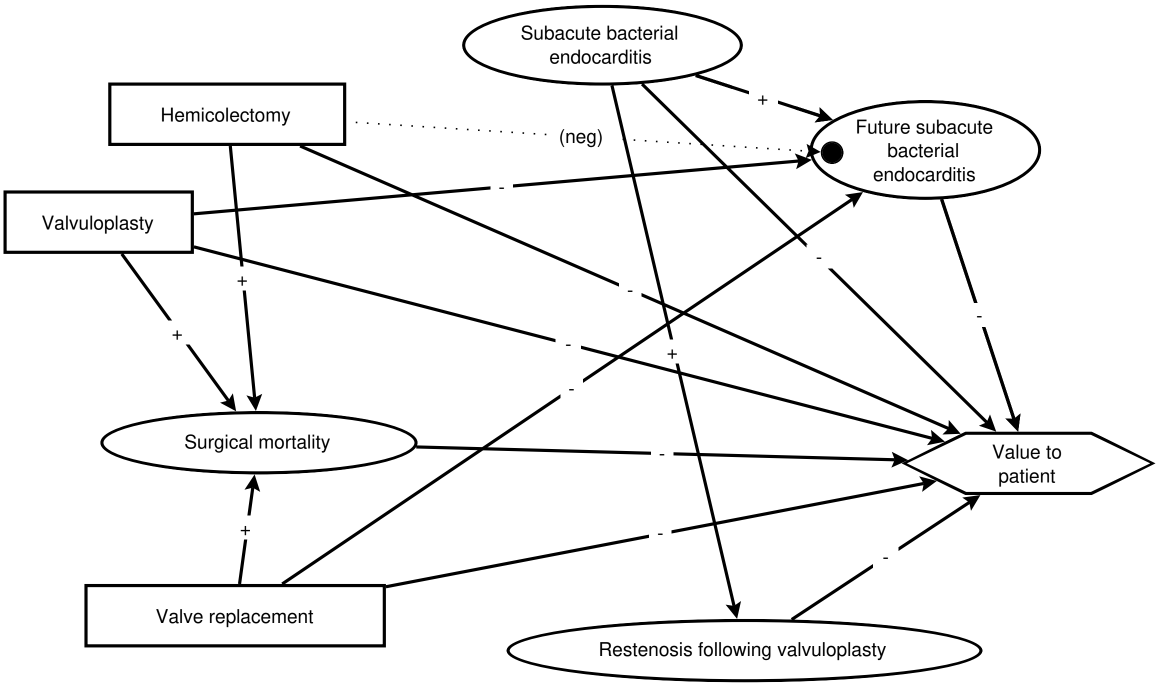

Figure 5.17: The model constructed automatically by Gramarye for case 5.

The manually composed model for case 5 is shown in Figure 5.16. The informational arc from the node CHOOSE to the node Subsequent aortic valve replacement is redundant, given the contingency arc between the same two nodes. This double arc is the result of my decision-tree–to–QCID translator, and would not occur in the traditional influence-diagram notation. The considerations that I distilled from that model are listed in Table 5.5. All considerations are listed with their original (context-ignorant) classifications, and I introduced no terms other than those from nodes in the manual model.

Given the list of considerations used in the manual model, and given one clarifying selection from the user, Gramarye constructed the model in Figure 5.17. As in all

Consideration	*Classification*
Surgical mortality	⟨treatment complication⟩
Subacute bacterial endocarditis	⟨present disease⟩
Restenosis following valvuloplasty	⟨present malady complication⟩
Hemicolectomy	⟨ablative treatment⟩
Valvuloplasty	⟨curative treatment⟩
Valve replacement	⟨curative treatment⟩

Table 5.5: Considerations entered into Gramarye for case 5.

of Gramarye's models, the decision alternatives are represented by separate and explicit decision nodes. Both current and future probability distributions for subacute bacterial endocarditis are represented in Gramarye's model.

Gramarye combines all surgical mortalities, which the manual model does not do, but could, depending on the translation from the original decision tree to the influence diagram in Figure 5.16. Mortality from subacute bacterial endocarditis may be represented implicitly in the negative arc from Future subacute bacterial endocarditis to Value to patient. However, a model that combined surgical mortalities with mortality from endocarditis might be clearer visually. Gramarye might have derived this latter structure if Mortality—rather than Surgical mortality—were entered and was classified both as a ⟨treatment complication⟩ and as a ⟨malady complication⟩.

Neither Subacute bacterial endocarditis nor Restenosis following valvuloplasty is contingent (or dependent) on operative procedures, as each is in the manual model. Had the two nodes been classified under ⟨treatment complication⟩, then Gramarye would have arrived at the structure modeled manually. Although this shortcoming may be simple for the user to amend *after* Gramarye has finished its derivation, the underlying problem is more serious: The simple list of terms entered into Gramarye does not convey necessary contextual information.

If the grammar were modified to classify terms such as Restenosis under more explicit roles—for example, Restenosis as a failure of treatment—then Gramarye might avoid such modeling mistakes. Alternatively, if the grammar were designed to group certain treatments with their criteria for success, then Gramarye could arrive at the correct dependency structure. Neither of these solutions is practical for human users of Gramarye. The former solution would require that the grammar's designer anticipate the possible contexts for future treatments and physiologic events. The latter solution would require that the grammar include within it a detailed understanding of medical relationships.

5.4.6 Case 6

The sixth of the decision problems that I used for testing involves a 63-year-old man who has persistent angina. The man has undergone three coronary-artery–bypass operations; he is on a regimen of four concurrent medical therapies, and he still experiences chest pain with only a small amount of exertion. The man's physician must weigh the relative merits of continued medical management against percutaneous transluminal coronary angioplasty (PTCA) with possible stent placement should the

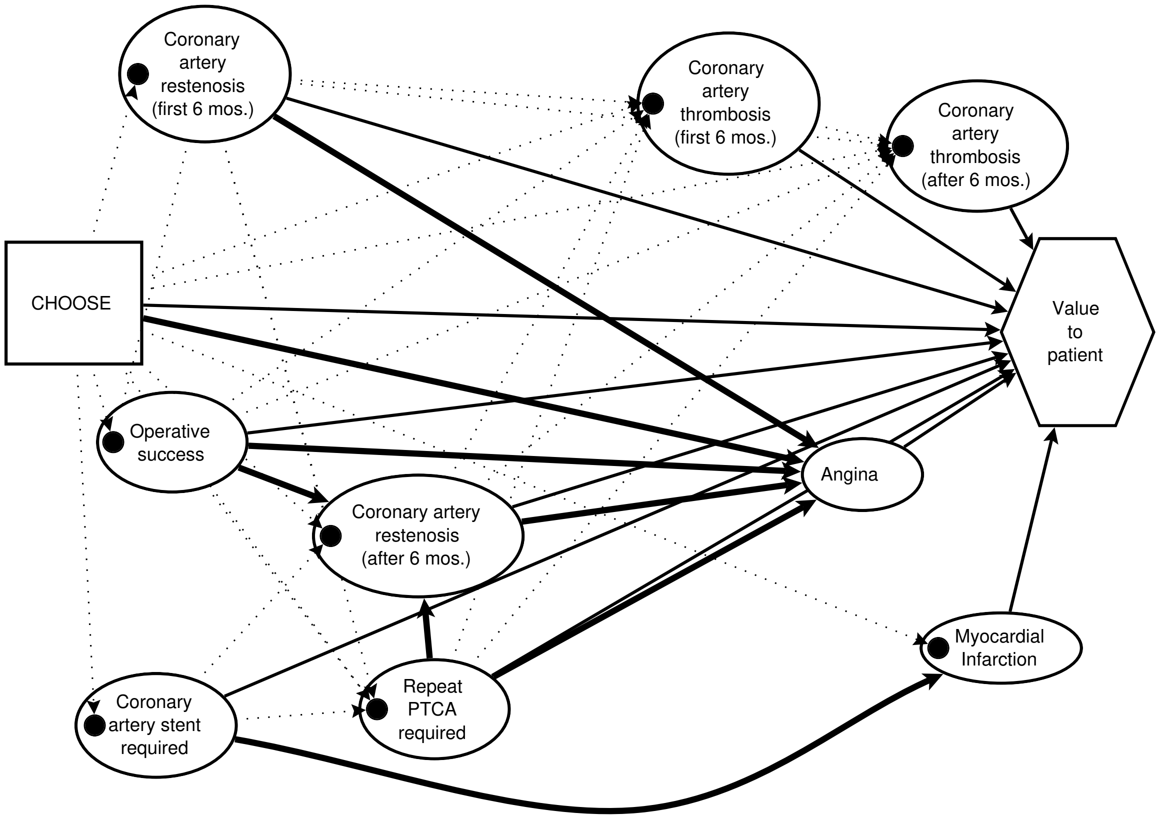

Figure 5.18: The manually composed model for case 6. Arcs with target nodes other than Value to patient are thickened for visual clarity.

angioplasty be unsuccessful. The physician considers restenosis, thrombus formation, acute myocardial infarction, and procedure-related mortality to be risks associated with the latter plan of action.

The manually composed model for case 6 is shown in Figure 5.18. The considerations and their context-ignorant classifications are given in Table 5.6. I introduced no additional concepts that were not in the decision-tree model, and I reclassified Myocardial infarction, which I had classified previously as a ⟨crisis event⟩, rather than as a ⟨present malady complication⟩. I replaced the manual model's consideration Operative

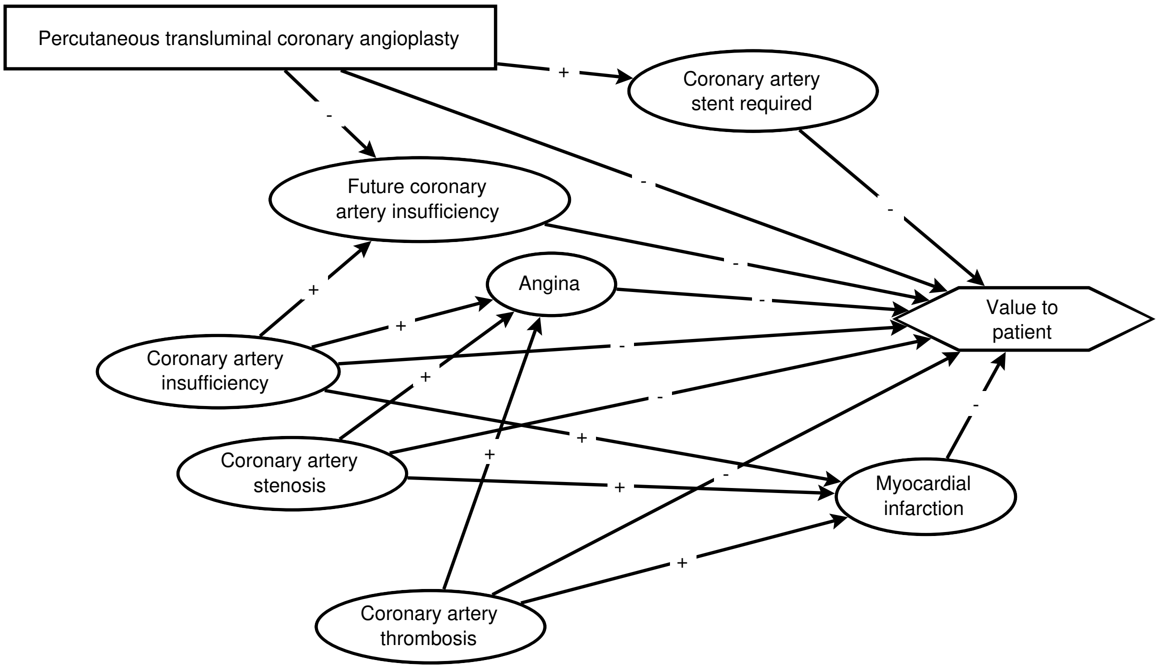

Figure 5.19: The model constructed automatically by Gramarye for case 6. (PTCA = percutaneous transluminal coronary angioplasty; mos. = months.)

success with the related concept Coronary artery insufficiency.

Consideration	*Classification*
Coronary artery insufficiency	⟨present disease⟩
Coronary artery stenosis	⟨present disease⟩
Coronary artery thrombosis	⟨present disease⟩
Coronary artery stent required	⟨treatment complication⟩
Myocardial infarction	⟨present malady complication⟩
Angina	⟨present malady complication⟩
Percutaneous transluminal coronary angioplasty	⟨curative treatment⟩

Table 5.6: Considerations entered into Gramarye for case 6.

Given this list of considerations from the manual model, Gramarye derived the model in Figure 5.19. Gramarye required only three user selections for assistance during this derivation. Note that modifiers such as "after 6 months," and general considerations such as Operative success have no corresponding node-label abstraction in the current grammar. Consequently, these notions are absent from the terms entered and from the model that Gramarye generated.

