Formulation of Tradeoffs in Planning Under Uncertainty

by

Michael Paul Wellman

S.B., Massachusetts Institute of Technology (1983)
S.M., Massachusetts Institute of Technology (1985)

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Philosophy

at the

Massachusetts Institute of Technology

September 1988

Signature of Author____
Department of Electrical Engineering and Computer Science
July 29, 1988

Certified by_____ [illegible]
Peter Szolovits
Associate Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by___ [illegible]
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Formulation of Tradeoffs in Planning Under Uncertainty

by

Michael Paul Wellman

Submitted to the Department of Electrical Engineering and Computer Science
on July 29, 1988
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

The *tradeoff formulation* task is to identify the central issues in a decision problem by recognizing strategies that are inadmissible on qualitative grounds. SUDO-PLANNER formulates tradeoffs for an example medical decision problem by proving decision-theoretically that certain plan classes are dominated based on a knowledge base of qualitative relations in the domain. The classical planning representation of goals as predicates is inadequate for choice among plans that may achieve objectives in part or with uncertainty. SUDO-PLANNER's *dominance-proving architecture* is a general framework for planning for partially satisfiable goals. Dominance-proving planners characterize the space of admissible plans by maintaining a specialization graph of plan classes annotated with dominance conditions derived from a domain model.

Qualitative Probabilistic Networks (QPNs) are decision models expressing constraints on the joint probability distribution over a set of variables. *Qualitative influences* describe the direction of the relationship between two variables. *Qualitative synergies* describe interactions among influences. The probabilistic definitions of these constraints justify sound and efficient inference procedures based on graphical manipulations of the network. SUDO-PLANNER's dominance prover uses these procedures to establish dominance relations among plan classes. SUDO-PLANNER constructs decision models (QPNs) from a multilevel knowledge base describing the effects of actions and relations among events. The planning process alternates between model construction and dominance proving, producing a plan graph with dominance conditions ruling out the inadmissible therapy strategies for its medical decision example.

Thesis Supervisor: Peter Szolovits

Title: Associate Professor of Computer Science and Engineering

Acknowledgments

The contributions of my thesis committee members to this work went far beyond committee duties. Peter Szolovits, my career supervisor, has provided needed encouragement from day one. His solid thinking helped keep me on course (for the most part). Steve Pauker opened to me the world of medical decision problems, and served as the paragon of decision-analytic thinking. Ramesh Patil has been ever-willing to exchange embryonic ideas about AI and medical decision making.

The Clinical Decision Making Group at MIT has been a source of friendship as well as colleagueship. Thanks to all those who made it such a pleasurable and productive working environment. I am specifically thankful for comments on thesis drafts supplied by Tze-Yun Leong, Oksana Senyk, and Ira Haimowitz. The heroic draft-reading efforts and substantive suggestions of Elisha Sacks and Jon Doyle were especially valuable; I hereby declare them Honorary Members of my thesis committee. Thanks also to Tom Russ for assorted LaTeXhackery.

I learned a great deal from five years of Fellows at the New England Medical Center's Division of Clinical Decision Making. Collaborations with Jim Hollenberg, Alan Moskowitz, Mark Eckman, Frank Sonnenberg, and Craig Fleming were especially profitable.

This research has also benefited from discussions with Jack Breese, Tony Cox, Al Drake, Ron Rivest, Ross Shachter, the Stanford Medical probabilists: Greg Cooper, David Heckerman, Eric Horvitz, and Curt Langlotz, and others I have forgotten to mention.

Finally, thanks to Kate for love and support, which are more important than even editing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Tradeoff Formulation

A typical decision whether to perform surgery on a patient involves a tradeoff. The operation might alleviate the patient's disease, but it also carries a risk of mortality or other undesirable outcomes. Before resolving this dilemma, we might perform a diagnostic test to assess the surgery's potential effectiveness. In contrast to the original decision, determining how the result of the test should influence our surgical policy presents no tradeoff. It is obvious that our willingness to operate should increase as the test indicates surgery is more likely to be effective.

Most of the choice problems we recognize as "decisions" are tradeoff situations. Strategies such as "operate only if the test suggests surgery will be ineffective" never occur to us because they violate common sense.[1] Unfortunately, we cannot rely on decision-making computer programs to limit themselves automatically to only the sensible options. Given a rich enough set of actions and limited only by syntax, a planning program is free to assemble strategies that are arbitrarily ridiculous.

The task of *tradeoff formulation* is to separate the "real decisions" from the trivial

[1]According to Minsky [92, Chapter 27], the humor of patently irrational policies such as this is evidence of subconscious censors on nonsensical ideas.

choices. Plans that can be ruled out on simple, uncontroversial grounds are called *inadmissible* and are unworthy of serious consideration. Tradeoffs indicate potential controversy, therefore plans involved in a tradeoff should be considered admissible pending further investigation.

Once the tradeoffs are identified, other decision-making mechanisms must be employed to *resolve* them. Even when powerful decision methods are available, however, there are several reasons to accord tradeoff formulation special status in the planning architecture.

Epistemology. The knowledge required to formulate tradeoffs is more abstract, robust, and modular than that needed for tradeoff resolution or decision making in general. The situation above is described in terms of highly general concepts like surgical mortality and effectiveness. In contrast, to resolve the tradeoff we would need to specify the particular disease and surgery under consideration. For tradeoff formulation, it is possible to reason at abstract levels because the conclusions are valid in general situations.

Decision making generally requires precise assessments of such magnitudes as the relative desirability of living with the disease compared to death. The qualitative assertions needed for tradeoff formulation, for example, "more severe disease is less desirable," typically hold with much greater confidence.

Qualitative assertions hold in a wide variety of contexts—diseases are bad for patients of all ages, sizes, and shapes. The magnitude assessments, in contrast, often depend on these and other known patient features. The context-insensitivity of qualitative properties translates to significant modularity advantages in representing tradeoff formulation knowledge.

Explanation. A program strengthens its justifications by relying on the weakest premises. An explanation of a non-tradeoff, therefore, should avoid reference to the stronger assertions employed for tradeoff resolution. In addition to being more controversial, an argument in terms of an overly powerful mechanism fails to reflect the salient issues in a decision. Ruling out the strategy "operate only if surgery will be ineffective" on the basis of a low evaluation score overlooks the commonsense reason for rejecting this policy.

Efficiency. In a large knowledge-based decision system, recognizing classes of inadmissible plans at a high level before applying general decision methods may improve performance. The advantages of precompiling the admissible plan space can be great when a large fraction of the syntactically valid plans are nonsensical.

This thesis describes an implemented tradeoff formulator, called SUDO-PLANNER, and the role of tradeoff formulation in a general architecture for planning. SUDO-PLANNER (the Synergy-driven, Utilitarian, Dominance-Oriented Planner) employs decision-theoretic principles to formulate tradeoffs in domains characterized by partially satisfiable goals and actions with uncertain effects.

1.2 An Example

A patient with a known history of coronary artery disease (CAD) and cerebrovascular disease (CVD) presents with a large abdominal aortic aneurysm (AAA).[2] The aneurysm is a dilatation of the arterial wall of the aorta, of concern because it could rupture causing death. There is a surgical procedure to repair the AAA, but this carries the risk of operative mortality or disability. The operation is especially risky for this patient because the CAD increases the likelihood of a heart attack (MI, for *myocardial infarction*) during AAA surgery and the CVD enhances the probability of a stroke.

Other available actions include tests and treatments to gauge and alleviate the

[2]This case is taken from the files of clinical decision consultations of the Division of Clinical Decision Making at the New England Medical Center [26]. In previous work, I have used the manual formulation of this decision problem to illustrate mechanisms for reasoning about preferences [156].

CAD and CVD. Table 1.1 lists the major events and actions discussed in this example.

Disease	Test	Assoc. Event	Treatment
AAA	—	rupture	AAA repair
CAD	catheterization	MI	CABG
CVD	arteriography	stroke	endarterectomy

Table 1.1: Events and actions associated with each disease in the example case.

The tradeoff formulation task for this case is simply to characterize the space of reasonable therapy strategies. Although we cannot tell from the given information whether AAA repair is recommended, we can determine that our willingness to perform the surgery should increase with the aneurysm's size, because larger aneurysms are more likely to rupture. On the other hand, indications from the diagnostic tests that the underlying diseases are worse would argue against fixing the aneurysm. From this we can conclude, for example, that the strategy "perform AAA repair if and only if catheterization reveals severe CAD" is inadmissible.

Some strategies include treatment of one or both of the underlying diseases before proceeding with the AAA surgery. A coronary artery bypass graft (CABG), for example, could be performed to treat the CAD and therefore reduce the risk of MI during AAA repair. To alleviate the risk of stroke, the patient's CVD might be treated with carotid endarterectomy, a procedure to clean out vessels leading to the brain.

Our willingness to perform these treatments is positively influenced by the severity of the underlying diseases as indicated by the test results. Therefore, there is some threshold value (CAD_1 in Figure 1.1a) for CAD extent as measured by catheterization beyond which CABG is recommended.[3] Because CAD extent is an argument against

[3]To ensure that the threshold is well-defined, we assume there exists an ordered scale for CAD extent. The discussion here and throughout the thesis also assumes that the CAD is not so extensive as to be untreatable.

AAA repair, the CAD threshold policy for this procedure is reversed. As shown in Figure 1.1b, the repair should be performed if catheterization indicates CAD less than CAD_2, but not otherwise.

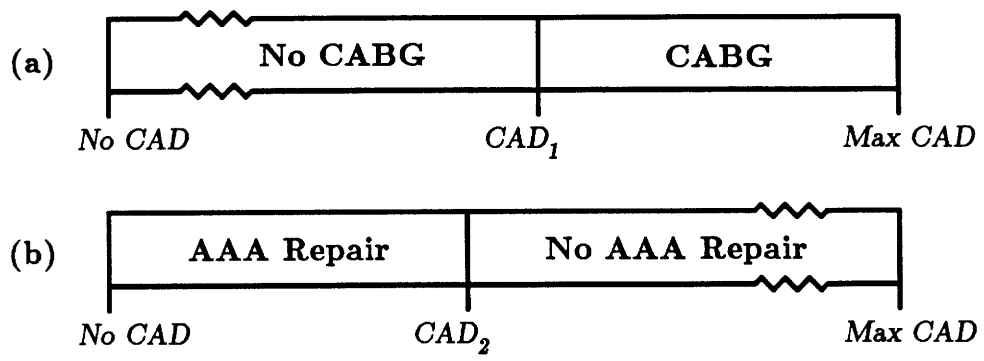


Figure 1.1: Threshold policies: (a) Perform CABG if CAD is greater than CAD_1. (b) Perform AAA repair if CAD is less than CAD_2. The accordion lines indicate that the box widths convey no scale information.

Finally, the tradeoff formulator must take into account interactions among the actions. The CAD threshold for an action is really a function of the other actions performed and other observations. In this case, because CABG alleviates the CAD, the threshold for AAA repair given CABG must be at least as high as the threshold given No CABG. In other words, CAD_2(CABG) $\geq$ CAD_2(No CABG)—performing CABG shifts the threshold to the right. Thus, if aneurysm repair is recommended for three-vessel disease (CAD extending to three coronary vessels) without CABG, it must also be recommended after a triple-bypass is performed.

1.3 Barriers to Tradeoff Formulation

Most research on decision formulation is oriented toward aiding human decision makers. The research on autonomous decision-making that takes formulation seriously also tends to treat this phase as necessarily heuristic, domain-dependent, and separate from the task of choosing among alternatives. The common view of the formulation task is as generator in a generate-and-test planning architecture.

The approach described here departs from the common view by treating the formulation task as a formal decision problem no different from choice within a more refined set of options. Because of this uniformity, plans are constructed according to the same set of principles used to compare them. Nevertheless, we can exploit the advantages mentioned above by recognizing the decisions that do not require the precise, context-sensitive knowledge needed for tradeoff resolution.

The next three sections discuss central problems in automating the reasoning behind the tradeoff formulation behavior illustrated above. The solutions incorporated in SUDO-PLANNER represent the primary technical contributions of this work. The problems are not specific to formulation; they arise in the broader task of planning under uncertainty for partially satisfiable goals. I present the problems in their full generality, even though the "SUDO solutions" that follow do not always match them in scope. Nevertheless, while SUDO-PLANNER only plans "up to tradeoffs," some of the mechanisms developed to deal with these problems are applicable to the broader planning problem as well.

1.4 Uncertainty and Partial Goal Satisfaction

Classical AI robot planners solve problems by searching for a plan of action guaranteed to transform the initial situation into one that satisfies some goal predicate. Robots taking this approach in the real world are likely to be defeated because the classical paradigm has nothing to say when a guaranteed plan cannot be found, or when one does not exist. Such situations are typical for two reasons:

1. Knowledge of the world is imperfect; in practice it is not possible to guarantee much about the result of performing actions in a given situation.

2. Predicates on world states cannot express reasonable goals for real-world agents.

1.4.1 Problem: Uncertain Effects of Actions

The classical planning paradigm is categorical. Although there may be uncertainty implicit in the incompleteness of the planner's theory, there is no provision for expressing anything about the degree to which an event is likely and its relationship to other uncertain events.

Categorical planners have the luxury of deductive inference. While it is impossible to derive everything true about the result of applying an action in a situation, many logical consequences of pre- and post-conditions can be derived. When the logical properties of a plan provably entail the goal, the planner succeeds. Figure 1.2 diagrams this planning paradigm.

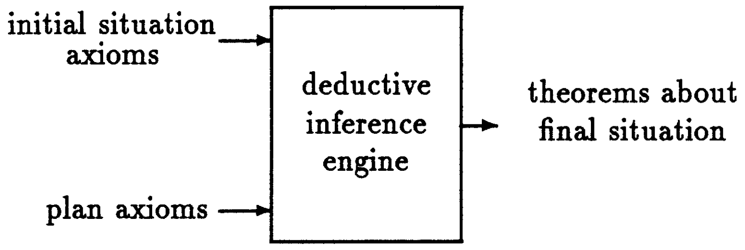


Figure 1.2: Categorical planning. Situations and actions are described by axioms. The task is to find a plan that achieves the goal predicate G in the final situation.

Because the effects of actions in the world are uncertain—at least from any robot's perspective—it is rarely possible to establish with certainty that a given plan will ensure that its goal G holds in the final situation. Failing this, a robot must consider the probability of achieving the goal with the various courses of action open to it.[4]

[4]McCarthy and Hayes, in the classic paper on classical planning [88, page 490], recognize that "the formalism will eventually have to allow statements about the probabilities of events." More often recalled from that discussion is their declaration that numerical probabilities are "epistemo-

It would appear that taking this notion seriously alters our basic planning picture dramatically. The robot must keep track of the various possible situations resulting from executing a plan and their probabilities. One way to do this, illustrated in Figure 1.3, is to relate a given plan to the likelihood that the goal is satisfied after its execution via a *probabilistic model.*

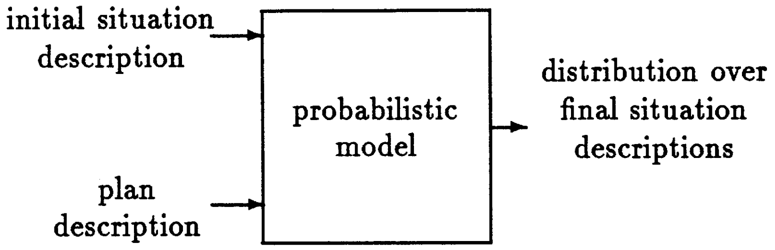

initial situation description
plan description
probabilistic model
distribution over final situation descriptions

Figure 1.3: Planning under uncertainty. A probabilistic model relates plans to outcomes. The task is to find the plan with the greatest probability of achieving G.

A major problem with this approach is the difficulty of representing knowledge required to generate probabilistic predictions. While research on categorical planning has produced modular representations that associate all knowledge with individual actions, such encodings are more difficult to create for the uncertain case. I return to this issue in Section 1.5.

Given a probabilistic model, the task presented by Figure 1.3 may still be computationally less attractive than that of Figure 1.2. In categorical planning, search can be terminated as soon as a plan guaranteed to achieve the aspiration level G is found. In general, optimizing is harder than satisficing [132]. The stopping criteria for planning under uncertainty are stricter because it is rarely possible to determine locally that a given plan maximizes the probability of goal achievement.

logically inadequate." This latter statement, however, was an objection to the proposal that the basic knowledge representation should attach probabilities to all sentences, an objection strongly endorsed in Section 1.5.1 below. It is unfortunate that subsequent work in AI planning has proceeded virtually as if likelihood did not matter.

We can avoid this problem by satisficing on the probability of goal satisfaction. This approach retains the predicate representation of goals (here the predicate contains statements about likelihood), but it is ultimately unsatisfactory for the reasons presented below.

1.4.2 Problem: Goals are not Predicates

Even if the difficulties related to uncertainty could be overcome, serious obstacles to planning within the classic framework would remain. Rarely will the robot limit its cares to the probability of a particular goal predicate being satisfied in the plan result state. Instead, the degree to which a goal is satisfied may vary, or several objectives may be achieved in partial measure. Achieving a goal with probability less than unity is one important kind of partial satisfiability, but there are others as well.

The inadequacy of a predicate representation is apparent as soon as we attempt to express the goals relevant to our medical example. Broadly speaking, our objectives are to maximize lifespan and achieve the best quality of life by minimizing disabilities and other health problems. Predicates on world situations can only divide them into situations in which the objectives are satisfied and situations in which they are not. This binary classification cannot capture even the basic preference for longer lifespans. Even when it is heuristically advantageous to plan according to aspiration levels, an agent should also have access to some representation of its actual preferences.

While it may have appeared possible to salvage much of the basic planning paradigm while admitting uncertainty, the deficiency of goal predicates undermines the fundamental structure of the methodology. Some actual planners have tried to patch this hole in the framework by including heuristic rules to handle anticipated planning decisions. McDermott's NASL [89], for example, uses *choice rules* to arbitrate among alternative task reduction paths that arise in planning. As McDermott recognizes, such an approach is vulnerable to harmful interactions with other eventualities in the planning environment, as is any scheme that associates actions directly with situations. The only way to cope with unanticipated choice contexts is to consider explicitly the predicted effects of actions and select among the alternatives according to a more general decision criterion.

1.4.3 Problem: Decision Theory and Planning

At this point we might reconsider whether it makes sense to adopt the planning paradigm at all. Indeed, it may seem that I have been setting up a straw man all along; it should have been obvious from the start that traditional planning methods are not up to the general task of planning under uncertainty with multiple objectives.

A decision criterion of the generality required is provided by Bayesian decision theory [121]. Decision theory replaces the goal predicate with a utility function mapping *outcomes* to real numbers (called *utilities*), and prescribes maximization of expected utility for decision making. The general picture is the same as Figure 1.3, augmented with a utility model for evaluating the final-situation descriptions.

With or without uncertainty, planning with decision theory inherits the computational problems of the optimization task mentioned above. In practice, decision-theoretic applications (using the methodology of *decision analysis* [109]) have been possible only because the set of alternatives is manually restricted to a small collection of strategies. This is precisely the formulation task we are trying to automate.

The space of alternatives available to an AI planner is usually specified indirectly as those generable from a given collection of primitive actions within the syntax of a plan language. This is typically an enormous combinatorial space, with nonsensical plans forming a large fraction of the syntactically allowed strategies. AI planning research has concentrated on techniques to construct complex strategies from primitive actions by searching this space.

Charniak and McDermott [13, page 523] say the following about the relationship between decision theory and planning:

> One might think that an elegant theory of this kind would have been assimilated into robot planners, but this has so far not been the case. The issues addressed by the two approaches are complementary. Planning research has focused on how plans are *constructed*; decision theory has focused on how they are *evaluated*.

Combining the two methodologies is difficult because in the ideal integration they are mutually dependent. Conventional application of decision theory requires a restricted plan set, but supplying one is precisely the formulation problem we are trying to solve. Principled plan construction calls for a decision criterion with the generality of expected utility maximization, even at the earliest formulation stages.

Previous attempts to apply decision theory to planning have avoided this apparent paradox in one of two ways. The first approach employs heuristic methods to generate a restricted plan set without reference to decision-theoretic criteria [60]. For example, the generate-and-test architecture of ONYX [76] completely separates the decision-theoretic evaluation module from the candidate generator. This separation is ultimately unsatisfactory because the heuristic generator cannot justify its decisions with respect to the agent's objectives.

The second approach adds restrictions on the form of probability and utility models to fit particular planning algorithms. For example, Feldman and Sproull [29] present some techniques that inherently take the utility of a plan to be additive in its steps. Prerequisites of this strength are rarely satisfied by planning problems, and therefore cannot serve as the basis for domain-independent planning architectures.

1.4.4 SUDO Solution: A Dominance-Proving Architecture for Planning with Partially Satisfiable Goals

The key to resolving the paradox above is to change the perspective of the planner from evaluating the results of individual plans to reasoning about the admissibility, or reasonableness, of plan classes. Rather than viewing probability and utility models as pieces of a static *absolute* evaluator, the SUDO-PLANNER architecture uses decision theory to establish the *relative* value of plans.

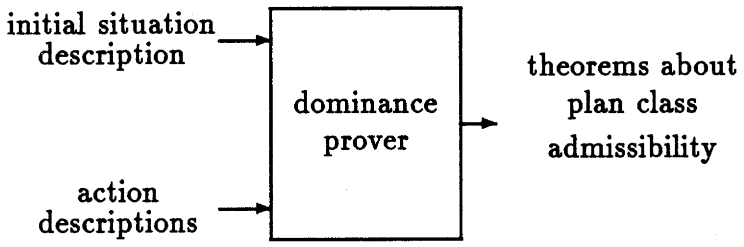

initial situation description
action descriptions
dominance prover
theorems about plan class admissibility

Figure 1.4: SUDO planning. The task is to characterize the space of admissible plans.

As illustrated in Figure 1.4, SUDO-PLANNER applies a dominance prover to characterize the space of admissible plans. The dominance prover uses knowledge about the effects of actions and the relative desirability of outcomes to derive facts about the preferred plan. In our AAA case, for example, among the plans that include CABG conditional on the result of catheterization, the dominance prover determines that the condition must call for CABG to be performed whenever the indicated CAD is more extensive than some threshold value. Nonsensical plans that include CABG for mild CAD but not for severe CAD (all else being equal) are therefore deemed inadmissible.

In order to express results of this sort, we need a representation for classes of plans. The dominance result above refers to the classes "plans with catheterization" and "plans with CABG conditional on catheterization result," among others. SUDO-

PLANNER encodes these classes by the series of constraints that distinguish them from other plan classes. A partial plan produced by a conventional constraint-posting planner can be viewed as an abstract representation of the set of all plans generated by its potential completions.

The class of plans with "CABG if catheterization result is above threshold" is a subclass of those that include catheterization with no further restrictions. SUDO-PLANNER maintains a *plan graph* that partially orders the plan classes by generality. This data structure, annotated with results derived by the dominance prover, represents the space of admissible plans.

The concept of plan class dominance is founded on the basic decision-theoretic notion of a preference relation. A preference order on prospective outcomes provides the flexibility (notably absent from situation predicates) required to express the partial satisfiability of goals. By appropriately constraining the preference order, we can specify, for example, that longer lifetimes are preferred to shorter ones, and that disabilities are undesirable.