In contrast to the manual model's 23 contingency arcs, Gramarye's model has none. The arc into Coronary artery stent required could be modeled more accurately as a contingent arc, since the stent was proposed as a postangioplasty reparative procedure. Some of the contingent arcs in the manual model result from its separation of the postintervention periods into before and after 6 months. The current grammar would need a general representation for temporal intervals to make such modeling features explicit at the structural level. Other contingent arcs, such as those linking the treatment decision to the probability distributions of thrombosis and stenosis, would be modeled more accurately as probabilistic arcs. In a context-ignorant setting, Coronary artery thrombosis and Coronary artery stenosis were classified as primary diseases, and, consequently, Gramarye omits arcs to them entirely.

Gramarye includes arcs to Myocardial infarction from all three coronary artery diseases: Coronary artery insufficiency, Coronary artery thrombosis, and Coronary artery stenosis. Since these arcs are the result of specific user selections, the current grammar can be neither blamed nor credited for this difference between Gramarye's model and the manual model.

5.4.7 Case 7

The seventh decision problem that I used for testing involves a 28-year-old pregnant woman who has a membranous ventricular septal defect. She and her physician must decide whether they should terminate this pregnancy, and then repair her heart defect before any subsequent pregnancies.

From the manually composed model in Figure 5.20, I entered the list of considerations shown in Table 5.7 into Gramarye. This derivation required no reclassification of terms and no assistance from the user during the derivation. I added the concept High-risk pregnancy since no disease was included in the manual model. The derivation resulted in the model shown in Figure 5.21.

Consideration	*Classification*
Mortality of pregnancy for mother	⟨mortality⟩
Morbidity for mother	⟨morbidity⟩
Mortality for baby	⟨mortality⟩
Morbidity for baby	⟨morbidity⟩
High-risk pregnancy	⟨present disease⟩
Abortion	⟨ablative treatment⟩

Table 5.7: Considerations entered into Gramarye for case 7.

In this derivation, Gramarye's model differed from the manual model in only the former's explicit representation of High-risk pregnancy. The merit of this addition to the manual model cannot be inferred from my own experiments, but I suspect that, in this case, the additional disease and future-disease nodes are not helpful to the assessment process.

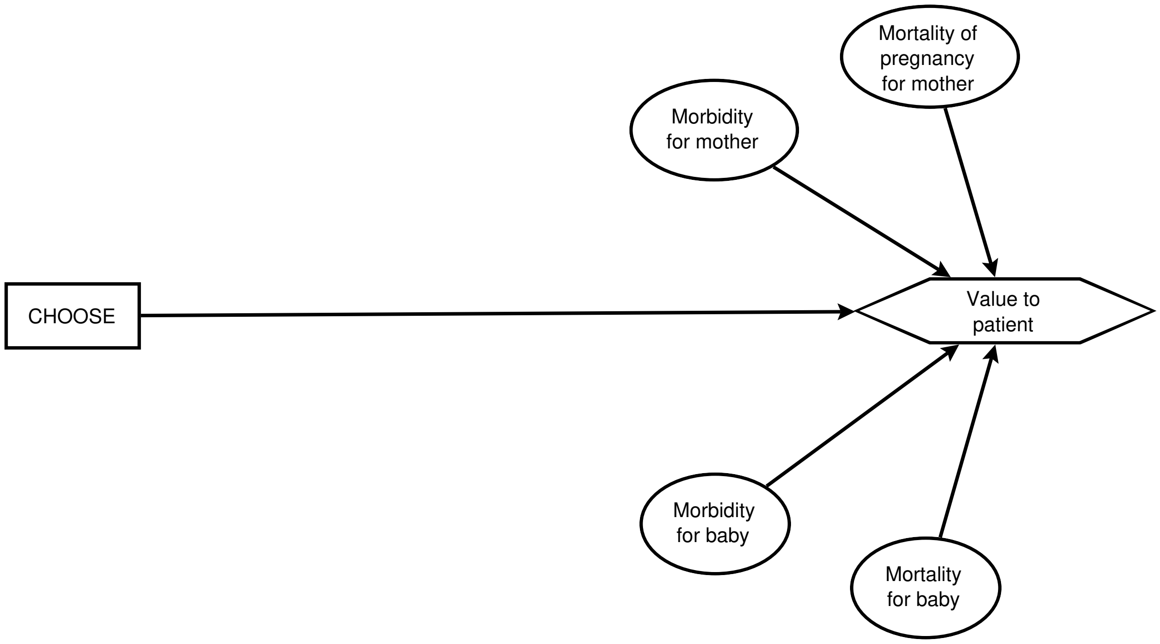

Figure 5.20: The manually composed model for case 7.

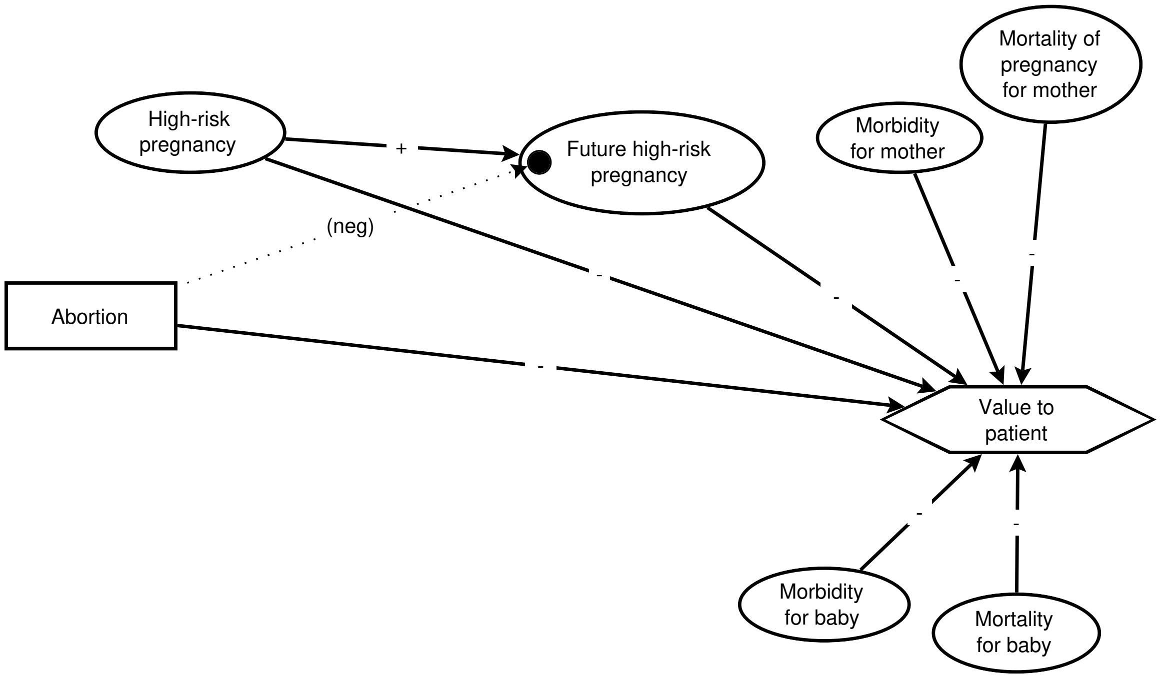

Figure 5.21: The model constructed automatically by Gramarye for case 7.

5.4.8 Case 8

The eighth decision problem involves a 42-year-old woman who was treated for Hodgkin's disease 12 years previously with 4500 Rads of radiation. She presents evidence of having a thyroid carcinoma and cardiomyopathy, both of which possibly are results of past radiation therapy. Her physician must decide whether to recommend surgical removal of her thyroid, despite the additional risks of surgery due to her cardiomyopathy.

For case 8, the manually composed model is shown in Figure 5.22. From this model, I entered the list of considerations shown in Table 5.8 into Gramarye. This derivation required no reclassification of terms, no additional concepts, and two selections from the user during the derivation. The derivation resulted in the model shown in Figure 5.23.

Consideration	*Classification*
Thyroid cancer	⟨present disease⟩
Type of thyroid cancer	⟨present disease⟩
Surgical mortality	⟨treatment complication⟩
Mortality over 5, 10 years	⟨mortality⟩
Thyroidectomy	⟨ablative treatment⟩

Table 5.8: Considerations entered into Gramarye for case 8.

In this case, Gramarye's model shows a few subtle differences from the manual model. Gramarye's node Type of thyroid cancer is not contingent on the Thyroid cancer. This example illustrates the need for models of a disease more detailed than

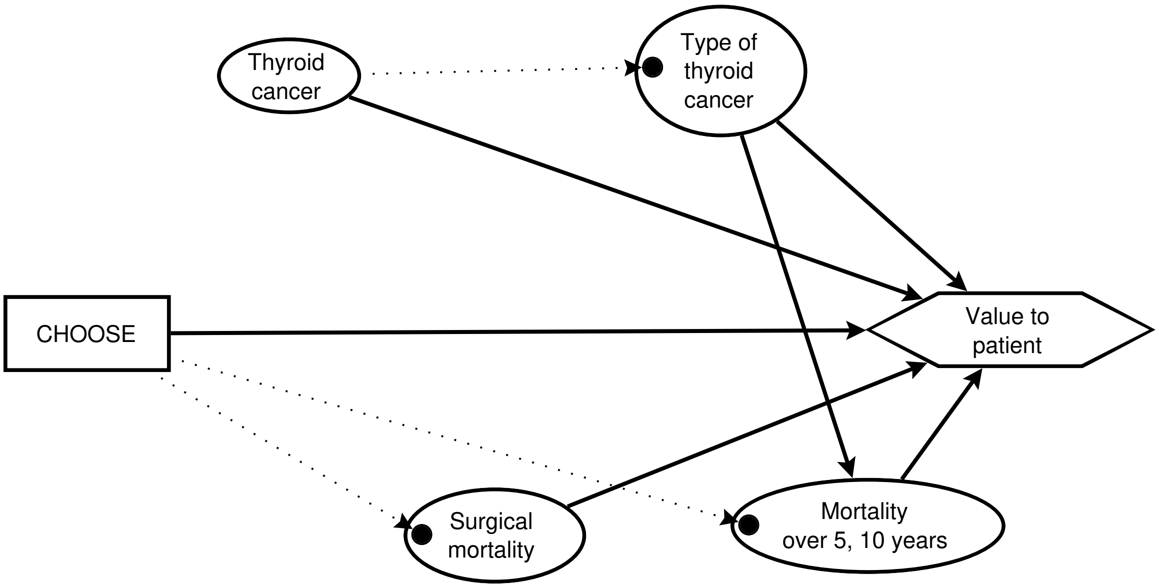

Figure 5.22: The manually composed model for case 8.

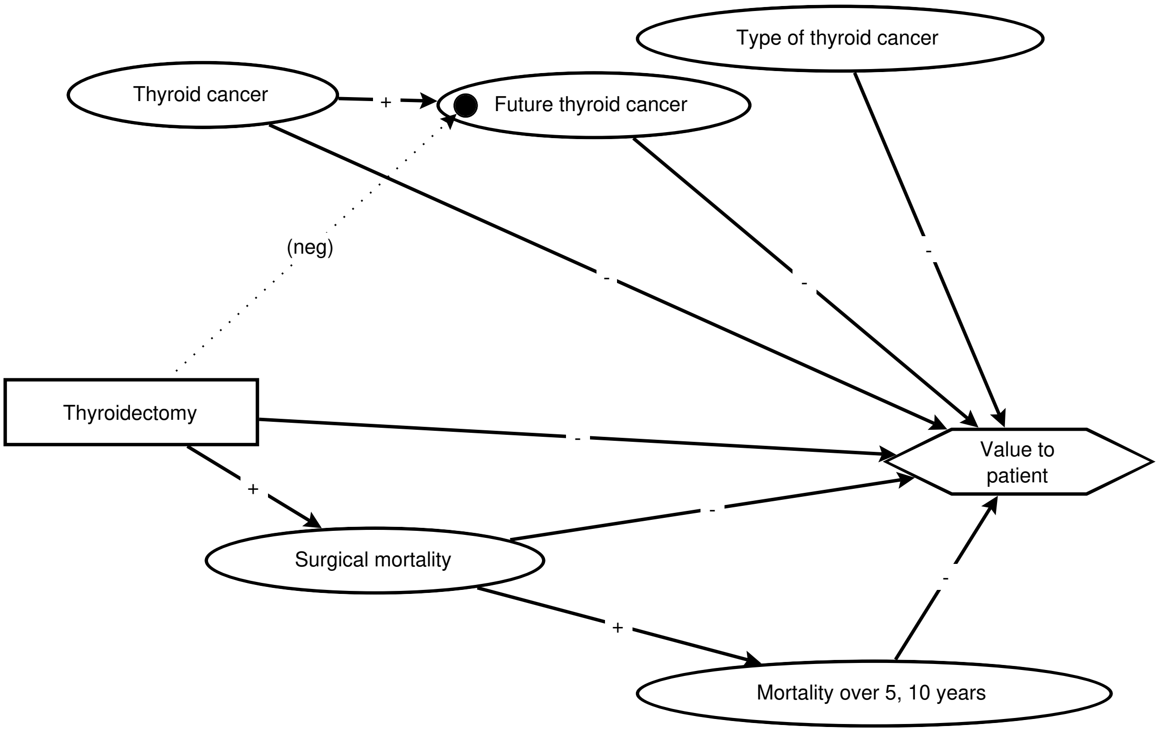

Figure 5.23: The model constructed automatically by Gramarye for case 8.

the single random variable. Pradhan[11] has suggested that a separate diagram might model the distributions for severity and type of any disease that has a nonnegligible probability of existence. In any event, both the manual model and the automatically generated model exhibit awkwardness in the modeling of Type of thyroid cancer as a separate event from Thyroid cancer. The current grammar requires a general abstraction for disease descriptors to model considerations such as Type of thyroid cancer appropriately. Although Gramarye's model is less explicit about the relationship between Thyroid cancer and Type of thyroid cancer, the utility function can be formulated to make the two models equivalent in this respect.