To prove dominance, the planner needs to relate some feature of plans to the preference order on prospective outcomes, and hence to expected utility. Once identified, this relation can be incorporated into the plan graph by recording the dominance condition on the appropriate plan classes. Using the fact that dominance is inherited in the plan graph (as well as other properties of the formal definition of dominance, presented in Section 2.3), the implications of this result are propagated to other related plan classes.

In tradeoff formulation, the dominance prover's task is to detect inadmissible plan classes by applying the planner's qualitative knowledge of the effects of actions. Ideally, at the end of tradeoff formulation the plan graph includes dominance conditions ruling out these strategies. Figure 1.5 shows a piece of the plan graph derived by SUDO-PLANNER for the AAA/CAD/CVD example of Section 1.2.

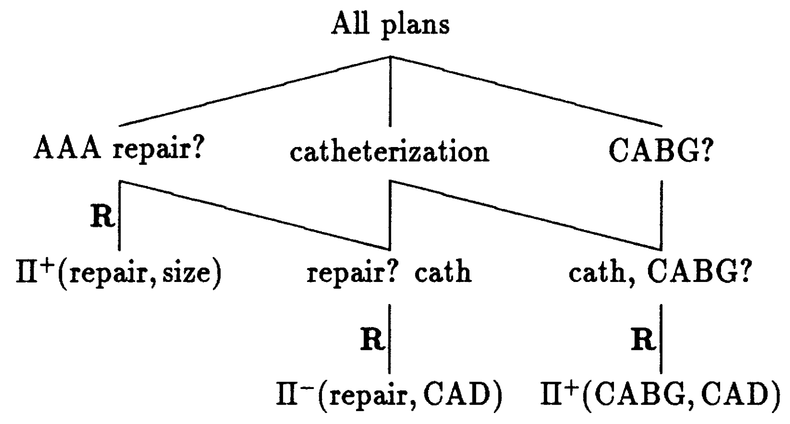


Figure 1.5: A plan graph fragment for tradeoff formulation in the AAA/CAD/CVD example.

Because I have not yet defined the plan and constraint languages, the plan classes in Figure 1.5 are informally described. The graph is a partial order on its classes, with the most general on top. Thus, the class of plans where AAA repair is under consideration ("AAA repair?") is a subclass of the universal plan class and a superclass of those in which the repair policy is a monotonically increasing function of the aneurysm size. The **R** annotation on the specialization link indicates that the dominance prover has derived a *restriction* relation (defined in Section 2.3), which implies that the planner can limit its attention to policies of the specialized form.

A complete discussion of the dominance-proving architecture, including formal definitions of the various relations mentioned here, appears in Chapter 2. The particular representations for plan classes employed by SUDO-PLANNER are introduced in Chapter 3.

1.5 Representations for Uncertain Knowledge

The dominance-proving architecture provides a framework for planning under uncertainty with partially satisfiable goals. However, the performance of the dominance prover depends critically on effective representations for the effects of actions and the desirability of outcomes. We have already seen that representations from classical planning are inapplicable to reasoning about uncertain events. In this section, I argue that traditional decision-analytic models are also insufficient for knowledge representation, and sketch the ideas behind the qualitative probabilistic networks that support tradeoff formulation in SUDO-PLANNER.

1.5.1 Problem: Decision Models are not Knowledge Bases

For purposes of decision making, a complete knowledge base (KB) specifies the joint probability distribution of all relevant actions and events. A *probabilistic model* is a representation from which a joint distribution (or parts thereof) can be computed in a reasonably direct manner. Researchers in decision analysis have developed representations called *decision models*, essentially probabilistic models augmented with a specification of preferences for the joint events.

Decision models are the obvious candidate for a knowledge representation to support planning under uncertainty. Recent advances in mechanisms for encoding and evaluating probabilistic and decision models, particularly the *belief networks* of Pearl [101, 103], have led to increasing interest in them within the AI research community.[5] From a knowledge engineering perspective, however, the idea of a decision model as a knowledge base has serious drawbacks.

The main problem is scalability. It appears feasible to build decision models of modest size, perhaps approaching the scope of today's expert systems [2, 53]. Knowledge bases in this range constrain the problem solver to a narrow decision context applicable only to a restricted set of selected cases. The more ambitious goal of building planners that operate with near autonomy over a broad range of decision

[5]Although a variety of other modeling formalisms exist, belief networks and the related *influence diagram* representation [61, 123] are sufficiently representative that little generality is lost by devoting exclusive attention to them in the following arguments.

environments calls for significantly larger knowledge bases. For example, to avoid the manual pre-selection of cases currently required by medical expert systems, a therapy planner would need at least the knowledge of a general practitioner to determine which specialized domain knowledge is relevant and applicable for a given patient. More general autonomous reasoners require a huge body of commonsense and world knowledge, perhaps on the order of Lenat's CYC KB [80].

There are several reasons to expect that decision models will not be viable knowledge representations for robot planners expected to work over a broad range of decision contexts. I discuss two of them as subproblems below and describe how these issues have influenced the design of **SUDO-PLANNER**.

1.5.2 Subproblem: Pre-Enumeration of Relevant Variables

Figure 1.6 depicts a belief network fragment presented by Kim and Pearl [70] to illustrate their representation. In the problem under consideration, an agent named Mr. Holmes gets a phone call from his neighbor reporting that his burglar alarm has sounded. He also hears a radio announcement of an earthquake in the vicinity, suggesting a possible cause of the alarm. The belief network models the relationship between the proposition that the Holmes home has been burglarized and the other factors in this situation.

Though it may be a perfectly suitable representation for this particular problem, it is unreasonable to expect that this model fragment would be available as a piece of some giant belief network KB supporting Mr. Holmes's generally intelligent behavior. Such a knowledge base commits the reasoner to a pre-enumerated set of parameters describing the relevant actions and events, and therefore could only apply to a narrow set of anticipated situations.

Imagine a KB designed to support general problem-solving about burglar alarms

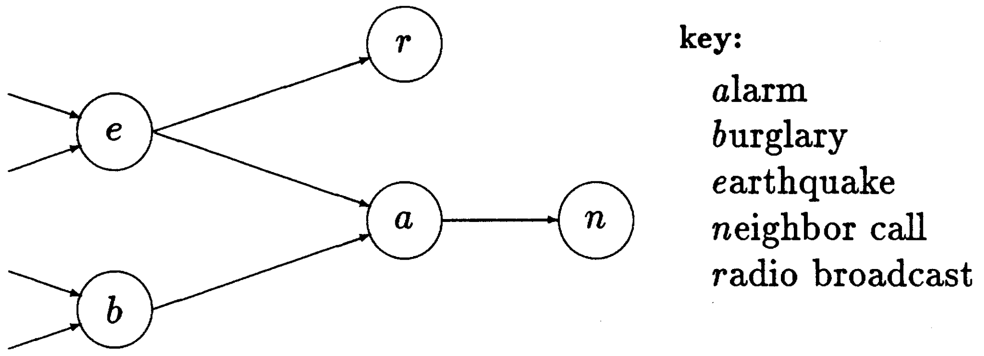


Figure 1.6: Fragment of a belief network for the burglar alarm problem.

and earthquakes. We would expect that a reasoner able to handle the problem described above should also behave reasonably when, for instance, some other neighbor makes the call, the situation takes place at night rather than day, Mr. Holmes is at some other location, or an earthquake or burglary occurred the previous day. While it is certainly possible to identify variables and relationships to encode each of these problem features, including all conceivably useful parameters in a static network KB would be infeasible.

Formally, the issue is analogous to the limitations of propositional as compared to first-order logic. If one could pre-enumerate a finite set of relevant atomic formulas, first-order quantification constructs would be superfluous. The ability to express facts about abstract situations without reference to complete detail, however, has proven to be essential in knowledge representation. Instantiation mechanisms to apply knowledge in this form to particular situations are part of every automated reasoning program.

A small increment of generality can be obtained through the use of template models. Templating supports a rigid form of instantiation where the parameter replacement paths are in effect hard-wired in advance. The set of parameters provided is restricted, and the topology of relationships is fixed.

Mr. Holmes could not be easily realized by a template. To handle the range of natural problem variants mentioned above, he would be best served by a large general body of knowledge about communication acts, news reporting, and alarm mechanisms. The ultimate pattern of instantiation from this general knowledge to concrete parameters for a given problem cannot be predicted.

1.5.3 SUDO Solution: Customized Model Construction

We can solve the pre-enumeration problem without dismissing decision models altogether by dynamically constructing models in response to the problem at hand. In this approach, the decision model is viewed as a target representation, not as a language for the KB itself.

Aside from the basic feasibility issue, customized decision models offer important advantages over monolithic decision-model knowledge bases. A model covering more than a narrow body of decision contexts would be inappropriate for any *particular* planning problem because the extraneous features entail an unnecessary computational burden and obscure explanations of the result. General models cannot take advantage of simplifying features that—while present in any given decision problem—vary from case to case.

The task of building customized models raises a wide range of issues, discussed in Section 1.6 below.

1.5.4 Subproblem: Non-Modularity of Probabilistic Assertions

The second major difficulty with decision models as KBs is the fundamental non-modularity of probabilistic assertions [51]. Unlike logical implications, statements of conditional probability cannot be combined without further information or assumptions about interactions among the conditions. Whereas in pure logical inference the derivation path leading to a fact is irrelevant to further conclusions,[6] under uncertainty the source of belief in a proposition may strongly affect its relation to other statements [102].

A consequence of this observation is that probabilistic assertions are highly sensitive to context. The association between a symptom and a disease generally depends on the patient's age, sex, and a host of other features. To support reasoning about the broad mix of patients naturally arising in practice, knowledge relating the symptom and disease must take these features into account. In the worst case (and perhaps the typical case), the probabilistic model requires a size exponential in the number of patient features [60, 144].

Expert systems and decision-analytic models cope with this problem by adopting narrow scopes. Problems are described by a relatively small number of parameters describing the important factors in the program's specialized domain. Other features are considered irrelevant or are implicitly taken into account as "background knowledge." The latter approach is reasonable as long as the population of cases fed to the program is homogeneous with respect to the background features. Builders of probabilistic systems for medical diagnosis have found that models validated for the population of a particular community cannot be reliably transported to medical centers with demographic differences [135].

1.5.5 SUDO Solution: A Qualitative Representation for Uncertainty

Another way to cope with non-modularity is to isolate some relatively context-insensitive components of probabilistic assertions and reason with these as far as

[6]The inadequacy of standard logical formalisms for representing real-world planning knowledge has led AI researchers to develop *nonmonotonic* logics that do not have this locality property and therefore share the modularity problems of probabilistic representations.

possible before resorting to knowledge encoded in less convenient representations. This approach is especially attractive for the task considered here because, as suggested in Section 1.1, knowledge required for tradeoff formulation is considerably more modular than that necessary for complete decision-making.

The *Qualitative Probabilistic Network* (QPN) formalism is designed to support precisely the knowledge required for tradeoff formulation. QPNs are abstractions of numeric probabilistic networks that encode only qualitative constraints on the joint probability distribution over the variables. Although these constraints do not determine probabilities uniquely, they support *relative* likelihood conclusions that are sufficient for the dominance results SUDO-PLANNER uses to characterize plan class admissibility.

QPNs contain two basic kinds of *qualitative relationship. Qualitative influences* describe the direction of the relationship between two variables. *Qualitative synergies* describe interactions among influences. The qualitative nature of these relationships affords robustness through context-insensitivity. While the precise numeric magnitude of the relationship between a symptom and a disease tends to vary with other factors, the *direction* of the relationship is often unaffected. For example, assessing the implications of a cardiac stress test on the probability of CAD requires consideration of the patient's age and smoking habits, among many other factors. On the other hand, the direction of the relation between test parameters and coronary disease is context-independent: for any given age and smoking behavior, CAD tends to decrease a patient's exercise tolerance.

An example of qualitative synergy is the interaction between AAA repair and CAD in their effect on MI. Both variables have a positive influence on MI: CAD increases the likelihood of a heart attack in any situation, and AAA repair increases MI risk for any given patient. The positive qualitative synergy, depicted in Figure 1.7 as a boxed plus sign connecting the two nodes to MI, holds because these influences are mutually reinforcing. That is, the *increase* in MI risk due to surgery for the aneurysm is greater for patients with more severe CAD. In other words, the expected joint effect on MI of increasing both CAD and AAA repair is greater than the sum of the two effects taken independently.

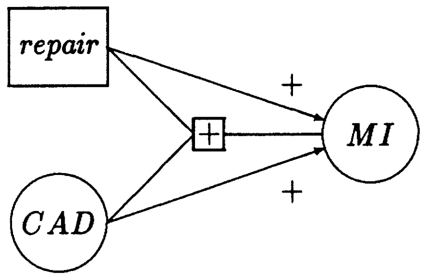


Figure 1.7: AAA repair and CAD are positively synergistic on MI. The rectangular node *repair* is a decision variable and the circular nodes are event variables beyond the planner's direct control.

Qualitative relations of this sort provide the basis for tradeoff formulation in SUDO-PLANNER. Because of their comparative modularity, a KB comprised of these constructs is more easily scaled up than one based on precise probabilistic assertions. Extending the KB to cover other cardiac disorders, for example, would not require modification of the CAD/stress-test relation, nor would a relocation of the reasoning system to a demographically dissimilar environment. This phenomenon is not merely an artifact of applying less precision; arbitrary weakenings of the assertions would offer meager modularity gains. Robustness depends on the extent to which qualitative relations capture causal structure in the domain.

Figure 1.8 illustrates a QPN for part of our AAA example. The network consists of four variables and the relationships among them. The decision variable *repair* represents the proposition that surgery to fix the aneurysm is performed. *Repair* has a negative influence on *rupt*, because surgery decreases the probability of aneurysm rupture. The *size* of the aneurysm, on the other hand, has a positive relation to *rupt*. The undesirability of ruptures is captured in the network by the negative link from *rupt* to the special *value* variable v (the hexagonal node in QPN diagrams). The negative influence from *repair* to v indicates that, aside from its known beneficial influence on ruptures, the effects of AAA surgery are undesirable.

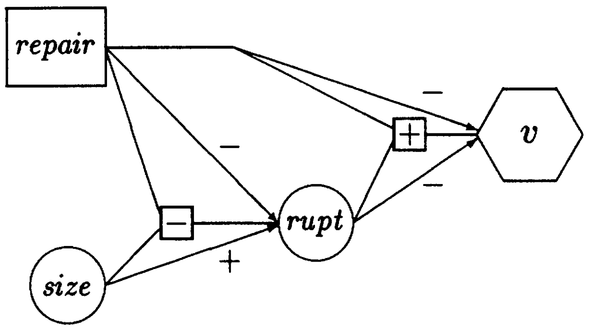


Figure 1.8: Qualitative probabilistic network for part of the AAA example.

The network also includes two qualitative synergies. First, *repair* and *size* are negatively synergistic on *rupt*. This statement means that the larger the aneurysm, the larger the *negative* influence of *repair* on rupture probability. A symmetric perspective is that performing the *repair* lessens the positive influence of *size* on *rupt*. Second, *repair* and *rupt* are positively synergistic on v because the negative effects of surgery are less important when ruptures occur.

Formal probabilistic definitions of these qualitative relations, along with a description of the QPN inference mechanisms, appear in Chapter 4. Inference in QPNs consists primarily of reduction rules for combining influences and synergies to derive the relations among indirectly related variables. For this network, reducing *rupt* reveals that *repair* and *size* are positively synergistic on v, which implies that the optimal repair policy is an increasing function of size. This simple conclusion is the dominance result derived by SUDO-PLANNER and posted on the leftmost path down the plan graph of Figure 1.5.

1.6 Constructing Decision Models

Part of our solution to the unsuitability of decision models as knowledge bases calls for dynamically constructing models from more reasonable knowledge representations. Previous attempts to automate the model generation task (reviewed in Section 6.5), however, have uncovered some serious obstacles to the goal.

1.6.1 Problem: Avoid Exhaustive Model Construction

Perhaps the greatest obstacle to automated model construction is the difficulty of avoiding exhaustive inclusion in the model of every factor in the KB. Because a model is effectively a closed world (that is, reasoners or evaluators apply a closed-world assumption when operating on the model), failure to include a factor is only justified when that factor is irrelevant to the task. However, it is typically difficult to establish that a given factor is irrelevant. In medicine, for example, it seems that any event can be related to any other by some conceivable path of associations. It would be surprising to find a large medical KB with significant disconnected components.

Reasoning with decision models cannot commence until the model is completed, or "closed." An exhaustivity constraint, therefore, delays the production of any results whatsoever from the planner. Worse, the KB could easily specify an infinite number of potentially relevant factors for inclusion in the model.

Paradoxically, this suggests that tractable models will only be generated by programs with small knowledge bases. In contrast, human decision analysts and knowledge engineers are capable of producing models and expert systems for customized tasks despite their knowledge of neighboring domains not included in their systems. To achieve similarly selective behavior, an automatic model builder requires a winnowing strategy, ideally based on sound decision-theoretic principles.

1.6.2 SUDO Solution: Model Construction at Multiple Levels of Abstraction

One powerful technique for selective model construction is abstraction. A valid abstraction is a license to ignore detail. Even detail that is relevant to the planning task as a whole need not be considered for every subtask. A planner can derive useful dominance results at high levels of abstraction before considering more detailed problem features.

For example, the direct negative link from *repair* to v in Figure 1.8 is based on knowledge found at the upper abstraction levels. Potentially undesirable consequences are associated with all treatments; *AAA repair* inherits the qualitative influence by virtue of its place in the action taxonomy. The more detailed description of the undesirable consequences of this treatment—possible surgical mortality, MI, or stroke—is inessential to SUDO-PLANNER's derivation of the threshold policy for aneurysm size, discussed above.

Explicit consideration of these specific events, however, is required to solve other pieces of this planning problem. For example, the relevance of CAD to the AAA-repair decision becomes apparent only when we introduce the variable *MI* as an event along the path from *repair* to v, shown in Figure 1.9.[7] Based on the derived implications of CAD, SUDO-PLANNER determines whether to pursue the possibility of performing CABG to decrease the risk of MI during AAA repair.

A planner capable of model-building at multiple levels of abstraction can tailor a separate model for each distinct issue it faces in designing a plan. This permits the reasoner to avoid simultaneous consideration of all the factors potentially relevant to

[7]The QPNs of Figures 1.8 and 1.9 are simplified for expository purposes from the networks directly constructed by SUDO-PLANNER. They are equivalent, however, to intermediate models produced by dominance-proving operations on the actual QPNs. The full account of SUDO-PLANNER's behavior on this example is presented in Chapter 8.

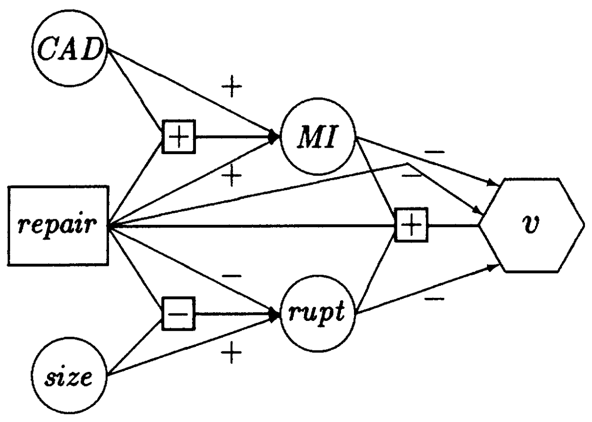


Figure 1.9: Explicit consideration of MI and CAD in the model.

the planning problem.

SUDO-PLANNER traverses its multilevel KB to generate a series of QPN models for analysis. Search is driven by the goal of deriving useful qualitative synergies. SUDO-PLANNER does not employ any sophisticated control strategies to optimize the path of model generation.[8]

As illustrated by the example, abstraction is central to SUDO-PLANNER's ability to derive useful dominance results early in the planning process. Without this feature, planning from decision models would be infeasible for large knowledge bases.

For a complete description of the model construction algorithm, see Chapter 6. Although the current methods are sufficient to demonstrate some interesting multi-level behavior, the model construction component of SUDO-PLANNER is not the final word in decision model generation. Experience with this task has suggested a set of desiderata for multilevel reasoning (presented in Section 6.1) that offer perspective on the competence of SUDO-PLANNER, the status of other AI work on abstraction, and promising topics for further research in this area.

[8]Although attention to this issue will undoubtedly be important as the KB grows, reliance on an optimized model generation procedure for a problem of the size demonstrated here would indicate extreme fragility.

1.7 SUDO-Planner Overview

1.7.1 Basic Architecture

Figure 1.10 illustrates the input/output behavior of SUDO-PLANNER. Planning problems are described in terms of changes. Assuming that current strategy is optimal given current knowledge,[9] any plan modifications must be grounded in situation changes.

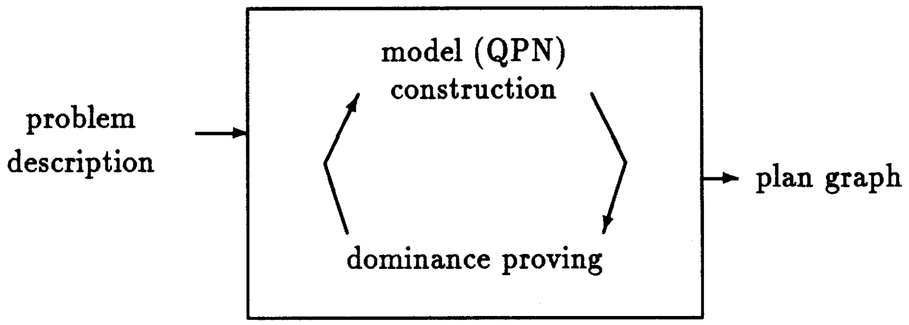


Figure 1.10: High-level behavior of SUDO-PLANNER.

SUDO-PLANNER's reasoning process consists of a repeating cycle of model construction and dominance proving. The dominance prover posts its conclusions on the plan graph as they are derived. The undominated fringe of the plan graph represents the space of admissible plans, which is the output of the tradeoff formulation process. This process does not necessarily terminate at some natural point; SUDO-PLANNER will continue to generate QPNs as long as there are potentially relevant variables in the KB.

1.7.2 SUDO-Planner on the Running Example

For the example case, SUDO-PLANNER is told only that the variable *AAA size* has changed. The initial model generated is the trivial QPN shown in Figure 1.11, which

[9]The application of this assumption is discussed in Section 6.4.

says only that aneurysm growth is bad.

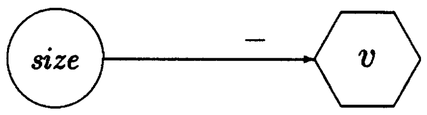


Figure 1.11: Initial qualitative probabilistic network for the example.

As it evolves the model, SUDO-PLANNER derives facts about plan class dominance, including, for example, that the AAA repair strategy should be a threshold policy on aneurysm size. Along the way, the program encounters some subtle tradeoffs, not resolvable by qualitative information alone. Given resolutions for these (discussed further in Section 7.4), SUDO-PLANNER proceeds to generate further dominance results. The QPN for this problem eventually reaches the complexity of Figure 1.12. The final plan graph incorporates all of the conclusions mentioned informally in Section 1.2, including the restriction relations shown in Figure 1.5.

1.7.3 Contributions of SUDO-Planner

The contributions of this research are the "SUDO solutions" presented in Sections 1.4 through 1.6. The main technical developments are in the dominance-proving planning architecture (Chapter 2) and the qualitative probabilistic network formalism (Chapter 4). Less tangible products of this work include:

- Identification of the tradeoff formulation task and its role in planning.
- Design and analysis of knowledge representations for plan classes, actions, and the effects of actions.
- Exploration of issues in the automatic construction of decision models.