The relationship between Surgical mortality and Mortality over 5, 10 years is somewhat more evident in Gramarye's model. However, this difference is partly an artifact of generating the model in Figure 5.22 from a decision tree, where the two nodes are implicit in the surgical outcome node and in various utility nodes.

Gramarye's addition of a future disease state may help to clarify the problem for the user during the assessment phase of decision analysis, but this difference is primarily stylistic, rather than substantial, as discussed in case 1. However, Gramarye makes what is arguably a mistake in omitting the arc from Type of thyroid cancer to Mortality over 5, 10 years that is present in the manual model. This omission is the result of the current grammar's poorly integrated abstraction ⟨mortality⟩, as discussed for case 2.

[11] Malcolm Pradhan. Personal communication.

5.4.9 Case 9

The ninth of the decision problems that I used for testing involves a 58-year-old woman who has polymyalgia rheumatica and headaches suggestive of temporal arteritis. An initial temporal-artery biopsy has shown no definitive signs of temporal arteritis. Her physician must decide whether her symptoms—and the risk of blindness from temporal arteritis—merit additional biopsies, empiric steroid treatment, or mere observation.

From the manually composed model in Figure 5.24, I entered the list of considerations shown in Table 5.9 into Gramarye. This derivation required no reclassification of terms, no additional concepts, and two selections from the user during the derivation. With this derivation, Gramarye generated the model shown in Figure 5.25.

Consideration	*Classification*
Temporal arteritis	⟨present disease⟩
Mortality	⟨mortality⟩
Morbidity due to steroid treatment	⟨treatment complication⟩
Blindness due to temporal arteritis	⟨present malady complication⟩
Biopsy of temporal artery	⟨test⟩
Steroid treatment	⟨curative treatment⟩

Table 5.9: Considerations entered into Gramarye for case 9.

In Gramarye's model, Mortality is disconnected erroneously from disease states and treatment decisions. This mistake stems from the current grammar's inadequate development of the ⟨mortality⟩ abstraction, as discussed in cases 2 and 8.

Gramarye's current grammar makes explicit future disease states and decision

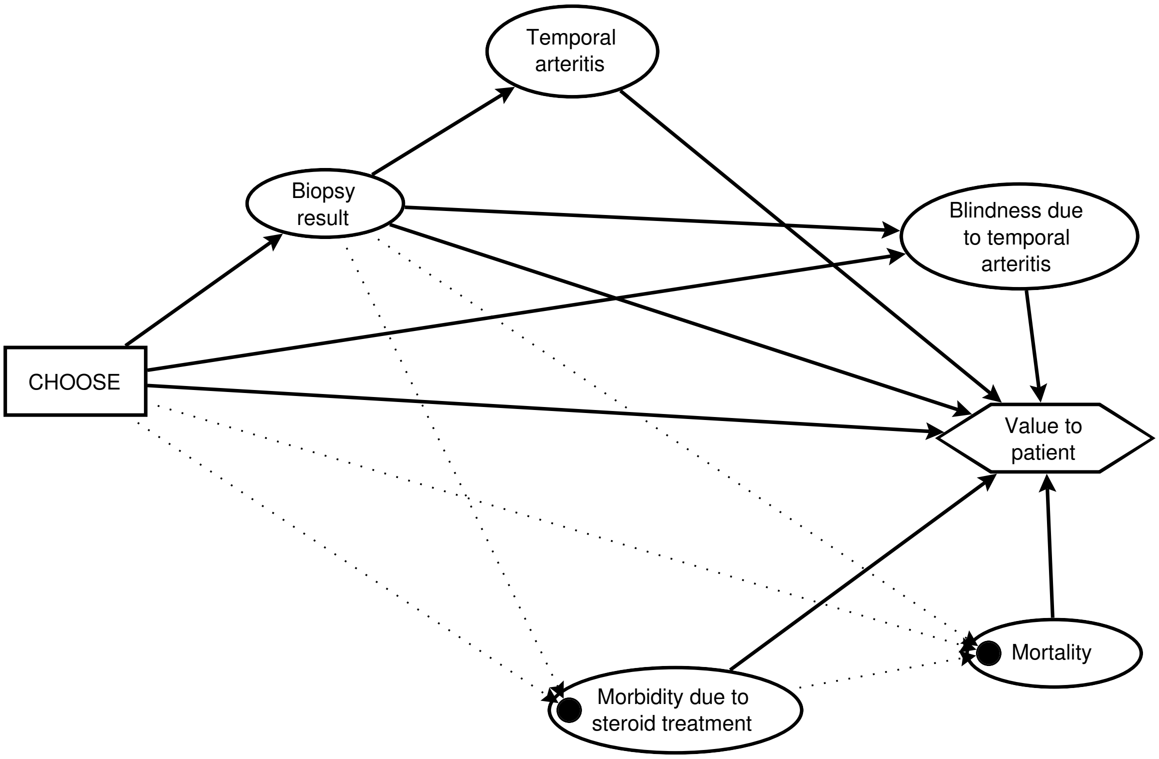

Figure 5.24: The manually composed model for case 9.

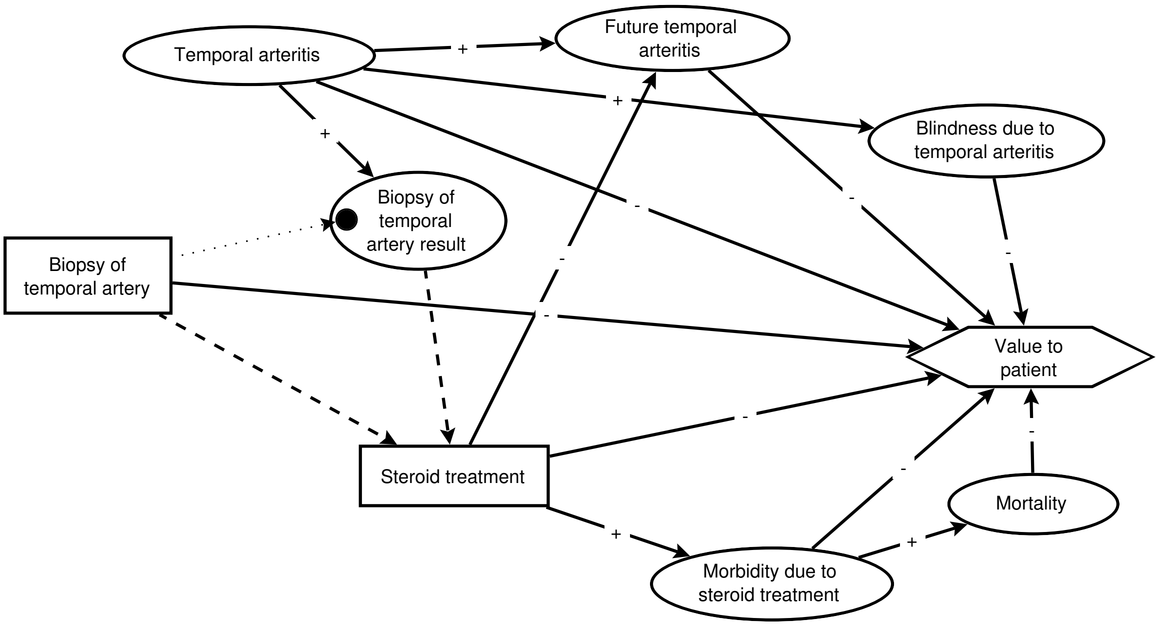

Figure 5.25: The model constructed automatically by Gramarye for case 9.

alternatives. However, as discussed in case 1, this difference between Gramarye and the manual model is primarily stylistic.

Gramarye does not make the utility function, Value to patient, dependent on the biopsy result. I do not regard this statement of independence as an error, because Value to patient is a function of the disease state and of the treatment decision, rather than of the informative node Biopsy of temporal artery result per se.

It might have been desirable for Gramarye to have included an arc from Future temporal arteritis to Blindness due to temporal arteritis. The current grammar's addition of ⟨present malady complication⟩ nodes should be replaced by a more robust addition of *general* malady complications, both present and future. This modification to the grammar would need to include a staging design that would delay the addition of any such ⟨complication⟩ nodes until after all ⟨treatment⟩ nodes have been added.

Gramarye's model replaces many of the contingencies present in the manual model with the more general probabilistic arcs. As discussed in Section 5.4.1, this difference is relatively minor, and does not need to affect the behavior of the assessed decision model.

5.4.10 Case 10

The final decision problem that I used in this evaluation involves a woman who has a brain lesion, which may represent a glioma (brain cancer). She also presents signs of a recent pulmonary embolism. Her physician must decide whether she would benefit from pulmonary angiography—used to confirm the diagnosis of pulmonary embolism—and whether she should undergo heparin treatment to reduce the risk of subsequent pulmonary emboli. The heparin therapy carries the risk of a potentially disabling or fatal hemorrhage. The additional risk of cerebral hemorrhage due to her glioma is unclear to the decision maker.

The manually composed model for case 10 is shown in Figure 5.26. The considerations and their context-ignorant classifications are given in Table 5.10. I introduced the single new concept Brain tumor. Gramarye's derivation of the model in Figure 5.27 required no reclassifications of terms, and the derivation required only one clarifying selection from the user.

Consideration	*Classification*
Pulmonary embolism	⟨crisis event⟩
Brain tumor	⟨present disease⟩
Mortality	⟨mortality⟩
Major hemorrhage due to heparin treatment	⟨treatment complication⟩
Disability due to hemorrhage	⟨morbidity⟩
Pulmonary angiography	⟨test⟩
Heparin treatment	⟨risk-reducing treatment⟩

Table 5.10: Considerations entered into Gramarye for case 10.

In this case, Gramarye's model includes a patently erroneous arc from Brain tumor to Pulmonary angiography result. This mistake is due to the current grammar's requirement that ⟨test result⟩ nodes provide information about ⟨present disease⟩ nodes, rather than about a ⟨crisis event⟩. This issue is complicated by the fact that the entered list

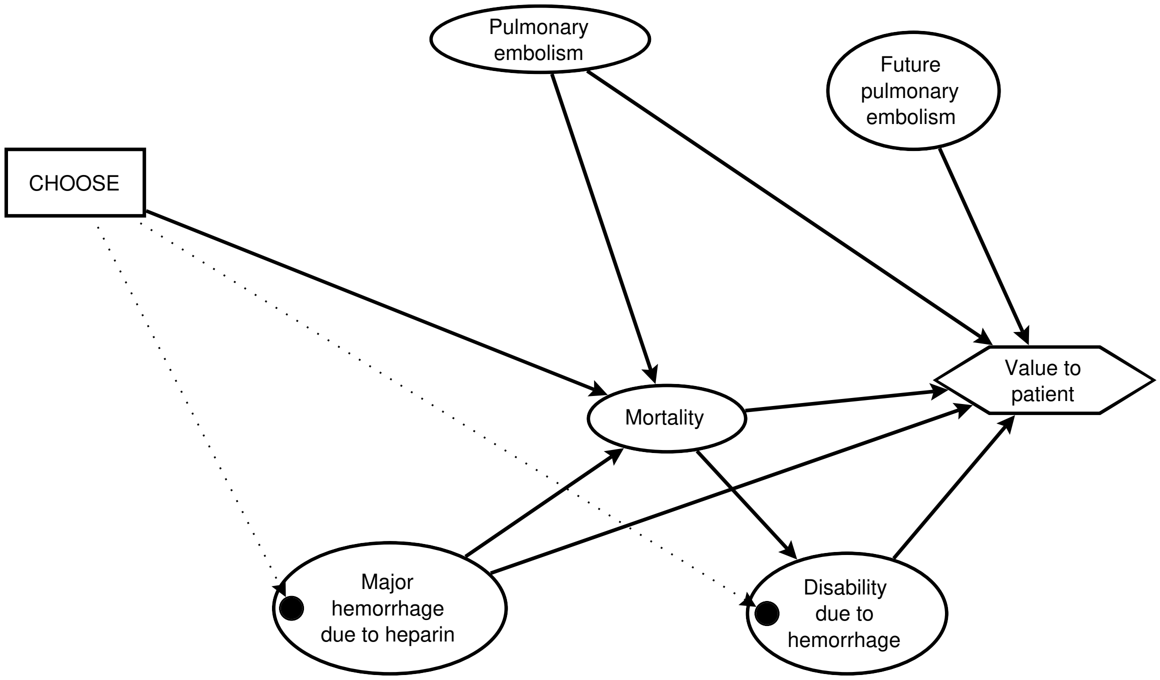

Figure 5.26: The manually composed model for case 10.