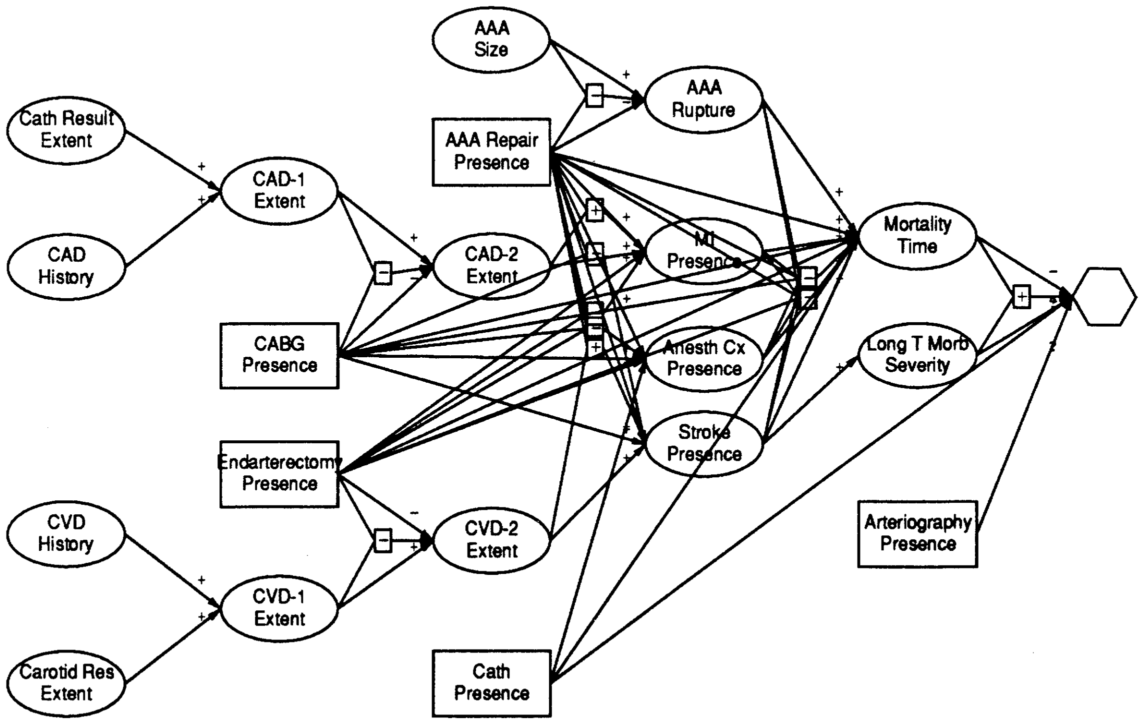


Figure 1.12: Exhaustive QPN for the AAA/CAD/CVD example.

- Demonstration of representations and reasoning strategies that effectively exploit knowledge at multiple levels of abstraction.

1.8 Thesis Preview

This introductory chapter has provided a broad perspective on the tradeoff formulation task, problems with current technology, and the solution approaches offered by SUDO-PLANNER. The remainder of the dissertation provides the details.

Chapter 2 introduces a dominance-proving framework for planning with partially satisfiable goals. Although illustrated primarily with examples from SUDO-PLANNER, the dominance-proving architecture is presented more generally, allowing for application to tasks other than tradeoff formulation. The chapter includes a generic specification for the architecture components, which is instantiated for the particular

SUDO-PLANNER mechanisms in subsequent chapters.

Chapter 3 describes the representations for plans, plan classes, and actions in SUDO-PLANNER. Discussion of the design decisions involved emphasizes the implications on computational complexity within the dominance-proving architecture of Chapter 2.

The qualitative probabilistic network formalism is motivated, rigorously defined, and illustrated in Chapter 4. The probabilistic semantics of qualitative influences and synergies justify efficient, powerful QPN inference procedures, as well as decision-theoretic results derived by SUDO-PLANNER's dominance prover. Like Chapter 2, this chapter is presented in a self-contained fashion because the results are applicable outside the scope of SUDO-PLANNER.

Some of the most difficult knowledge representation issues arise in specifying the effects of actions, the subject of Chapter 5. The chapter includes detailed descriptions of constructs for asserting effects in the SUDO-PLANNER KB, as well as discussion of the broader epistemological issues surrounding their design.

SUDO-PLANNER's techniques for automatic model construction at multiple levels of abstraction are presented in Chapter 6. To evaluate the performance of these methods, I consider properties of an ideal multilevel reasoning system and compare SUDO-PLANNER to other AI work in this area.

Chapter 7 focuses on the dominance-proving component of SUDO-PLANNER. Some subtle tradeoffs encountered in the running medical example are presented and analyzed. The example demonstrates how SUDO-PLANNER can proceed from these apparent dead ends by incorporating externally supplied tradeoff resolutions.

Chapter 8 presents SUDO-PLANNER's complete analysis of the running example.

Chapter 9 concludes the dissertation with a summary, a discussion of limitations, and some speculation about further developments and the outlook for SUDO-PLANNER.

Chapter 2

A Dominance-Proving Architecture for Planning

I have argued above (Section 1.4) that the classical planning framework is inadequate because goal predicates are not expressive enough for choice in the face of partially satisfiable goals. Decision theory, in contrast, offers a sufficiently general plan selection criterion, but says nothing about constructing plans from descriptions of the effects of component actions. In this chapter, I present a framework for applying the flexible decision-theoretic criterion in a constructive context by integrating a dominance prover into the planning process.

Dominance-proving planners reason about possible courses of action by establishing preference properties of classes of plans. The basic dominance-proving architecture represents a general approach to planning for partially satisfiable goals. The particular knowledge representations and dominance proving techniques employed by SUDO-PLANNER, developed in subsequent chapters, apply specifically to tradeoff formulation. In this chapter, I characterize the architecture abstractly, independent of this or any other specialized planning task. The architecture is illustrated with a brief discussion of SUDO-PLANNER's components. The final sections of the chapter discuss general planning issues within this framework.

2.1 The Plan Graph

Let Ω be the set of all syntactically valid plans, called the *universal plan class.* For example, if $A = \{a_1, \ldots, a_n\}$ is an alphabet of primitive actions, then $\Omega = A^*$ is the class of linear plans. The class of nonlinear plans is similar, extended by a partial order on plan steps. A *plan class* is any set of plans, $\Pi \subseteq \Omega$. We sometimes call Π a *partial plan*, to emphasize its representation as a collection of constraints incompletely specifying the plan of interest.

We can view the planning process as one of reasoning about properties of partial plans formed by adding constraints to candidate plan classes. The partial plans generated during the planning process can be organized in a specialization graph according to the subset relation. An example of a plan specialization graph appears in Figure 2.1. The node in the graph marked "$A^*a_1A^*$" denotes the set of all plans with at least one instance of action a_1. The set of plans *starting* with a_1 forms a subclass, as does the set of plans with an a_1 followed by an a_2.

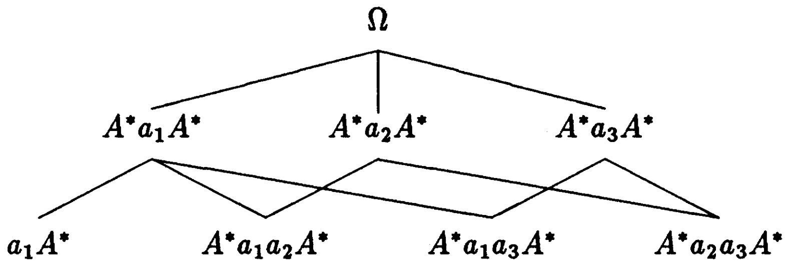


Figure 2.1: A plan specialization graph.

2.2 Constraint-Posting Planning

The plan graph representation of a search space supports a *constraint-posting* approach to planning. A constraint-posting planner—illustrated best by Stefik's MOL-

GEN [137]—refines the plan graph incrementally until some problem is solved regarding the plan to be executed. With the traditional representation of goals as predicates, the problem is to identify a satisfying plan. With partially satisfiable goals, the problem is to find the best plan. But except for some special cases where convenient optimization techniques are applicable, it is not possible to determine whether a given plan is optimal by examining it in isolation. It may be more reasonable to answer questions about the optimal plan, without necessarily constructing a complete description. I consider the issue of reasoning objectives for a planner further in the description of the dominance-proving architecture in Section 2.4.1.

A constraint-posting planner can be more efficient than a planner that only evaluates complete plans because eliminating a partial plan prunes an entire region of the search space. However, this advantage depends on having some justification for the constraints based on properties of the partial plan. For example, MOLGEN knows that for a **screen** operation to be useful, it must select the appropriate bacteria. Thus, when adding a **screen** step to a plan, MOLGEN is justified in posting a constraint of the form (**resists** *antibiotic-1 bacterium-4*). Constraining the antibiotic to a particular chemical agent would be unjustifiably specific at this stage.

By adding only the constraints that have the best justifications, a planner implements a *least commitment* strategy. An extreme form of least commitment propagates only provable properties of admissible plans. In practice, however, planners have to make guesses when no provable constraints are available. The least commitment heuristic tends to minimize both the likelihood of wrong guesses and the extent of backtracking required to recover from such mistakes. A policy of working on plan classes at upper tiers of the plan graph is a form of least commitment strategy because the partial plans there are minimally constrained.

2.3 Dominance in the Plan Graph

A *dominance-proving* planner is a constraint-posting planner that justifies constraints by determining that the plan classes they define dominate other plan classes of interest. To speak meaningfully of dominance among plan classes, we need to introduce a *preference relation*, $\succ$, over plans. In categorical planning, for example, one plan is preferred to another if it achieves the goal and the other does not. To state this in terms of the *situation calculus* [88], we write:

$$\pi_1 \succ \pi_2 \Leftrightarrow G(result(robot, \pi_1, s_i)) \wedge \neg G(result(robot, \pi_2, s_i)) \tag{2.1}$$

G is the goal predicate, defined on situations resulting from the robot performing a plan π in a given situation. Here s_i denotes the initial situation. Two plans that both achieve or both do not achieve the goal are equally preferred, or *indifferent*, denoted by $\sim$. The expression $\pi_1 \succeq \pi_2$ means that π_1 is preferred or indifferent to π_2. Indifference is an equivalence relation, and $\pi_1 \succ \pi_2 \equiv \pi_1 \succeq \pi_2 \wedge \pi_1 \not\sim \pi_2$.

The preference relation characterizes the choice criterion employed by the planner. A planner based on expected utility takes

$$\pi_1 \succ \pi_2 \Leftrightarrow E\left[u(\pi_1)\right] > E\left[u(\pi_2)\right]. \tag{2.2}$$

The discussion of dominance that follows does not depend on any particular criterion for plan choice. We do need to assume, however, that the non-strict preference relation, $\succeq$, is a total order on plans.[1] Effective dominance proving may require further regularities in the preference relation.

A class of plans *dominates* another if for any plan in the second class, there is some plan in the first that is preferred or indifferent. Formally, the dominance relation is given by the following.

[1]The architecture does *not* require that the planner be given a complete specification of the preference relation. Indeed, for our purposes $\succ$ is merely a formal device for defining plan class dominance.

Definition 2.1 (D) Π_1 *dominates* Π_2, *written* $D(\Pi_1, \Pi_2)$, *iff*

$$\forall \pi_2 \in \Pi_2 \; \exists \pi_1 \in \Pi_1 \; \pi_1 \succeq \pi_2 \tag{2.3}$$

This definition leaves open the possibility that two plan classes be mutually dominating. Also, it should be emphasized that it is possible to prove dominance without identifying the particular superior plan π_1 corresponding to each π_2.

The strict version of dominance, D', is defined similarly, except that a particular plan in the first class is preferred to any in the second.[2]

Definition 2.2 (D') Π_1 *strictly dominates* Π_2, $D'(\Pi_1, \Pi_2)$, *iff*

$$\exists \pi_1 \in \Pi_1 \; \forall \pi_2 \in \Pi_2 \; \Pi_1 \succ \pi_2 \tag{2.4}$$

Strict dominance implies dominance. In addition, the properties below follow directly from the definitions:

$$D \text{ is reflexive, transitive, and complete.} \tag{2.5}$$

$$D' \text{ is anti-reflexive and transitive (and therefore anti-symmetric).} \tag{2.6}$$

$$D(\Pi_1, \Pi_2) \Leftrightarrow \neg D'(\Pi_2, \Pi_1) \tag{2.7}$$

$$\Pi_2 \subseteq \Pi_1 \Rightarrow D(\Pi_1, \Pi_2) \tag{2.8}$$

$$D(\Pi_1, \Pi_2) \land D(\Pi_3, \Pi_4) \Rightarrow D(\Pi_1 \cup \Pi_3, \Pi_2 \cup \Pi_4) \tag{2.9}$$

$$D(\Pi_1, \Pi_3) \lor D(\Pi_2, \Pi_3) \Leftrightarrow D(\Pi_1 \cup \Pi_2, \Pi_3) \tag{2.10}$$

$$D'(\Pi_1, \Pi_2) \land D(\Pi_2, \Pi_3) \Rightarrow D'(\Pi_1, \Pi_3) \tag{2.11}$$

[2]This difference is required by the possibility of infinite plan classes with no maximal elements. If (2.4) were exactly a strict version of (2.3), then such a class would strictly dominate itself. For the same reason, a definition of weak dominance which merely substituted $\succeq$ for $\succ$ in (2.4) would not entail the reflexive property. Assuming that every plan class has a maximal plan is unreasonable, even if it is appropriate to require that the universal class Ω does.

These properties serve as dominance propagation rules within the plan graph. Plan classes trivially dominate their subclasses (2.8) because adding plans to a class can only improve its optimum. By (2.8) and the transitivity of D, dominance by a particular class is inherited in the plan graph. Strict dominance is also inherited, by (2.8) and (2.11). Thus, markers or links indicating dominance relations need be stored only at the upper envelope of classes to which they apply. Application of the union properties (2.9 and 2.10), which also hold for D', propagate dominance upwards in the graph

A plan class is *restricted* by asserting that it is weakly dominated by one of its subsets. In the MOLGEN example given above, if Π_1 is the class of plans that include the **screen** operation, and Π_2 is the subclass defined by posting the constraint (**resists** *antibiotic-1 bacterium-4*), then $D(\Pi_2, \Pi_1)$ asserts that Π_2 restricts Π_1. The new dominance assertion represents progress because it lets us focus our attention on a smaller set of plans. Deriving these restrictions is an important task of the dominance prover.

Constraints might be posted to explore the search space even though the dominance relation does not provably hold. Often, such constraints are justified by identifiable *assumptions* that imply dominance. We can express this case by asserting the *conditional dominance relation*, D_S, for S an assumption proposition.

Definition 2.3 (Conditional Dominance) $D_S(\Pi_1, \Pi_2) \equiv S \Rightarrow D(\Pi_1, \Pi_2)$.

Normal dominance is just $D_{\mathbf{true}}$. As an example of conditional dominance, suppose that we are uncertain about the identity of the organism of interest: it could be *bacterium-2* or *bacterium-3*. For $i = 2$ and 3, let S_i be the proposition "Bacterium-i is the organism of interest" and Π_i the plan class that restricts Π_1 to those plans in which the **resists** relation holds between *antibiotic-1* and *bacterium-i*. Then we have $D_{S_2}(\Pi_2, \Pi_1)$, $D_{S_3}(\Pi_3, \Pi_1)$, and $S_2 \vee S_3$. Applying Definition 2.3, we get

$D(\Pi_2, \Pi_1) \vee D(\Pi_3, \Pi_1)$. Property 2.10 yields the result $D(\Pi_2 \cup \Pi_3, \Pi_1)$.

Of particular value are conditional dominance relations where S itself contains dominance assertions. For example, if $S_1 \equiv D(\Pi_1, \Omega)$, then $D_{S_1}(\Pi_2, \Pi_1)$ asserts that if we know the optimal plan is in Π_1, we can further confine attention to Π_2.[3] Reasoning of this form can be useful for deriving restriction relations. Indeed, conditional dominance provides an interpretation for the strategies employed by Pednault [105] and Chapman [12] to limit the search space of their planners, discussed further in Section 2.8.1 below.

A dominance-based reasoner can be extended straightforwardly to handle conditional dominance by inserting a truth-maintenance layer between conditions (which correspond to assumptions or justifications) and their dominance implications [19, 86]. The interesting task for the dominance prover is to come up with meaningful conditions that imply useful dominance relations.

2.4 Dominance-Proving Planning

2.4.1 The Dominance-Proving Architecture

The schematic architecture of Figure 2.2 illustrates the roles of the plan graph and dominance relation in a dominance-proving planner.

The active modules of the planner operate on the plan graph and domain model (KB) in the following ways:

- The *problem solver* reasons about the relations among actions and events in the domain model.

[3]In categorical planning, D_{S_1} is the same as D (that is, $D_{S_1}(\Pi_1, \Pi_2) \Leftrightarrow D(\Pi_1, \Pi_2)$) due to the binary nature of the preference relation (2.1).

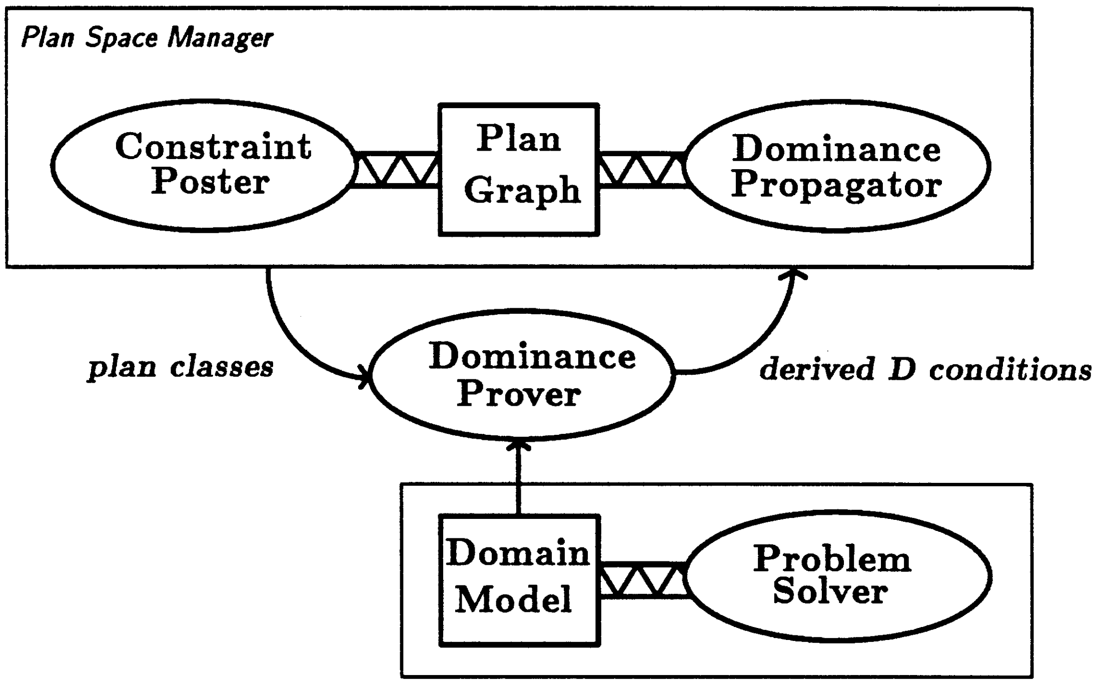


Figure 2.2: The basic dominance-proving architecture.

- The *constraint poster* generates new plan classes by adding constraints to undominated classes in the plan graph.

- The *dominance prover* applies knowledge from the domain model to derive dominance conditions (including restrictions and conditional dominance) among the plan classes of interest.

- The *dominance propagator* updates the plan graph to reflect the dominance relation D according to its properties presented in Section 2.3.

Together, the plan graph and the modules that operate on it constitute the *plan space manager*.

In this architecture, planning is not a search for a single plan to execute, but an exploration of properties of admissible plans. A planner performs useful work by refining the plan graph, even if it never reduces the lowest-level classes to singleton sets. Narrowing the admissible plans to a set that contains 10^{400} or even an uncountable infinity of plans may seem like little progress. But if we can determine that all of them contain, for instance, an appendectomy, we solve a significant problem. In general, cardinality is not an accurate measure of the refinement of a plan class. If the plan language includes real-valued parameters, then all but the tightest constraints still leave an uncountable set of candidate plans. The plan class "Administer a dose of drug X within the next minute" includes individual plans where the time the drug is given is any point in the 60-second interval.

The prevailing view of planning as the construction of a completely specified course of action is never totally accurate. Planners devote their resources to isolated questions, such as whether to perform an appendectomy or what to do *next*, without specifying all other features of the plan. A plan to obtain some bananas, for example, is complete only with respect to a *have-bananas* goal; in the larger context of satisfying all physical and emotional needs forever, the agent never stops planning. Figuring out how to get the bananas is a small act of refinement on THE GRAND PLAN.

The classic concept of a *decision* can be understood in terms of the dominance-proving architecture as a choice among exclusive and exhaustive plan classes. Decisions to eat a banana or to join the Air Force are commitments to a class of plans having a particular property, at the expense of those lacking that property. The decision-making agent remains uncommitted to the innumerable other plan class properties under its control.

The decision-theoretic formulation of decision-making as choice among basic plans (called "acts" in that literature) is a gross idealization, made explicit by Savage in his discussion of "small worlds" and "grand worlds" [121]. The plan graph representation lets us dispense with this idealization, as far as plans are concerned.[4] A dominance-proving planner can proceed as if it were working on THE GRAND PLAN because it needs never approach an explicit encoding of individual acts.

2.4.2 Instantiating the Architecture

The framework presented so far is an abstract model of planning for partially satisfiable objectives. It generalizes the case of goal predicates and applies to uncertain situations. Rather than prove that a plan necessarily achieves a goal, as in traditional AI planning, the planner tries to prove properties of the optimal plan. These properties define the class of admissible plans.

To instantiate the abstract model to a particular planning mechanism, one needs to specify each of the modules of Figure 2.2 and their interfaces. In particular, realization of a dominance-proving planner requires design of:

[4]That is, the plan classes refer to grand-world acts. But we cannot remove the idealization completely; the domain models remain small worlds, after all. For further discussion of small worlds, see Section 6.6.3.

- A universal plan class, Ω.
- A constraint language, or representation for partial plans. The set of plan classes expressible in the constraint language is $\mathcal{P} \subseteq 2^{\Omega}$.
- A domain modeling language, including a way to describe the effects of actions and a representation for the preference relation, $\succ$.

The structures described earlier—the plan graph and dominance relations—serve mainly as theoretical machinery for analysis of this class of planners. Specifying the languages and modules is the real work in designing a planner.

As a simple illustration, consider mathematical optimization techniques as planners from this perspective. Optimization is a special case of dominance proving where the program tries to find a singleton dominator, often in one step. For example, if our plan language is $\Re^n$ and the domain model consists of a linear objective function and a set of linear constraints among the elements of the vector, then our dominance prover should be a linear programming algorithm. In this case there are no partial plans. Branch-and-bound integer programming is an example of an optimization procedure that does make use of partial plans and explicit dominance proving.