of considerations includes only a single term for pulmonary emboli, whereas the problem is concerned with two types of pulmonary emboli: There is the current episode of suspected pulmonary embolism, about which the angiography test may provide information, and there is the possibility of pulmonary emboli in the future, for which heparin is being considered as a risk-reducing treatment. This distinction might have been clearer to me had I used more descriptive labels for the manual model's nodes Pulmonary embolism and Future pulmonary embolism. Nonetheless, this example highlights Gramarye's need for a knowledge representation more descriptive than QCID's for describing current and future events. The modification of the QCID formalism to include temporal relationships, other than those indicated by informational arcs,

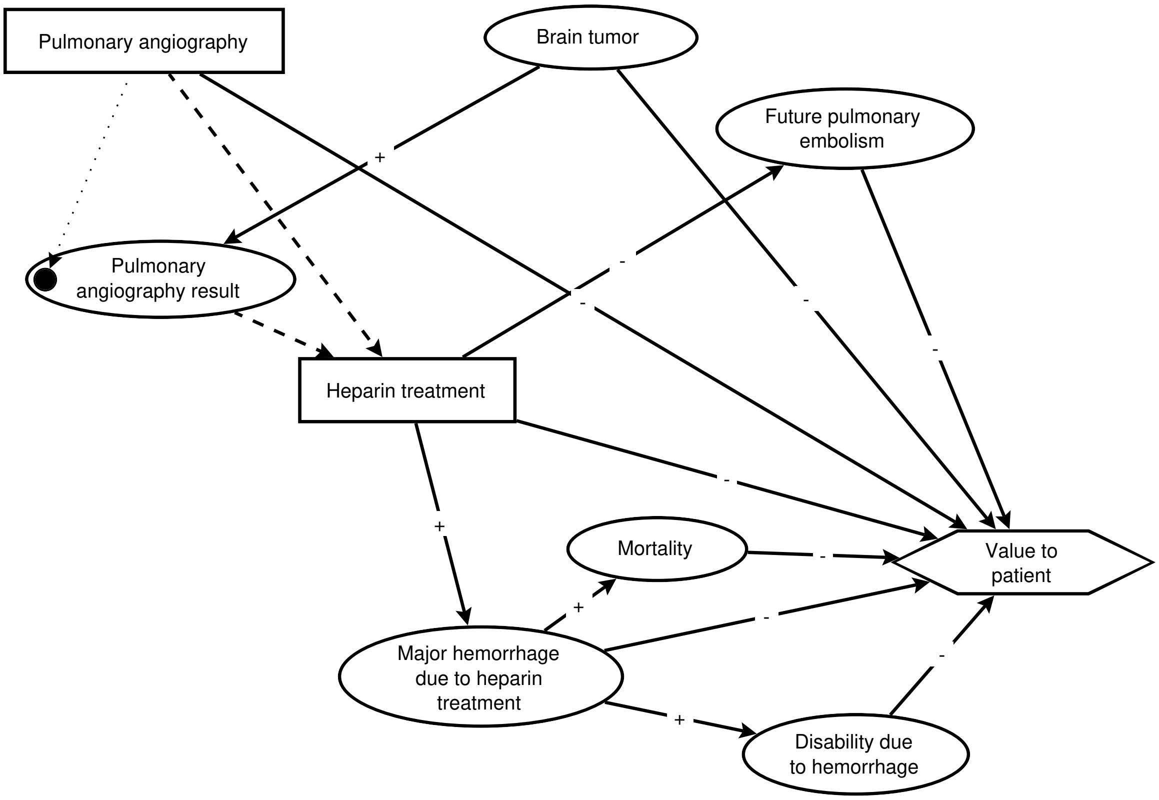

Figure 5.27: The model constructed automatically by Gramarye for case 10.

might help decision analysts to distinguish structural patterns for events that have happened and can be detected from patterns for events that can be prevented.

Gramarye's model omits arcs from Pulmonary embolism to Mortality, and from Mortality to Disability due to hemorrhage. For the latter arc, I regard the manual model's explicit representation of the trivially deterministic relationship between death in the near future and disability in the more distant future as either a contingent relationship, or one that might easily be left implicit and inside the utility function. The absence of an arc from Pulmonary embolism to Mortality is the result of the lack of a node for Pulmonary embolism in Gramarye's model. Again, the node missing is not the instance of Pulmonary embolism that Gramarye removed in the process of adding the risk-reducing treatment Heparin treatment; the missing node corresponds to the *current* embolic event, and, in the list entered into Gramarye for this case, there is no term corresponding to the current embolic event.

5.5 Differences Due to Graphical Representation

Of the differences that distinguish Gramarye's decision models from the manually composed test models, some of these differences are due to the manual models' inception as decision trees. Since the decision-tree formalism emphasizes combinations of outcomes, many more contingencies are present in the manual models. These contingencies may reflect reasonable abbreviations to the Cartesian product of all chance and decision-node outcomes. However, it is easy for an analyst using the decision-tree formalism to dismiss combinations of outcomes that may be worth considering. One example of such an omission is in case 6, where the manual model considers myocardial infarction to be worth considering only if a surgical intervention has been performed. The patient in this case had an occluded left anterior descending and right coronary arteries, and his circumflex artery was 80-percent stenosed. Given the description in the case history, I suspect that the modeler inadvertently omitted the possibility of myocardial infarction for nonsurgical interventions.

Another tendency that the decision-tree formalism promotes is the modeling of outcomes associated with more than one physical entity. For example, several of the manual models modeled the outcomes of tests as true positive, true negative, false positive, and false negative. In some cases, such as cases 9 and 10, the false positive and false negative were omitted completely, giving the user infallible tests that make prior probabilities inconsequential.

Decision trees hide the actual structure of probabilistic dependency in binding assignments, which often appear in inscrutable locations in the decision tree. I have not investigated the level to which this feature of the Decision Maker modeling environment may have led to questionable modeling practices. Indeed, I have deliberately resisted emphasizing Gramarye's ability to avoid such mistakes, since the mistakes often are not apparent in the structural model, and since they probably would not occur for modelers who use the influence-diagram notation.

On the other hand, I have noted where the QCID formalism lacks the expressiveness that would make relationships in the decision model more apparent at the graphical—rather than at the numeric—level. The QCID representation can express only three relationships: probabilistic relevance, contingency, and information availability. Additional relationships that could express temporal and anatomical relationships between concepts would enhance the clarity of the QCID model and might lead to additional prototypical patterns that could be incorporated into the graph grammar. For example, an explicit distinction between past and future episodes of pulmonary embolism might have helped Gramarye to avoid several of the modeling mistakes that it made in case 10.

5.6 Usefulness

In this evaluation of Gramarye, I used 10 manually composed models, each with an average of seven nodes and 14 arcs (Table 5.11). The average derivation took six considerations as input, and required two selections from the user, and, on average, less than a single reclassification or addition from the user for each case (Table 5.12). These data provide only an indirect answer to the fundamental question: How useful was Gramarye to the modeling of medical decisions?

Gramarye's usefulness depends on an appreciable fraction of a model's arcs—or

Case	*Manual Model*		*Gramarye's Model*	
Case	*Nodes*	*Arcs*	*Nodes*	*Arcs*
Case 1	9	19	11	22
Case 2	7	12	7	11
Case 3	7	11	10	17
Case 4	3	2	7	10
Case 5	10	19	8	19
Case 6	11	41	9	17
Case 7	6	5	8	9
Case 8	6	9	7	10
Case 9	7	15	9	16
Case 10	7	11	9	15
Total	73	144	85	146

Table 5.11: Sizes of models used and created in the evaluation.

lack of arcs—being the result of the grammar's design, rather than the result of user selections at derivation time, or of classifications that the *user* performs prior to the derivation. Table 5.12 provides an indication that the grammar delivered significant guidance to each derivation. However, it is also apparent that both user selections and term classifications contributed appreciably to Gramarye's derivations of models for the 10 cases.

If I had access to decision analysts who (1) focused on medical decision for individual patients; (2) used the influence-diagram notation; (3) structured their models in a single, initial step; and (4) were willing to enter terms into Gramarye prior to structuring their decision problems, then I might have examined the marginal time saved—or spent—by decision analysts who use a graph-grammar derivation system. I might also have looked at the relative merits of models that were created with and

Case	*Entered terms*	*Selections*	*Reclassifications*	*Additions*
Case 1	9	5	3	1
Case 2	4	0	0	0
Case 3	9	4	2	1
Case 4	4	2	0	0
Case 5	6	1	0	0
Case 6	7	3	1	0
Case 7	6	0	0	1
Case 8	5	2	0	0
Case 9	6	2	0	0
Case 10	7	1	0	1
Total	63	20	6	4

Table 5.12: User assistance required for case problems in the evaluation.

without Gramarye's assistance. However, Gramarye's current grammar is easy for me to comprehend and imitate, and I suspect that decision analysts who used Gramarye and looked at its simple grammar could learn to make the same modeling decisions quickly, without using the actual derivation system.

With Gramarye, I was able to develop easily a reasonable initial model for each of the 10 decision problems, and I was able to perform all this modeling is less than 1 day. Without Gramarye, decision analysts have spent several weeks structuring a single decision model that is no larger than the average of Gramarye's models for these 10 cases. However, human analysts do not begin modeling with the help of a finite list of considerations. Also, human analysts could probably increase their modeling speed by using the same prototypical patterns that Gramarye used. Consequently, for decision analysts who might use a graph-grammar derivation system such as Gramarye, the time that they could save is difficult to estimate from this research. My suspicion is that the current technology would not prove useful to professional decision analysts; however, it probably would assist users who are less experienced at decision modeling.

Gramarye's current implementation lacks the ability to guide the user to make specific reclassifications and to add missing considerations. The command-line dialogue for making selections also warrants improvement. Users should be able to see the host graph as it evolves during the derivation, and should be able to select node–vertex matches *graphically*, rather than communicating through a series of obtuse true–false questions. The application of graph-grammar productions could be animated, so that users could correct errant derivations steps as those steps occur. Also, although Gramarye can generate structural models for several actual decision-analysis tools, the assessment of variables—particularly the association of specific outcomes and contingencies to those variables—is often a process that is more naturally integrated with the structuring of a model, in contrast to Gramarye's isolation of assessment as a subsequent step in the decision-analysis cycle.

Graph-grammar derivation systems such as Gramarye could encourage the iterative nature of decision modeling. Although Gramarye's grammar checker assumes that the initial graph for each derivation is the single utility node, a user may use a previously derived—and, perhaps, user-modified—model as the initial graph for subsequent derivations, which could incorporate just a few *additional* considerations. Alternatively, the derivation system could be modified to retain all user selections and postderivation modifications. With these modifications, if the user wanted to add or delete a few concerns, then the repeat derivation could begin from the single initial utility node, and the repeat derivation would not require any repetitive input from the user.

From my own evaluation, I can report only that the graph-grammar approach to structuring decision models has been helpful in defining a style and a set of modeling rules for medical decision making. I cannot establish that, for users other than myself, the reward of automatic model generation is worth the chore of listing considerations, classifying those considerations, modifying the list, reclassifying terms, and providing assistance during the derivation.

The foregoing evaluation focused on a particular grammar, yet it provided insights into my graph-grammar approach to modeling. Gramarye has merits that are independent of the particular graph-grammar used. For example, the grammar-checking routines may be useful for all QCID grammars, even for those outside of the medical domain. The compiler that translates decision trees into QCIDS provides a useful means for communicating models between two communities of decision-analysis research. The translator that converts abstract graphs (i.e., a Lisp S-expression composed of a list of node labels and a list of arcs) into editable graphs has been a useful tool for examining many types of graphs. The translators that convert these editable graphs into the requisite belief-network and influence-diagram formats for several academic and commercial tools may help to make graphical models of probabilistic dependence more accessible to other researchers. Finally, the actual graph-grammar derivation system may be used as a research tool for other generative grammars, where the graphical knowledge representation may be completely unrelated to the QCID formalism. My research has focused on merely a single application of the Gramarye graph-grammar derivation system.

5.7 Summary

My evaluation has established that graph grammars can be used to generate automatically decision models that are comparable to those produced by decision analysts. The **scope** of the current grammar's applicability has been all of medicine, and the 10 cases—which were chosen by an independent collaborator who had only a limited familiarity with Gramarye—were completely within the grammar's domain. The **user interface** lacks guidance for users who are unfamiliar with the grammar itself. Gramarye has demonstrated its ability to **interoperate** with other decision-model representations (i.e., decision trees), and with several decision-model assessment and inference tools. The models that Gramarye generated lack the **validity** of those that humans created, but they provide a reputable initial model for the user to modify by adding or subtracting a few arcs. The overall **usefulness** of this particular graph-grammar derivation system for medical decision analysis is questionable. Nonetheless, Gramarye provides a useful research tool for studying modeling skills and prototypical patterns found in decision analysis.

Chapter 6

Conclusions

Truth, n. An ingenious compound of desirability and appearance.

—*Ambrose Bierce, The Devil's Dictionary*

The construction of qualitative models from prototypical patterns appears to be a feasible task for a graph-grammar derivation system. My limited experience with deriving fairly complex decision models using Gramarye has met with qualified success. A more important result of my research has been insight into the strengths and weaknesses of the graph-grammar approach to medical decision modeling. Gramarye has illustrated how graph grammars can formalize medical abstractions and patterns, and how these graph grammars can automate portions of modeling task for a wide range of specific situations.

6.1 Medical Abstractions and Patterns

Medical abstractions and patterns can be represented in a graph grammar. Once these patterns are encoded as a graph grammar, a graph-grammar derivation system, such as Gramarye, can use these patterns to provide automatic modeling of influence diagrams. The current grammar of 15 prototypical patterns provided guidance for the entire spectrum of decision problems represented by the 10 test cases in Section 5.4.

The grammar that I devised (Section 3.6.1) has centered on patterns that distinguish different types of decision nodes. Other patterns that could make distinctions among chance nodes based on typical pathophysiologic roles might lead to more sophisticated modeling. Once such patterns are recognized and defined, there is still the problem of finding a label for the class of nodes to which that pattern applies; as discussed in Section 5.1, clinical parlance may not have the abstract terms for the topological motifs that are described in a grammar. Consequently, the current grammar has emphasized the relationship between common clinical abstractions, such as palliation and prevention, and has not investigated general patterns that lack descriptors in common medical parlance.

Whether additional patterns—employing temporal, physiologic, and anatomic relationships that are not available in the QCID representation—might help to define concepts in medical vocabularies and in medical ontologies is an interesting topic for future research. The distinctions made in medical decision modeling should be represented in medical vocabularies, but whether any sizeable fraction of concepts in a medical vocabulary can be *defined* by a pattern in a graphical knowledge representation is unclear.

Other researchers are currently investigating the possibility that canonical patterns in the conceptual-graph representation may lead to a principled representation of the medical record (Campbell et al., 1994). If this approach to the formalization of the medical record is feasible, then a graph-grammar derivation system may provide the necessary mechanism for generating complex medical histories from canonical building blocks.

Gramarye's predefined medical vocabulary covered few of the concepts used in the 10 test cases from Section 5.4. Given the frequent instances where the classification for a particular term depended on the context of the situation, further elaboration of the vocabulary may not be helpful to Gramarye's performance. Instead, either the user of Gramarye should be given additional help in classifying new considerations for each derivation, or the input should be more expressive than the simple list of terms. A formal knowledge representation for medical records might provide a system such as Gramarye with the necessary information for context-specific classification of terms under decision-modeling abstractions. Since the abstractions in the current grammar might be misinterpreted easily by a user who is unfamiliar with the grammar, I am convinced that the classification task would have to be automated before a system such as Gramarye could become useful to medical decision analysts.

The particular modeling mistakes that Gramarye made with the test cases from Section 5.4 might be avoided by specific repairs to the current grammar. However, the necessary elaborations to the node-label classification tree would make the task of classification even more difficult for the user, since these elaborations would create additional abstractions under which the user could misclassify clinical considerations.

6.2 Modeling Capabilities of Graph Grammars

The modeling capabilities of graph grammars extend beyond automated structuring of medical QCIDs. Göttler's operational graph-grammar formalism provides a terse depiction of prototypical patterns of relationships in a graphical knowledge representation. These prototypical patterns have the operational semantics (as described in Section 3.3 and in Appendix A) that allow derivation systems to use these patterns as building blocks, and thereby to generate graphical models.

Also, automated routines can evaluate a graph grammar to ensure that the latter maintains specific properties for all models in the language that it describes. My research (Chapter 4) has shown how a QCID grammar may be checked to ensure that it maintains several properties that I have considered important in decision models. Certain of these properties, such as acyclicity, are useful in other knowledge representations, such as computer-assisted software engineering (Engels et al., 1986), and database management (Ehrig and Kreowski, 1980).

Decision analysts may find that the formalization of prototypical patterns as a graph grammar may have value that is independent of the use of a graph-grammar derivation system. Graph grammars provides decision analysts with a language for recording features of decision models in a particular domain. Once the decision analyst learns to think in terms of these patterns, the actual task of structuring a model can be simplified, and the analyst's need for computer-assisted model structuring is lessened. Moreover, analysts may communicate their preferred styles to one another through such a grammar.

The current grammar for QCID models enforces a particular modeling style. My work with Gramarye has convinced me that such a style can be flexible enough to accommodate a wide variety of situations. When a graphical knowledge representation adheres to a particular style, then parsing of a particular graphical model may be simple and unambiguous. When a graphical knowledge representation does not adhere to a particular style, then subtleties and ambiguities may be difficult for computer-based systems to interpret. By enforcing style conventions, graph grammars may provide a useful guide for graphical knowledge representations.

Because all information that is specific to a particular knowledge representation and domain is contained in the graph grammar, a graph-grammar–based modeling environment, such as Gramarye's, may adapt to entirely different domains and graphical representations. Although my research has not investigated any representations other than the QCID representation, the scope of patterns defined for medical decision making may help other researchers to find patterns in their own representation, and to use Gramarye to test how those patterns interact and form coherent models for their domain.

The basic intuition behind graph-grammar–based modeling is that prototypical patterns found within models can be reused to guide the construction of new models. This notion of reuse has been advocated for human problem solving (Polya, 1945, pages 37–46), machine learning (Carbonell, 1983), knowledge acquisition (Musen, 1992), software engineering (Prieto-Díaz, 1993), and the entire spectrum of modeling tasks. The fundamental insight that Gramarye can offer to these creative endeavors is that, for graphical models, Göttler's operational graph-grammar formalism can provide

1. *Computational clarity* necessary for automated modeling systems to create models from prototypical patterns

2. *Visual clarity* necessary for knowledge engineers to design and understand modeling grammars that are based on these prototypical patterns

6.3 Summary

In summary, I have found that my graph-grammar production system, with the current grammar of 15 prototypical patterns, supports automatic modeling of medical dilemmas. Graph grammars address relationships between medical concepts other than lexical ordering; consequently, graph grammars are ideally suited for deriving a decision model from an unordered list of medical concerns. The grammar that I employ does not ensure that the models produced are appropriate for a practitioner's problem, but it does guide the practitioner to compose reasonable initial models of medical decision making.

Appendix A

Göttler's Graph-Grammar Formalism

In terms of Barthelmann's notation (1990), I define a labeled directed graph as comprising three sets:

1. A set of vertices (V), with labels (L_V) and a mapping ($l_V : V \rightarrow L_V$) from vertices to their labels

2. A set of permissible edge labels (L_E)

3. A set of labeled directed edges ($E \subseteq V \times V \times L_E$)

A.1 Connected Vertices

A **spanned subgraph,** $\text{span}(V', G)$, of the host graph G and spanning vertices V' consists of vertices ($V' \subseteq V_G$), their labels (L_{V_G}), the edges between vertices in V' ($V' \times V' \times L_{V_G} \cap E_G$), their labels ($L_{E_G}$), and the restricted labeling function $l_V(V')$. A **chain** is a sequence of edges, without regard to their direction. So, a chain between v_0 and v_N exists if and only if

$$(\forall i \in \{1, \ldots, n\}) \quad (\exists m \in L_E)((v_{i-1}, v_i, m) \in E \vee (v_i, v_{i-1}, m) \in E). \tag{A.1}$$

For a given vertex v in production p, and for a given set of vertices V_X, all the vertices v' that are reachable by some chain through vertices in V_X are considered **connected** to v, and this relation is denoted $v \sim_p v'$. I shall use the connection relation to determine which of the vertices in the indeterminate region are to be mapped to the host graph, where I shall add edges as specified in the production.

A.2 Applicability of a Production

If all vertices in the left and bottom regions of the graph-grammar production are matched to nodes in the host graph, and if the edges among the vertices in the left and bottom regions of the production are matched to corresponding edges in the host graph, then the derivation system can apply the rule. However, it must note which nodes and arcs in the host graph match vertices and edges in the indeterminate region and are connected, by the reflexive $\sim_p$ relation described in Section A.1, to the nodes and edges that it has already matched to the left and bottom regions of the production.

A **monomorphism** δ from P to G, written $\delta : P \to G$, is a complete mapping from vertices and edges in a production to host-graph nodes and arcs such that the directed-graph structure and the arc labels are equivalent, and either the node labels are equivalent, or one node label is a descendant of the other in the node-label classification hierarchy.[1] A production rule p is **applicable** to some subgraph of G indicated by the monomorphism $\delta : \mathrm{span}(V_{\mathrm{L},p} \cup V_{\mathrm{B},p}, p) \to G$ if and only if

$$(\forall v \in V_{\mathrm{L},p})(\forall (\delta(v), v', m) \in E_G)$$
$$(\delta(v), v', m) \in \delta(E_p \cap V_{\mathrm{L},p} \times V_{\mathrm{L},p} \times L_E) \cup \bigcup_{C \in V_{\mathrm{A},p}/\sim_p} \bigcup_{\mu \in M_\delta(C)} \mathrm{Old}(\mu), \quad \text{(A.2)}$$

and

$$(\forall v \in V_{\mathrm{L},p})(\forall (v', \delta(v), m) \in E_G)$$
$$(v', \delta(v), m) \in \delta(E_p \cap V_{\mathrm{L},p} \times V_{\mathrm{L},p} \times L_E) \cup \bigcup_{C \in V_{\mathrm{A},p}/\sim_p} \bigcup_{\mu \in M_\delta(C)} \mathrm{Old}(\mu), \quad \text{(A.3)}$$

where

$$M_\delta(C) = \{\mu : \mathrm{span}(V_{\mathrm{L},p} \cup V_{\mathrm{B},p} \cup C, p) \to G | \mu(\mathrm{span}(V_{\mathrm{L},p} \cup V_{\mathrm{B},p})) = \delta\}, \quad \text{(A.4)}$$

$$\mathrm{Old}(\mu) = \{(\delta(v), \mu(v'), m) \mid (v, v', m) \in E_p \cap V_{\mathrm{L},p} \times (V_{\mathrm{A},p} \cup V_{\mathrm{B},p}) \times L_E\}$$
$$\cup \{(\mu(v'), \delta(v), m) \mid (v', v, m) \in E_p \cap (V_{\mathrm{A},p} \cup V_{\mathrm{B},p}) \times V_{\mathrm{L},p} \times L_E\} \quad \text{(A.5)}$$

[1] A more general definition for monomorphisms may be found in any text on category theory, such as Pierce's (1991).

The set of arcs $\mathrm{Old}(\mu)$ contains those arcs between the nodes to be removed and the embedding environment in the host graph. The set C represents connected vertices in V_{A}. The set of edges $M_\delta(C)$ contains those arcs in the host graph that are matched to those in V_{A} according to μ—a particular (indeterminate) extension of the vertex monomorphism, δ. The edges in the set $\bigcup_{C \in V_{\mathrm{A},p}/\sim_p} \bigcup_{\mu \in M_\delta(C)} \mathrm{Old}(\mu)$ are those arcs that connect nodes to be removed with arcs in matched subgraphs in their $\sim_p$-connected environment, along with arcs in that matched subgraph. So, all host-graph arcs incident to the nodes that match vertices in V_{L}, according to δ, can be divided into

1. Arcs that match edges in V_{L}
2. Arcs that match edges (including matched target subgraphs) from V_{L} to the embedding environment
3. Arcs that do not match edges in the production

I distinguish the target subgraphs of the second group of arcs, because they are precisely those subgraphs that may need to be connected to the inserted subgraph (V_{R}) according to the production.

A.3 Effects of a Production

When a production p is applied to the host graph, there are three basic effects, executed by the derivation system:

1. It removes labeled nodes matching vertices in V_{L}.
2. It adds labeled nodes corresponding to vertices in V_{R}.
3. It adds labeled arcs among the new nodes and between the new nodes and subgraphs of the host graph that were connected to the removed nodes and matched according to a particular extension of δ, all according to the production.