For the example presented in Figure 2.1, the universal plan class is A^*: strings of actions in the action alphabet A. The constraint language consists of the *regular expressions* [59] over A.[5]

In the development of the planning model to this point, I have not addressed the issue of efficiency. The computational viability of a planner depends on a judicious choice of the languages and algorithms that define it. Although it is difficult to characterize performance at the present level of generality, we can identify a few high-level issues critical to efficiency. First, the addition of constraints during graph refinement cannot be arbitrary. The planner must generate constraints that relate to

[5]The term A^* in partial plan expressions is syntactic sugar for $(a_1 + \cdots + a_n)^*$.

the problem at hand and are meaningful to the dominance prover. Unless the prover can establish dominance relations on the graph, refinement is irrelevant. Second, it is important to consolidate the plan graph to avoid redundancy and further the propagation of dominance relations. I examine this topic further in Sections 2.6 and 2.7 below.

2.5 The SUDO-Planner Architecture in Brief

The design of languages and reasoning modules for SUDO-PLANNER illustrates one instantiation of the dominance-proving architecture. I present brief descriptions of SUDO-PLANNER's components below; subsequent chapters specify its design in detail.

A *plan* is a function from observations (events the agent can test in the world) to actions (controllable events). Any set of such functions is a *plan class*, but the plan graph may contain only those classes $\Pi \in \mathcal{P}$ expressible in SUDO-PLANNER's constraint language. The *universal plan class* is the partial plan with no constraints (Ω). A partial plan is specified by posting a series of constraints on Ω. Constraints fall in three basic categories:

- *Action.* Restricts the plan to include an action of the given type. For example, we can form the plan class "plans with CABG" by posting an action constraint of type CABG[6] on the universal plan class.
- *Action policy.* Restricts the plan's policy for the given action to have some regular (monotonic, for example) relationship to the given observable event. An action policy constraint of type "monotonic increasing" specifies the class of plans where CABG is an increasing function of CAD. This plan class, ap-

[6]The actual type specification looks more like "CABG, presence = **true**," due to the particular representation of actions in SUDO-PLANNER, described in Section 3.2. For expository reasons, in this chapter I present only the simplified versions.

pearing in the plan graph of Figure 1.5 under the notation Π^+(CABG, CAD), corresponds to the threshold policy of Figure 1.1a. For binary actions, a monotonic policy implies the existence of a threshold on the observed variable.

- *Conditional action or action policy.* As above, in effect only under a given condition on observable events. To generate the class "CABG if three-vessel disease," we would post a conditional action constraint with action CABG and condition specifying that the observed CAD (from catheterization) is three-vessel disease.

The plan constraint language is defined formally in Section 3.4.

The domain modeling language for SUDO-PLANNER is based on qualitative probabilistic networks (QPNs). Qualitative relations serve to represent the effects of actions, the connections among events, and preferences. Some QPN fragments for the AAA example appear in Figures 1.7 through 1.9.

Given these representations, the next step is to specify the modules in the architecture schematic of Figure 2.2. SUDO-PLANNER's constraint poster is a passive module, generating plan classes opportunistically. That is, a new class is introduced to the plan graph only when the dominance prover derives a potentially useful result concerning that class. In contrast, a planner under heuristic control would apply an active constraint poster to direct the search for dominance results.

SUDO-PLANNER's problem solver generates QPNs from the KB (summarized in Section 1.6.2, discussed in depth in Chapter 6). There are two parts of the domain model at any given time: the KB as a whole and a dynamically constructed QPN. The problem solver uses facts from the KB to progressively modify the QPN throughout the process of tradeoff formulation.

The dominance prover operates on QPNs rather than directly on the KB. Dominance proving in SUDO-PLANNER consists of manipulating QPNs to derive qualitative relations with implications for plan class dominance. The central basis for deriving dominance in QPNs is qualitative synergy between an action and an event on the value node. Such synergies dictate action policy constraints on plan classes, as explained in Section 4.7.3.

For example, based on a QPN generated by the problem-solver, SUDO-PLANNER determines that CABG and CAD are positively synergistic on *value*. This relationship directly implies that the optimal CABG policy is increasing in CAD severity. Let "cath, CABG?" denote the class of plans that include catheterization and make some commitment about whether to perform CABG. Any plan in this class that is not a threshold policy must be inferior to some plan in $\Pi^+(\text{CABG}, \text{CAD})$. The situation satisfies the condition for plan class dominance (2.3), yielding

$$D(\Pi^+(\text{CABG}, \text{CAD}), \text{"cath, CABG?"}).$$

Because $\Pi^+(\text{CABG}, \text{CAD})$ is a subclass of "cath, CABG?" the dominance relation is a restriction, as posted on the plan graph of Figure 1.5.

The dominance propagator records these results on the plan graph and derives their consequences for dominance among other plan classes. For instance, adding a new plan class intermediate between these two would modify the restrictions recorded on the plan graph, shown in Figure 2.3.

$$\begin{tikzcd}
\text{cath, CABG?} \arrow[dd, dash, "\mathbf{R}"'] & & \text{cath, CABG?} \arrow[d, dash, "\mathbf{R}"'] \\
{} \arrow[rr] & & \textit{intermediate } \Pi \arrow[d, dash, "\mathbf{R}"'] \\
\Pi^+(\text{CABG}, \text{CAD}) & & \Pi^+(\text{CABG}, \text{CAD})
\end{tikzcd}$$

Figure 2.3: Dominance propagation upon insertion of an intermediate plan class.

The overall behavior of SUDO-PLANNER is best described by a superposition of the architecture schematics of Figures 1.10 and 2.2. The product of the process is the plan graph reflecting dominance results SUDO-PLANNER derives from the sequence of QPNs produced by its problem solver.

2.6 Searching the Plan Space

Different designs for the plan constraint language, knowledge representations, and processing modules can lead to dominance-proving planners with disparate behaviors. Nevertheless, the common architectural skeleton provides a basis for a general discussion of some computational issues for this family of planners.

One important generic issue concerns management of the search space. In the dominance-proving architecture, the search space is represented by the plan graph, and the dominance relation supplies the criterion for pruning the space.

Search in constraint-posting planners proceeds to refine the plan graph by adding constraints to partial plans. The straightforward application of this procedure, however, can lead to considerable redundancy in search. For example, Figure 2.4a shows a refinement to the left side of the plan graph of Figure 2.1. The new partial plan is obtained by posting a constraint on a_1A^* that the second action be an a_2.

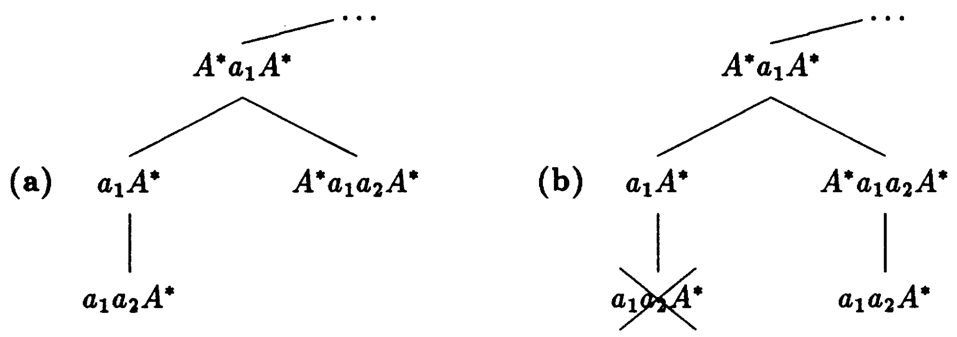


Figure 2.4: Redundancy in plan graph search. (a) Posting a constraint on a_1A^* yields $a_1a_2A^*$. (b) The same plan class is obtained by posting a constraint on $A^*a_1a_2A^*$.

Suppose that after further reasoning the planner decides to prune away the new plan class based on dominance or some other criterion. The same class may be re-introduced on refinement of the next branch of the plan graph. As illustrated in Figure 2.4b, constraining the first A^* to be empty in $A^*a_1a_2A^*$ defines a partial plan identical to that already pruned. A planner that did not recognize such coincidences would duplicate much of its dominance proving and other reasoning efforts.

The *chronological backtracking* strategy described above is inefficient because it cannot transfer reasoning results between contexts. *Dependency-directed backtracking* [136] avoids considerable redundancy by determining the most general reasons for a failure and pruning a larger region of the search space. In our example, however, the redundancy problem emerges on separate branches of the search space. More flexible dependency-directed reasoning mechanisms, such as truth maintenance systems (TMSs) [23, 86], are required to transfer results across arbitrary contexts.

2.6.1 Classification

The redundancy of Figure 2.4b went unrecognized because the plan graph at the first stage (Figure 2.4a) failed to reflect all specialization relations. This suggests that we add to the plan space manager a third module operating directly on the plan graph, as shown in Figure 2.5. When the constraint poster generates partial plans, the *classifier* situates them in the plan graph by computing their greatest lower and least upper bounds. This classification operation is identical to that performed by terminological knowledge representation languages like KL-ONE [8, 122].

The correct classification for $a_1a_2A^*$ in the plan graph of Figure 2.4 places this class under both a_1A^* and $A^*a_1a_2A^*$. Consolidating the plan graph in this way avoids redundancy in plan space search. The additional specialization links enhance pruning and provide more pathways for propagation of dominance results throughout the plan graph.

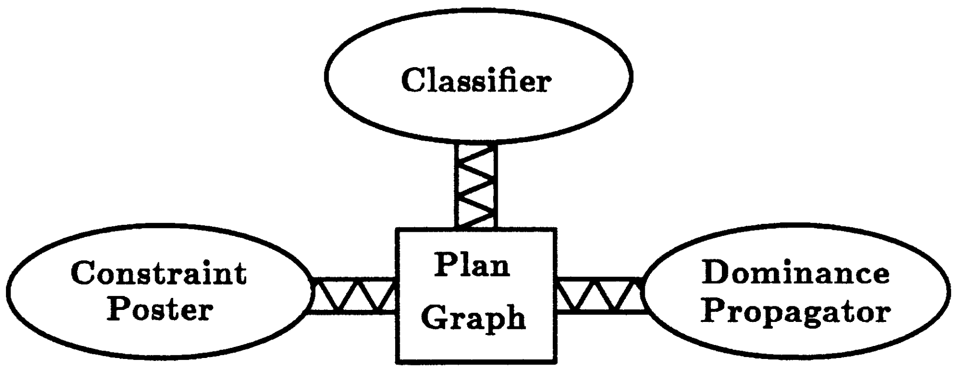


Figure 2.5: The plan space manager augmented with a classifier to consolidate the plan graph.

2.6.2 Dependency Mechanisms

In the context of plan space search, classification can be viewed as a kind of dependency mechanism for minimizing duplication in reasoning. Comparison of classification to other techniques is difficult because a precise characterization of the avoidable redundancies is often lacking in descriptions of dependency-directed problem solvers.

From the perspective of dependency maintenance, the dominance relation serves to define so-called *nogood* contexts. Planners without a notion of dominance can consider plan classes nogood only if they are inconsistent or provably cannot achieve the goal. Other pruning criteria must be built into the control mechanism. The availability of nogoods based on dominance can dramatically reduce the search space.

Following the terminology used by de Kleer in describing his *assumption-based* truth maintenance system (ATMS) [19], each plan class is an *assumption context* represented in terms of the constraint language. The plan specialization graph corresponds to the context graph of the ATMS with context subset replaced by plan specialization. Indeed, an implementation using an ATMS would be equivalent to the classification scheme presented here *provided that* we could construct a propositional interface [20] describing the content of partial plans. In general, however, this may not be the most convenient representation of plan classes.[7] As shown in Section 2.7, a mapping of the constraint language to sets of propositions may necessitate an unacceptable growth in the size of partial plan descriptions.