Stated formally, the effect of applying production p on graph G at the subgraph indicated by $\delta : \mathrm{span}(V_{\mathrm{L},p} \cup V_{\mathrm{B},p}, p) \to G$ is the graph H:

$$\begin{aligned}
V_H &= V_G \setminus \delta(V_{\mathrm{L},p}) \cup V_{\mathrm{R},p}, &\text{(A.6)}\\
l_{V_H} &= l_{V_G}(V_G \setminus \delta(V_{\mathrm{L},p})) \cup l_{V_p}(V_{\mathrm{R},p}), &\text{(A.7)}\\
E_H &= (E_G \cap (V_G \setminus \delta(V_{\mathrm{L},p})) \times (V_G \setminus \delta(V_{\mathrm{L},p})) \times L_E) & \\
&\quad \cup (E_p \cap V_{\mathrm{R},p} \times V_{\mathrm{R},p} \times L_E) & \\
&\quad \cup (\bigcup_{C \in V_{\mathrm{A},p}/\sim_p} \bigcup_{\mu \in M_\delta(C)} \mathrm{New}(\mu)), &\text{(A.8)}
\end{aligned}$$

where

$$\begin{aligned}
M_\delta(C) &= \{\mu : \mathrm{span}(V_{\mathrm{L},p} \cup V_{\mathrm{B},p} \cup C, p) \to G | \mu(\mathrm{span}(V_{\mathrm{L},p} \cup V_{\mathrm{B},p})) = \delta\}, &\text{(A.9)}\\
\mathrm{New}(\mu) &= \{(\mu(v), v', m) | (v, v', m) \in E_P \cap (V_{\mathrm{A},p} \cup V_{\mathrm{B},p}) \times V_{\mathrm{R},p} \times L_E\} & \\
&\quad \cup \{(v', \mu(v), m) | (v', v, m) \in E_P \cap V_{\mathrm{R},p} \times (V_{\mathrm{A},p} \cup V_{\mathrm{B},p}) \times L_E\}, &\text{(A.10)}
\end{aligned}$$

and where the labeling set does not change:

$$L_{V_H} = L_V,$$

$$L_{E_H} \quad = \quad L_E.$$

The edges $\bigcup_{C \in V_{\mathrm{A},p}/\sim_p} \bigcup_{\mu \in M_\delta(C)} \mathrm{New}(\mu)$ are the arcs that connect nodes to be added with the matched subgraph in the connected environment, along with arcs in the subgraph that is matched to V_{A}.

A.4 Constructive Derivation of Graphs

To make Göttler's formalism convenient for modeling medical decisions, I have adopted four modifications:

1. When the application of a production cannot be determined from the host graph, the monomorphism and the set C are determined by the user of my derivation system.

2. The labeling function also matches nonterminal symbols (denoted by angle brackets) in a production to terminal symbols (instances) in a classification hierarchy.

3. The derivation system synthesizes variant symbols from nodes already matched in a monomorphism, and adds the new terminal symbols to the classification hierarchy.

4. Those vertices in V_{R} that match nodes that already exist in the host graph are not duplicated (Equations A.6 and A.7).

The first modification allows the user to resolve indeterminisms that may occur in a derivation when multiple entered considerations are classified under the same node-label abstraction. The second modification enables my graph grammars to express general relations among classes of concepts. An equivalent, but impractical, alternative would be to use single-node replacement rules, which, as discussed in Section 3.3, are equivalent to production rules in a context-free string grammar. The third modification allows me to use a more compact node-label hierarchy. Also, it allows for accessory variant nodes ($V_{\mathrm{R}_{\mathrm{var}},p}$) matching vertices in V_{R} to be added as needed. I partition vertices in the right region of a production into two sets: $V_{\mathrm{R}_{\mathrm{new}}}$ and $V_{\mathrm{R}_{\mathrm{var}}}$. Those vertices ($V_{\mathrm{R}_{\mathrm{var}}}$) with nonterminal labels that are marked as variant labels in the node-label classification hierarchy are matched to new nodes with labels that consist of the indicated prefix or suffix and a stem that is identical to the label string of another matched node in the production.[2] When a variant node, derived from a node already matched in the monomorphism, is identical to a node in the host graph, it is replaced by the host-graph node in that monomorphism. My last modification to Göttler's formalism correspond to my assumption that nodes are uniquely identified by their label. None of my modifications invalidate any important generic property

[2]When there are zero or more than one vertex in V_{LAB} of a sibling or parent class to a variant label in V_{R}, additional notation will be needed to resolve this ambiguity. My grammar has not yet required such extensions to the graph-grammar formalism.

of graph grammars or of graph derivations.

I define a **derivation** (Δ) to be a sequence of **partial derivation stages** (∂_i). Each partial derivation stage comprises a set of monomorphisms (δ_i) of various—not necessarily distinct—productions that are applied to the host graph in an arbitrary order. Each monomorphism must introduce at least one node with a label from the input list ($L_{V_{\text{input}}}$) to the host graph. I also require that each application in a particular stage must (1) introduce a new node to the host graph, and (2) be applicable to the host graph prior to any modifications made by other applications of that derivation stage. That is, ∂_i consists of all possible applications of productions such that each adds a different label from $L_{V_{\text{input}}}$ to the host graph, and such that the following properties hold for each application δ :

$$V_{\mathrm{R}_{\text{new}},p} \cap V_G = \emptyset, \tag{A.11}$$

$$l_{V_H}(V_{\mathrm{R}_{\text{new}},p}) \in L_{V_{\text{input}}}, \tag{A.12}$$

$$V_{\mathrm{R}_{\text{new}},p} \subseteq V_H, \tag{A.13}$$

where G is the host graph before ∂_i, and H is the host graph after ∂_i. A derivation Δ is successful if the host graph resulting from the last derivation stage contains a node corresponding to each term in the input list, $L_{V_{\text{input}}}$. A grammar is **ambiguous** if multiple graphs can be derived from the same input. Here, *input* refers to both the list of terms, $L_{V_{\text{input}}}$, and the user's responses to V_{A}-matching queries from the system.

Appendix B

Qualitative Contingent Influence Diagrams

A **qualitative contingent influence diagram** (**QCID**) is a semantic network that describes a set of possible scenarios, S. For each scenario, $s \in S$, there is a set of decisions, D_s, and a set of random variables, X, the possible values of which correspond to particular events or outcomes. There is exactly one utility variable, v, whose value the decision maker wishes to make as high as possible by choosing the optimal alternatives for the decisions in $D = \{D_s : s \in S\}$. The expected utility of a scenario is given by $U(s)$. Visually, I denote members of D by square decision nodes, the members of X by circular chance nodes, and v by a hexagonal utility node. All these nodes are part of a **directed acyclic graph.** Arcs whose target is in $X \cup \{v\}$, called **relevance arcs,** represent either probabilistic dependence, or functional dependence of the target variable on the source variable. Arcs whose target is a decision node in D, called **informational arcs,** specify exactly which outcomes and prior decisions corresponding to that decision node are known to the decision maker at the time of the decision. Informational arcs imply a temporal ordering: The information is known before the decision is made. Since decision analysis generally requires that there be a total ordering of decisions in each scenario, and that the decision maker does not forget information known at the time of a previous decision, I can abbreviate the diagram by omitting informational arcs from a node to all but the earliest of those decisions that have information about that outcome.

Relevance arcs are labeled with a symbol from $L_E = \{$"+", "−", "0", "?"$\}$. For binary (true-or-false) nodes A and B, a plus sign, +, on the arc from A to B indicates that $P(B|A,\xi) \geq P(B|\bar{A},\xi)$. For nonbinary nodes whose random variables have any one of ordinal, interval, or ratio values for their possible outcomes, a plus sign, +, indicates that an upward shift in the mean for B's probability distribution, with all other relevant variables held constant, results in an upward shift in the mean for A's distribution. Similarly, a minus sign, −, on the arc from A to B indicates that $P(B|A,\xi) \leq P(B|\bar{A},\xi)$ for binary nodes, and that there is an oppositely directed shift in their variable's distribution for nonbinary nodes. A zero, 0, indicates probabilistic independence. Such arcs are generally omitted. A question mark, ?, indicates that the direction of such qualitative dependency is nonmonotonic or unclear. Informational arcs should not be labeled in this manner, as the specification of which alternative is optimal should be contained elsewhere in the assessed influence-diagram model.

Contingent nodes[1] play a role only when their explicit conditions on previous outcomes have been met. Following Fung and Shachter's notation, I shall use Ω_j to refer to the joint state space (i.e., all combinations of outcomes) of the variables in the set of nodes $J = \{j_1, j_2, \ldots, j_n\}$:

$$\Omega_j = (\Omega_{j_1} \times \Omega_{j_2} \times \ldots \times \Omega_{j_n}).$$

The direct successors of a node i are those nodes that are targets of arcs originating at node i. The direct predecessors of a node i are those nodes that are sources of arcs terminating at node i. The ancestors of a node i are all nodes that occur prior to i along a path in the directed acyclic graph.

For each node i, there is a joint state space (Ω_K) for all possible outcomes of its ancestors. This state space can be partitioned into component contingency spaces ($\Omega_\kappa = \{\Omega_{1,\kappa}, \Omega_{2,\kappa}, \ldots, \Omega_{n,\kappa}\}$), each of which is defined by a set of conditions. A condition ($\Omega_{i,\kappa}$) for node i is a statement about the outcome of some subset of the ancestors of i. In a contingent influence diagram, more that one node with the same label—representing the same decision or random variable—can be drawn; however,

[1]Robert M. Fung and Ross D. Shachter. Personal communication.

these identically labeled nodes must be associated with mutually exclusive contingency spaces. A contingent node may have one or more contingencies, κ, represented visually by one or more dots on its left side.

Appendix C

User-Assistance Dialogue

(The following dialogue between Gramarye and the user occurred during the derivation described in Section 5.2, and depicted in Figure 5.5. Gramarye's output is listed in sans serif font. The user's response is printed in bold type.)

<cl> **(derive '(cad cvd aaa cad-hx cvd-hx aaa-width repair-aaa cabg endarterectomy cardiac-catheterization carotid-arteriography mi stroke aaa-rupture anaesth-complics pt-morbidity pt-mortality))**

Remake *label-table*? **y**

Rebuilding the hash table...done.

term-list: (CAD CVD AAA CAD-HX CVD-HX AAA-WIDTH REPAIR-AAA CABG ENDARTERECTOMY CARDIAC-CATHETERIZATION CAROTID-ARTERIOGRAPHY MI STROKE AAA-RUPTURE ANAESTH-COMPLICS PT-MORBIDITY

PT-MORTALITY)

term-list: (PT-MORTALITY PT-MORBIDITY ANAESTH-COMPLICS AAA-RUPTURE STROKE MI CAROTID-ARTERIOGRAPHY CARDIAC-CATHETERIZATION ENDARTERECTOMY CABG REPAIR-AAA AAA-WIDTH CVD-HX CAD-HX AAA CVD)

term-list: (CVD AAA CAD-HX CVD-HX AAA-WIDTH REPAIR-AAA CABG ENDARTERECTOMY CARDIAC-CATHETERIZATION CAROTID-ARTERIOGRAPHY MI STROKE AAA-RUPTURE ANAESTH-COMPLICS PT-MORBIDITY)

term-list: (PT-MORBIDITY ANAESTH-COMPLICS AAA-RUPTURE STROKE MI CAROTID-ARTERIOGRAPHY CARDIAC-CATHETERIZATION ENDARTERECTOMY CABG REPAIR-AAA AAA-WIDTH CVD-HX CAD-HX AAA)

term-list: (AAA CAD-HX CVD-HX AAA-WIDTH REPAIR-AAA CABG ENDARTERECTOMY CARDIAC-CATHETERIZATION CAROTID-ARTERIOGRAPHY MI STROKE AAA-RUPTURE ANAESTH-COMPLICS)

term-list: (ANAESTH-COMPLICS AAA-RUPTURE STROKE MI CAROTID-ARTERIOGRAPHY CARDIAC-CATHETERIZATION ENDARTERECTOMY CABG REPAIR-AAA AAA-WIDTH CVD-HX CAD-HX)

... while trying to add CAD-HX ...

The following labels could map to PRESENT-DISEASE:

(AAA CVD CAD)

Should AAA fill the PRESENT-DISEASE role?**n**

Should CVD fill the PRESENT-DISEASE role?**n**

Should CAD fill the PRESENT-DISEASE role?**y**

term-list: (CAD-HX CVD-HX AAA-WIDTH REPAIR-AAA CABG ENDARTERECTOMY CARDIAC-CATHETERIZATION CAROTID-ARTERIOGRAPHY MI STROKE AAA-RUPTURE)

... while trying to add AAA-RUPTURE ...

The following labels could map to DISEASE:

(AAA CVD CAD)

Should AAA fill the DISEASE role?**y**

term-list: (AAA-RUPTURE STROKE MI CAROTID-ARTERIOGRAPHY CARDIAC-CATHETERIZATION ENDARTERECTOMY CABG REPAIR-AAA AAA-WIDTH CVD-HX)

... while trying to add CVD-HX ...

The following labels could map to PRESENT-DISEASE:

(AAA CVD CAD)

Should AAA fill the PRESENT-DISEASE role?**n**

Should CVD fill the PRESENT-DISEASE role?**y**

term-list: (CVD-HX AAA-WIDTH REPAIR-AAA CABG ENDARTERECTOMY CARDIAC-CATHETERIZATION CAROTID-ARTERIOGRAPHY MI STROKE)

... while trying to add STROKE ...

The following labels could map to DISEASE:

(AAA CVD CAD)

Should AAA fill the DISEASE role?**n**

Should CVD fill the DISEASE role?**y**

term-list: (STROKE MI CAROTID-ARTERIOGRAPHY CARDIAC-CATHETERIZATION

ENDARTERECTOMY CABG REPAIR-AAA AAA-WIDTH)

... while trying to add AAA-WIDTH ...

The following labels could map to PRESENT-DISEASE:

(AAA CVD CAD)

Should AAA fill the PRESENT-DISEASE role?**y**

term-list: (AAA-WIDTH REPAIR-AAA CABG ENDARTERECTOMY

CARDIAC-CATHETERIZATION CAROTID-ARTERIOGRAPHY MI)

... while trying to add MI ...

The following labels could map to DISEASE:

(AAA CVD CAD)

Should AAA fill the DISEASE role?**n**

Should CVD fill the DISEASE role?**n**

Should CAD fill the DISEASE role?**y**

term-list: (MI CAROTID-ARTERIOGRAPHY CARDIAC-CATHETERIZATION

ENDARTERECTOMY CABG REPAIR-AAA)

... while trying to add REPAIR-AAA ...

The following labels could map to PRESENT-DISEASE:

(AAA CVD CAD)

Should AAA fill the PRESENT-DISEASE role?**y**

term-list: (REPAIR-AAA CABG ENDARTERECTOMY CARDIAC-CATHETERIZATION

CAROTID-ARTERIOGRAPHY)

... while trying to add CAROTID-ARTERIOGRAPHY ...

The following labels could map to PRESENT-DISEASE:

(AAA CVD CAD)

Should AAA fill the PRESENT-DISEASE role?**n**

Should CVD fill the PRESENT-DISEASE role?**y**

term-list: (CAROTID-ARTERIOGRAPHY CARDIAC-CATHETERIZATION

ENDARTERECTOMY CABG)

... while trying to add CABG ...

The following labels could map to PRESENT-DISEASE:

(AAA CVD CAD)

Should AAA fill the PRESENT-DISEASE role?**n**

Should CVD fill the PRESENT-DISEASE role?**n**

Should CAD fill the PRESENT-DISEASE role?**y** term-list: (CABG ENDARTERECTOMY CARDIAC-CATHETERIZATION)

... while trying to add CARDIAC-CATHETERIZATION ...

The following labels could map to TX:

(CABG REPAIR-AAA)

Should CABG fill the TX role?**y**

... while trying to add CARDIAC-CATHETERIZATION ...

The following labels could map to PRESENT-DISEASE:

(AAA CVD CAD)

Should AAA fill the PRESENT-DISEASE role?**n**

Should CVD fill the PRESENT-DISEASE role?**n**

Should CAD fill the PRESENT-DISEASE role?**y**

term-list: (CARDIAC-CATHETERIZATION ENDARTERECTOMY)

... while trying to add ENDARTERECTOMY ...

The following labels could map to PRESENT-DISEASE:

(AAA CVD CAD)

Should AAA fill the PRESENT-DISEASE role?**n**

Should CVD fill the PRESENT-DISEASE role?**y**

term-list: (ENDARTERECTOMY)

!graph=((ENDARTERECTOMY FUTURE-CVD

RESULT-OF-CARDIAC-CATHETERIZATION CARDIAC-CATHETERIZATION CABG

FUTURE-CAD RESULT-OF-CAROTID-ARTERIOGRAPHY CAROTID-ARTERIOGRAPHY REPAIR-AAA FUTURE-AAA MI AAA-WIDTH STROKE CVD-HX AAA-RUPTURE CAD-HX ANAESTH-COMPLICS AAA PT-MORBIDITY CVD PT-MORTALITY CAD VALUE-TO-PATIENT) ((CVD '+ FUTURE-CVD) (ENDARTERECTOMY '- VALUE-TO-PATIENT) (FUTURE-CVD '- VALUE-TO-PATIENT) (ENDARTERECTOMY '- FUTURE-CVD) (CAD '+ RESULT-OF-CARDIAC-CATHETERIZATION) (CARDIAC-CATHETERIZATION 'C RESULT-OF-CARDIAC-CATHETERIZATION) (RESULT-OF-CARDIAC-CATHETERIZATION 'I CABG) (CARDIAC-CATHETERIZATION 'I CABG) (CARDIAC-CATHETERIZATION '- VALUE-TO-PATIENT) (CAD '+ FUTURE-CAD) (CABG '- VALUE-TO-PATIENT) (FUTURE-CAD '- VALUE-TO-PATIENT) (CABG '- FUTURE-CAD) (CVD '+ RESULT-OF-CAROTID-ARTERIOGRAPHY) (CAROTID-ARTERIOGRAPHY 'C RESULT-OF-CAROTID-ARTERIOGRAPHY) (RESULT-OF-CAROTID-ARTERIOGRAPHY 'I REPAIR-AAA) (CAROTID-ARTERIOGRAPHY 'I REPAIR-AAA) (CAROTID-ARTERIOGRAPHY '- VALUE-TO-PATIENT) (AAA '+ FUTURE-AAA) (REPAIR-AAA '- VALUE-TO-PATIENT) (FUTURE-AAA '- VALUE-TO-PATIENT) (REPAIR-AAA '- FUTURE-AAA) (MI '- VALUE-TO-PATIENT) (CAD '+ MI) (AAA '+ AAA-WIDTH) (STROKE '- VALUE-TO-PATIENT) (CVD '+ STROKE) (CVD '+ CVD-HX)

(AAA-RUPTURE '- VALUE-TO-PATIENT) (AAA '+ AAA-RUPTURE) (CAD '+ CAD-HX) (ANAESTH-COMPLICS '- VALUE-TO-PATIENT) (AAA '- VALUE-TO-PATIENT) (PT-MORBIDITY '- VALUE-TO-PATIENT) (CVD '- VALUE-TO-PATIENT) (PT-MORTALITY '- VALUE-TO-PATIENT) (CAD '- VALUE-TO-PATIENT)))

NIL

<cl>

Bibliography

Andersen, S. K., Olesen, K. G., Jensen, F. V., and Jensen, F. (1989). HUGIN: A shell for building Bayesian belief universes for expert systems. In Sridharan, N. S., editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1080–1085, Detroit, MI. Morgan Kaufmann.

Applied Decision Analysis (1992). DPL: Decision programming language. Technical report, Applied Decision Analysis, Inc., Menlo Park, CA.

Barthelmann, K. (1990). Describing Göttler's operational graph grammars with pushouts. In Ehrig, H., Kreowski, H.-J., and Rozenberg, G., editors, *Graph Grammars and Their Application to Computer Science: Proceedings of the Fourth International Workshop*, pages 98–112, Bremen, Germany. Springer-Verlag.

Beinlich, I. A. and Herskovits, E. H. (1990). Ergo: A graphical environment for constructing bayesian belief networks. In Bonissone, P. and Henrion, M., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Sixth Conference*, pages 276–283, Cambridge, MA. North-Holland.

Beinlich, I. A., Suermondt, H. J., Chavez, R. M., and Cooper, G. F. (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In Hunter, J., Cookson, J., and Wyatt, J., editors, *Proceedings of the Sixth Conference on Artificial Intelligence in Medical Care*, pages 247–256. Springer-Verlag, London.

Bernauer, J. (1991). Conceptual graphs as an operational model for descriptive findings. In Clayton, P. D., editor, *Proceedings of the Fifteenth Symposium on Computer Applications in Medical Care*, pages 214–218, Washington, D.C. McGraw-Hill, Inc.

Bouckaert, R. R. (1992). Optimizing causal orderings for generating DAGs from data. In Dubois, D., Wellman, M. P., D'Ambrosio, B., and Smets, P., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Eighth Conference*, pages 9–16, Stanford, CA. Morgan Kaufmann.

Bradshaw, J. M., Covington, S. P., Russo, P. J., and Boose, J. H. (1991). Knowledge acquisition techniques for decision analysis using AXOTL and AQUINAS. *Knowledge Acquisition*, 3:49–77.

Breese, J. S. (1992). Construction of belief and decision networks. *Computational Intelligence*, 8:624–647.

Bunke, H. (1982). On the generative power of sequential and parallel programmed graph grammars. *Computing*, 29:89–112.

Call, H. J. and Miller, W. A. (1990). A comparison of approaches and implementations for automating decision analysis. *Reliability Engineering and System Safety*, 30:115–162.

Campbell, K. E., Das, A. K., and Musen, M. A. (1994). A logical foundation for representation of clinical data. Technical Report KSL-94-02, Knowledge Systems Laboratory, Stanford University, Stanford, CA.

Carbonell, J. G. (1983). Learning by analogy: Formulating and generalizing plans from past experience. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning*, pages 137–162. Morgan Kaufmann, Los Altos, CA.

Chavez, R. M. and Cooper, G. F. (1990). A randomized approximation for probabilistic inference on bayesian belief networks. *Networks*, 20:661–685.

Cooper, G. F. (1988). A method for using belief networks as influence diagrams. In Shachter, R. D., Levitt, T. S., Kanal, L. N., and Lemmer, J. F., editors, *Proceedings of the AAAI Workshop on Uncertainty in Artificial Intelligence*, pages 55–63, Minneapolis, MN.

Corradini, A., Montanari, U., Rossi, F., Ehrig, H., and Lowe, M. (1990). Graph grammars and logic programming. In Ehrig, H., Kreowski, H.-J., and Rozenberg, G., editors, *Graph Grammars and Their Application to Computer Science: Proceedings of the Fourth International Workshop*, pages 221–237, Bremen, Germany. Springer-Verlag.

Côté, R., Rothwell, D., Palotay, J., Beckett, R., and Brochu, L., editors (1993). *The Systematized Nomenclature of Medicine: SNOMED International*. College of American Pathologists, Northfield, IL.

Cousins, S. B., Chen, W., and Frisse, M. E. (1992). CABeN: A collection of algorithms for belief networks. Technical Report WUCS-91-25, Medical Informatics Laboratory, Washington University, St. Louis, MO.

Dagum, P. and Galper, A. (1993). Additive belief-network models. In Heckerman, D. and Mamdani, A., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 91–98, Washington, D.C. Morgan Kaufmann.

D'Ambrosio, B. (1989). Symbolic probabilistic inference. Technical report, Computer Science Department, Oregon Statae University.

Davis, R. (1984). Interactive transfer of expertise. In Buchanan, B. G. and Shortliffe, E. H., editors, *Rule-Based Expert Systems: The MYCIN Experiments of the*

Stanford Heuristic Programming Project, chapter 9, pages 171–205. Addison-Wesley, Reading, MA.

Díez, F. J. (1993). Parameter adjustment in Bayes networks. the generalized noisy-OR gate. In Heckerman, D. and Mamdani, A., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 99–105, Washington, D.C. Morgan Kaufmann.

Ehrig, H. and Kreowski, H.-J. (1980). Applications of graph grammar theory to consistency, synchronization, and scheduling in data base systems. *Information Systems*, 5:225–238.

Ehrig, H., Kreowski, H.-J., and Rozenberg, G., editors (1990). *Graph Grammars and Their Application to Computer Science: Proceedings of the Fourth International Workshop*, Bremen, Germany. Springer-Verlag.

Ehrig, H., Rozenberg, G., and Rosenfeld, A., editors (1986). *Graph-Grammars and Their Application to Computer Science: Proceedings of the Third International Workshop*, Warrenton, VA. Springer-Verlag.

Engels, G., Lewerentz, C., and Schäfer, W. (1986). Graph grammar engineering: A software specification method. In Ehrig, H., Rozenberg, G., and Rosenfeld, A., editors, *Graph Grammars and Their Application to Computer Science: Proceedings of the Third International Workshop*, pages 186–201, Warrenton, VA. Springer-Verlag.

Farr, B. R. and Shachter, R. D. (1992). Representation of preferences in decision-support systems. *Computers and Biomedical Research*, 25:324–335.

Feder, J. (1971). Plex languages. *Information Science*, 3:225–241.

Finkel, A. J., editor (1990). *Physicians' Current Procedural Terminology.* American Medical Association Press, Chicago, IL, 4 edition.

Friedman, C., Cimino, J. J., and Johnson, S. B. (1993). A conceptual model for clinical radiology reports. In Safran, C., editor, *Proceedings of the Seventeenth Symposium on Computer Applications in Medical Care*, pages 829–833, Washington, D.C. McGraw-Hill, Inc.

Fung, R. M. and Shachter, R. D. (1990). Contingent influence diagrams. Technical report, Engineering-Economic Systems Department, Stanford University.

Geiger, D. (1992). An entropy-based learning algorithm of bayesian conditional trees. In Dubois, D., Wellman, M. P., D'Ambrosio, B., and Smets, P., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Eighth Conference*, pages 92–97, Stanford, CA. Morgan Kaufmann.

Goldman, R. P. and Breese, J. S. (1992). Integrating model construction and evaluation. In Dubois, D., Wellman, M. P., D'Ambrosio, B., and Smets, P., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Eighth Conference*, pages 92–97, Stanford, CA. Morgan Kaufmann.

Goldman, R. P. and Charniak, E. (1990). Dynamic construction of belief networks. In Bonissone, P. and Henrion, M., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Sixth Conference*, pages 90–97, Cambridge, MA. North-Holland.

Göttler, H. (1989). Graph grammars, a new paradigm for implementing visual languages. In Dershowitz, N., editor, *Rewriting Techniques and Applications: Proceedings of the Third International Conference*, pages 152–166, Berlin, Germany. Springer-Verlag.

Göttler, H. (1990). Usefulness of graph grammars in applications. In Ehrig, H., Kreowski, H.-J., and Rozenberg, G., editors, *Graph Grammars and Their Application to Computer Science: Proceedings of the Fourth International Workshop*, pages 48–49, Bremen, Germany. Springer-Verlag.

Göttler, H. (1992). Diagram editors = graphs + attributes + graph grammars. *International Journal of Man–Machine Studies*, 37:481–502.

Göttler, H., Günther, J., and Nieskens, G. (1990). Use of graph grammars to design CAD-systems! In Ehrig, H., Kreowski, H.-J., and Rozenberg, G., editors, *Graph Grammars and Their Application to Computer Science: Proceedings of the Fourth International Workshop*, pages 396–410, Bremen, Germany. Springer-Verlag.

Greenes, R. A., McClure, R. C., Pattison-Gordon, E., and Sato, L. (1992). The findings–diagnosis continuum: Implications for image descriptions and clinical databases. In Frisse, M. E., editor, *Proceedings of the Sixteenth Symposium on Computer Applications in Medical Care*, pages 383–387, Baltimore, MD. McGraw-Hill, Inc.

Habel, A., editor (1992). *Hyperedge Replacement: Grammars and Languages.* Springer-Verlag, Berlin.

Heckerman, D. E. (1991). *Probabilistic Similarity Networks.* M.I.T. Press, Cambridge, MA.

Heckerman, D. E. (1993). Causal independence for knowledge acquisition and inference. In Heckerman, D. and Mamdani, A., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 122–127, Washington, D.C. Morgan Kaufmann.

Heckerman, D. E. and Horvitz, E. J. (1990). Problem formulation as the reduction of a decision model. In Bonissone, P. and Henrion, M., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Sixth Conference*, pages 82–89, Cambridge, MA. North-Holland.

Heckerman, D. E. and Nathwani, B. N. (1992). Toward normative expert systems: Part ii. probability-based representations for efficient knowledge acquisition and inference. *Methods of Information in Medicine*, 31:106–116.

Henrion, M. (1992). Demos release notes, Macintosh version 2.0b2. Technical report, Lumina Decision Systems, Inc., Palo Alto, CA.

Herskovits, E. H. and Cooper, G. F. (1990). Kutató: An entropy-driven system for construction of probabilistic expert systems from databases. In Bonissone, P. and Henrion, M., editors, *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 54–62, Cambridge, MA. North-Holland.

Højsgaard, S. and Thiesson, B. (1992). BIFROST—block recursive models induced from relevant knowledge, observations, and statistical techniques. Technical Report R 92-2010, Institute for Electronic Systems, Aalborg University, Aalborg, Denmark.

Hollenberg, J. P. (1984). The decision-tree builder: An expert system to simulate medical prognosis and management. *Medical Decision Making*, 4:531.

Holtzman, S. (1988). *Intelligent Decision Systems.* Addison-Wesley, Reading, MA.

Horsch, M. C. and Poole, D. (1990). A dynamic approach to probabilistic inference using bayesian networks. In Bonissone, P. and Henrion, M., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Sixth Conference*, pages 155–161, Cambridge, MA. North-Holland.

Howard, R. A. (1984a). Information value theory. In Howard, R. A. and Matheson, J. E., editors, *The Principles and Applications of Decision Analysis*, pages 779–783. Strategic Decisions Group, Menlo Park, CA.

Howard, R. A. (1984b). Proximal decision analysis. In Howard, R. A. and Matheson, J. E., editors, *The Principles and Applications of Decision Analysis*, pages 845–879. Strategic Decisions Group, Menlo Park, CA.

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*, pages 719–762. Strategic Decisions Group, Menlo Park, CA.

Jensen, F. V., Lauritzen, S. L., and Olesen, K. G. (1990). Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282.

Jimison, H. B. (1990). *A Representation for Gaining Insight into Clinical Decision Models.* PhD thesis, Stanford University, Stanford, CA.

Jones, C. V. (1990). An introduction to graph-based modeling systems, part i: Overview. *ORSA Journal on Computing*, 2:136–151.

Jones, C. V. (1991). An introduction to graph-based modeling systems, part ii: Graph-grammars and the implementation. *ORSA Journal on Computing*, 3:180–206.

Langlotz, C. P., Fagan, L. M., Tu, S. W., Sikic, B. I., and Shortliffe, E. H. (1987). A therapy planning architecture that combines decision theory and artificial intelligence techniques. *Computers and Biomedical Research*, 20:279–303.

Langlotz, C. P., Shortliffe, E. H., and Fagan, L. M. (1988). A methodology for generating computer–based explanations of decision–theoretic advice. *Medical Decision Making*, 8:290–303.

Laskey, K. B. (1990). A probabilistic reasoning environment. In Bonissone, P. and Henrion, M., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Sixth Conference*, pages 415–422, Cambridge, MA. North-Holland.

Laskey, K. B. (1993). Sensitivity analysis for probability assessments in bayesian networks. In Heckerman, D. and Mamdani, A., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 136–142, Washington, D.C. Morgan Kaufmann.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statist. Soc.*, 50:157–224.

Leong, T.-Y. (1991). Knowledge representation for supporting decision model formulation in medicine. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Leong, T.-Y. (1992). Representing context-sensitive knowledge in a network formalism: A preliminary report. In Dubois, D., Wellman, M. P., D'Ambrosio, B. D., and Smets, P., editors, *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 166–173, Stanford, CA. Morgan Kaufmann.

McCray, A. T. and Hole, W. T. (1990). The scope and structure of the first version of the UMLS semantic network. In Miller, R. A., editor, *Proceedings of the Fourteenth Symposium on Computer Applications in Medical Care*, pages 126–130, Washington, D.C. IEEE Computer Society Press.

Miller, R. A., McNeil, M. A., Challinor, S. M., Masarie, F. E., and Myers, J. D. (1986). The INTERNIST-1/Quick Medical Reference project: Status report. *The Western Journal of Medicine*, 145:824–832.

Montanari, U. and Rossi, F. (1986). An efficient algorithm for the solution of hierarchical networks of constraints. In Ehrig, H., Rozenberg, G., and Rosenfeld, A., editors, *Graph-Grammars and Their Application to Computer Science: Proceedings of the Third International Workshop*, pages 440–457, Warrenton, VA. Springer-Verlag.

Musen, M. A. (1992). Dimensions of knowledge sharing and reuse. *Computers and Biomedical Research*, 25:435–467.

Nagl, M. (1976). Formal languages of labelled graphs. *Computing*, 16:113–137.

Nolan, W., Rector, A., Kay, S., Horan, B., and Wilson, A. (1991). A patient-care workstation based on user-centered design and a formal theory of medical terminology: PEN&PAD and the SMK formalism. In Clayton, P. D., editor, *Proceedings of the Fifteenth Symposium on Computer Applications in Medical Care*, pages 855–857, Washington, D.C. McGraw-Hill, Inc.

North, D. W. (1968). A tutorial introduction to decision theory. *IEEE Transactions on Systems, Science and Cybernetics*, 4:200–210.

Olesen, K. J., Kjaelrulff, U., Jensen, F., Jensen, F. V., Falck, B., Andreassen, S., and Andersen, S. K. (1989). A MUNIN network for the median nerve. *Applied Artificial Intelligence*, 3:385–404.

Pavlidis, T. (1972). Linear and context-free graph grammars. *Journal of the ACM*, 19:11–23.

Pearl, J. (1986). Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29:241–288.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Mateo, CA.

Pierce, B. C. (1991). *Basic Category Theory for Computer Scientists.* MIT Press, Cambridge, MA.

Polya, G. (1945). *How to Solve It.* Princeton University Press, Princeton, NJ.

Prieto-Díaz, R. (1993). Status report: Software reusability. *IEEE Software*, 10(3):61–66.

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*, 15:299–307.

Robinson, R. W. (1976). Counting unlabeled acyclic digraphs. In Little, C., editor, *Proceedings of the Fifth Australian Conference on Combinatorial Mathematics*, pages 28–43, Melbourne, Australia. Springer-Verlag.

Rothwell, D., Côté, R., J.P.Cordeau, and Boisvert, M. (1993). Developing a standard data structure for medical language: The SNOMED proposal. In Safran, C., editor, *Proceedings of the Seventeenth Symposium on Computer Applications in Medical Care*, pages 695–699, Washington, D.C. McGraw-Hill, Inc.

Rothwell, D. J. and Côté, R. A. (1990). Optimizing the structure of a standardized vocabulary: The SNOMED model. In Miller, R. A., editor, *Proceedings of the Fourteenth Symposium on Computer Applications in Medical Care*, pages 181–184, Washington, D.C. IEEE Computer Society Press.

Rousseau, W. F. (1968). Method for computing probabilities in complex situations. Technical Report SEL-68-050, Department of Electrical Engineering, Stanford University, Stanford, CA.

Schocken, S. and Jones, C. V. (1993). Reframing decision problems: A graph grammar approach. *Information Systems Research*, 4:55–87.

Shachter, R. D. (1986). Evaluating influence diagrams. *Operations Research*, 34:871–882.

Shachter, R. D. and Bertrand, L. J. (1987). DAVID influence diagram processing system for the Macintosh. Technical report, Duke University, Center for Academic Computing, Durham, NC.

Shachter, R. D. and Peot, M. A. (1992). Decision making using probabilistic inference methods. In Dubois, D., Wellman, M. P., D'Ambrosio, B., and Smets, P., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Eighth Conference*, pages 276–283, Stanford, CA. Morgan Kaufmann.

Shwe, M. A., Heckerman, D. E., Henrion, M., Lehmann, H. P., and Cooper, G. F. (1991). Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base: I. The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30:241–255.

Smith, J. E., Holtzman, S., and Matheson, J. E. (1993). Structuring conditional relationships in influence diagrams. *Operations Research*, 41:280–297.

Sowa, J. F., editor (1984). *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading, MA.

Srinivas, S. (1993). A generalization of the noisy-or model. In Heckerman, D. and Mamdani, A., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 208–215, Washington, D.C. Morgan Kaufmann.

Srinivas, S. and Breese, J. S. (1990). IDEAL: A software package for analysis of influence diagrams. In Bonissone, P. and Henrion, M., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Sixth Conference*, pages 212–219, Cambridge, MA. North-Holland.

Suermondt, H. J. (1991). *Explanation of Probabilistic Inference in Bayesian Belief Networks*. PhD thesis, Stanford University, Stanford, CA.

Suermondt, H. J. and Cooper, G. F. (1992). An evaluation of explanations of probabilistic inference. In Frisse, M. E., editor, *Proceedings of the Sixteenth Symposium on Computer Applications in Medical Care*, pages 579–585, Baltimore, MD. McGraw-Hill, Inc.

Tuttle, M. S., Sperzel, W. D., Olson, N. E., Erlbaum, M. S., Suarez-Munist, O., Sheretz, D. D., Nelson, S. J., and Fuller, L. F. (1992). The homogenization of the Metathesauraus schema and distribution format. In Frisse, M. E., editor, *Proceedings of the Sixteenth Symposium on Computer Applications in Medical Care*, pages 299–303, Baltimore, MD. McGraw-Hill, Inc.

van Heijst, G., Terpstra, P., Wielinga, B., and Shadbolt, N. (1993). Using generalized directive models in knowledge acquisition. In Davis, J.-M., Krivine, J.-P., and Simmons, R., editors, *Second-Generation Expert Systems*. Springer Verlag, Berlin.

Volot, F., Zweigenbaum, P., Bachimont, B., Ben Said, M., Bouaud, J., Fieschi, M., and Boisvieux, J. (1993). Structuration and acquisition of medical knowledge using UMLS in the conceptual graph formalism. In Safran, C., editor, *Proceedings of the Seventeenth Symposium on Computer Applications in Medical Care*, pages 710–714, Washington, D.C. McGraw-Hill, Inc.

von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ.

Warner, H. R., Toronto, A. F., and Veasy, L. G. (1964). Experience with Bayes' theorem for computer diagnosis of congenital heart disease. *Annals of the New York Academy of Science*, 115:558–567.

Wellman, M. P. (1990a). *Formulation of Tradeoffs in Planning Under Uncertainty*. Morgan Kaufmann, San Mateo, CA.

Wellman, M. P. (1990b). Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44:257–303.

Wellman, M. P., Breese, J. S., and Goldman, R. P. (1992). From knowledge bases to decision models. *The Knowledge Engineering Review*, 7:35–53.

Wellman, M. P., Eckman, M. H., Fleming, C., Marshall, S. L., Sonnenberg, F. A., and Pauker, S. G. (1989). Automated critiquing of medical decision trees. *Medical Decision Making*, 9:272–284.

Zhang, N. L., Qi, R., and Poole, D. (1993). A computational theory of decision networks. Technical Report 93-6, Department of Computer Science, University of British Columbia, Vancouver, B.C.