Like the assumption-based approach, the plan graph structure facilitates the exploration of multiple consistent contexts simultaneously. But contrary to the ATMS view, we are not necessarily interested in finding all solutions (the class of all admissible plans). Rather, the planner may focus on an isolated decision defined by a particular distinction in the plan graph. Therefore, we can restrict the domain of assumption sets that need to be considered to those explicitly created as plan classes by the constraint poster.

2.6.3 Plan Graph Queries

Another application of classification is to answer dominance questions about plan classes not explicitly appearing in the plan graph. At any point in the planning process, the plan graph represents the current view of the admissible plan space. To find out whether a given plan class is admissible, we merely classify the class in the plan graph and propagate dominance as usual. Queries about the classes dominating or dominated by this new class can then be answered by local inspection of its position in the plan graph.

2.6.4 Plan Space Search: Summary

Redundancy in searching the plan space is avoided in the dominance-proving architecture by classifying plans in the plan graph as they are generated. The plan graph

[7]The design of SCHEMER [166, 167]—a dependency-directed interpreter for a non-deterministic LISP—illustrates some issues in constructing a propositional interface for arbitrary dependencies.

thus serves a dual role as representation of the admissible plan space and as the main structure for dependency maintenance.

The scheme presented here offers two benefits with respect to plan search efficiency. First, the dominance relation provides a major new class of nogoods, which may potentially shrink the search space. Second, classification of partial plans takes advantage of dependencies that might be obscured by an interface with a propositional TMS.

2.7 The Complexity of Subsumption

In terminological knowledge representation languages, the key operation for concept classification is the computation of *subsumption* relations [7, 96]. The same applies to classification of partial plans; the appropriate position of a plan class in the plan graph is determined by its subsumption relations to other classes. One plan class *subsumes* another if the latter must be a specialization purely by virtue of its description. We saw above that classifying plans as they are generated minimizes the search space.[8] Conversely, an optimal dependency mechanism (one that minimizes search) in effect computes these subsumptions.

The centrality of subsumption to classification suggests that the complexity of this operation has a large impact on the computational performance of this planning architecture. Consequently, analyses of dominance-proving planners should emphasize the complexity of subsumption in the chosen constraint language.

As an example, consider the plan graphs of Figures 2.1 and 2.4, where regular expressions describe the partial plans. In general, computing subsumption of regular

[8]For true optimality we must also determine the most general plan class that is dominated. This corresponds to extracting the minimal nogood assumption set, which is not generally feasible. Note also that minimality is only with respect to a particular constraint language; slight changes may have dramatic effects on the dominance prover's ability to derive nogood sets at high levels of generality.

expressions is intractable.[9] By restricting the constraint language to expressions that use disjunctions and Kleene stars exclusively in A^* terms, however, subsumption can be performed in linear time (the problem is essentially string matching with wildcards). All of the plan classes included in the figures above satisfy this restriction.

We can also analyze some more common partial plan representations in terms of subsumption complexity. *Nonlinear plans* [118] are partially ordered sets of actions, used as a basic representation by numerous constraint-posting planners [12]. A nonlinear plan is actually a plan class, comprising all possible linearizations of the partial order [21]. (A planner with a parallel execution capability (for example, a multi-agent planner) could truly have nonlinear plan individuals. The constraint language for such a planner would be more complex.) Unfortunately, partial order—and hence nonlinear plan—subsumption is NP-complete.[10] The exponential potential of the computation lies in the combinatorial number of possible mappings between the steps of the two plan classes. If, however, we can specify the correspondences between steps (for example, which *put-on* in Π_1 corresponds to which in Π_2) then subsumption is at worst quadratic. In practice we will not generally have complete correspondences, but typically the possible mappings between steps will be constrained by one or more of the following factors.

- Actions may only map to others of the same type.
- Compatibility among constraints on the arguments to actions.
- The planner may supply explicit identifications among steps when introducing them in several partial plans at once.

[9]Hunt et al. [65] show that the equivalence problem for regular expressions is PSPACE-hard. Equivalence can be trivially reduced to two (non-strict) subsumption operations.

[10]The proof, by reduction from EXACT COVER BY 3-SETS, was provided by Ronald L. Rivest, personal communication.

It is an empirical question whether these constraints will render subsumption feasible in practice.

The intractability of nonlinear plan subsumption illustrates the potential difficulty of specifying a propositional interface to generic dependency reasoning mechanisms like the ATMS. In the ATMS, contexts are represented by sets of propositions, and context subsumption is simply the subset operation. Determining whether one context specializes another can therefore be accomplished in time at worst quadratic in the size of the proposition sets. Given that nonlinear plan subsumption is NP-complete, the size of the propositional representation for the plan classes must be exponential in the partial order encoding, unless $P = NP$.

One should consider the effect on subsumption complexity of any proposed extension to the constraint language. For example, we could allow actions themselves to be expressed at multiple levels of abstraction (as in the sequence *low-dose steroid therapy* is-a *steroid therapy* is-a *drug therapy*) without significant cost in complexity, as long as action subsumption itself is not expensive.[11] I analyze the tradeoff between expressiveness and subsumption tractability for SUDO-PLANNER's constraint language in Section 3.5.

2.8 Miscellaneous Topics

The following sections discuss some implications of the dominance-proving architecture for two issues in knowledge-based planning.

[11]Given a static action graph, action subsumption takes linear time in the worst case. If actions are described more flexibly—perhaps as dynamically generated KL-ONE concepts—then subsumption is more complex [7].

2.8.1 Dominance-Based Knowledge Level Analysis

In his description of the *knowledge level*, Newell proposes that programs can be characterized in terms of the content as opposed to the form of their knowledge by analyzing their behavior with respect to principles of rationality [97]. The dominance relation can a be useful tool for such analyses, whether or not the planner resembles the dominance-proving architecture at the *symbol level*. By admitting partially satisfiable goals, the preference order on plans provides a more comprehensive and flexible rationality principle than one based entirely on goal predicate satisfaction. Note that a program may still be nondeterministic at the knowledge level because we may have an incompletely specified preference order.

Even though the categorical preference relation does not exercise its flexibility, the dominance relation can shed some light on classical planners. In a previous description of this work [158], I illustrated the dominance-proving architecture by recasting TWEAK [12] in its framework. Because TWEAK encapsulates the results of much of the state-of-the-art in nonlinear planning, an account of this planner embodies considerable generality.

The product of this exercise was an interpretation for TWEAK's central principle—the "modal truth criterion" [12, page 340]—in terms of dominance in the plan graph. TWEAK searches the plan space by posting constraints on partial plans until it finds one that achieves its goals. The search is not exhaustive, yet TWEAK is guaranteed to terminate with a solution if one exists. The source of its power lies in the modal truth criterion, which justifies TWEAK's selectivity in exploring the search space. Briefly, the criterion specifies that if a plan class contains a successful plan, then search can be further limited to the subclasses obtained by posting particular sequences of constraints. In our dominance-proving terminology, this is an instance of a *conditional restriction* as described in Section 2.3. For a formal statement of the dominance condition, see the previous paper [158]. Although the details have not been worked out, I expect that a similar interpretation could be developed for Pednault's results for a more expressive constraint language [105].

The utility of the dominance relation as an analytical device is potentially more significant in applications to planners that consider partially satisfiable goals. A variety of AI programs perform resource-allocation tasks, where goal satisfaction is always a matter of degree. In general, the programs employ domain-dependent search heuristics and ad hoc representations, so it is difficult to assess and compare alternative techniques.

One well-known AI resource planner is ISIS, a program that schedules production orders in a job shop [37, 38]. ISIS constructs schedules through a heuristic constraint relaxation process. The program is difficult to evaluate empirically because of its large number of parameters and the lack of a performance standard for its scheduling task. Theoretical analysis also presents a challenge because interactions among the constraints in the program's domain are not well understood.

Recasting ISIS in terms of the dominance-proving architecture may lead to insights about its performance. The framework of this chapter enforces a strong distinction between the non-relaxable constraints and those that are relaxable, called "organizational goals" or "preference constraints" in ISIS terminology. The relaxable constraints serve to define the preference relation on schedules, while the non-relaxable are part of the domain modeling language. The partial schedules ISIS constructs during search correspond to partial plans, as do the abstract schedule classes defined by sets of relaxable constraints. Preferences among various relaxations induce a dominance relation on the plan graph.

The sketchiness of this description of ISIS implies that any claims for the value of analyses of this sort are purely speculative. Nevertheless, the dominance relation provides a criterion for determining the soundness of ISIS's relaxation procedure, and classification in the plan graph is a standard for evaluating the redundancy of the program's search for optimal production schedules.

2.8.2 On Meta-Planning

The idea of *meta-planning* [24, 138, 163] is to apply the machinery of a planner to decisions about the planning process itself. Because planning algorithms are often complicated and opaque, a declarative encoding of the goals and operators at higher levels should lead to more understandable and modifiable planners.

It is important to distinguish two types of higher-level decisions that meta-planners address:

1. allocation of the planner's computational resources, and

2. choice among competing plans or plan fragments.

Most of Wilensky's "meta-themes" and "meta-goals" [163, Section 2.2] relate to the second type of decision.

Choice among competing plans is appropriately considered a "higher-level" issue if the object-level planner lacks the basis for making this decision. This is typically the case for traditional planners because the predicate representation of goals offers only a crude binary distinction. Designers of planners have historically dealt with this inadequacy by developing elaborate conflict resolution mechanisms. Wilensky's meta-planning enterprise was an attempt to demystify these schemes.

A flexible representation for plan choice via the preference relation, however, obviates the need for meta-level decision mechanisms. The meta-theme "don't waste resources" is more accurately expressed as a dominance condition on plans that use more resources, other things being equal. The dominance condition offers a precise semantics for such a statement; interpretation of the meta-theme, in contrast, is not so clear.

The first type of high-level decision—allocation of computational resources—remains a legitimate activity for meta-planning in the dominance-proving framework.

2.9 Summary

In this chapter I have presented and discussed a dominance-proving architecture for planning for partially satisfiable goals. The architecture takes a constraint-posting approach to planning, representing the search space by a plan graph that encodes the specialization relations among the partial plans generated. Goals are defined by a preference order over plans, which induces a dominance relation defined over plan classes. I motivated these concepts with simple examples from MOLGEN and a more detailed account of SUDO-PLANNER.

Redundancy in searching the plan space is minimized by classifying the partial plans in the graph as they are generated and pruning based on dominance. Recognizing the centrality of subsumption computation in classification suggests a novel approach to analyzing the complexity implications of plan constraint languages.

The dominance-proving architecture avoids the pitfalls of goals as predicates by adopting a more general representation for the planner's objectives. Exploiting this flexibility in a computationally feasible manner requires careful design of the component modules and representations that compose the dominance-proving architecture.

Chapter 3

Representations for Plans and Actions

In this chapter I specify the representations SUDO-PLANNER employs for expressing plan classes and actions. Section 3.1 characterizes the plan space abstractly as a formal basis for defining the semantics of plan class constraints. The next two sections describe the action and event representations employed by SUDO-PLANNER for both domain modeling and expressing plan classes. These specifications fulfill part of the requirements, outlined in Section 2.4.2, for instantiating the dominance-proving architecture in SUDO-PLANNER. The chapter concludes with an analysis of the computational complexity of subsumption for SUDO-PLANNER's plan constraint language.

3.1 Plans

A *plan* is a specification for a course of action, directly executable by an agent. SUDO-PLANNER does not manipulate plans directly; the dominance-proving architecture prescribes computations on plan classes only. Therefore, we can design this type of planner without specifying an encoding for individual plans, as long as the interpretation for plan class constraints is clear. Of course, an integrated planning and execution system would require a complete plan language specification.

3.1.1 Planning and Execution

Abstractly, plans are functions from observations to actions. Formally, given

$\mathcal{O}$ — the set of possible patterns of observation over time, and

Ω_{exe} — the set of executable courses of action,

the universal plan class Ω consists of all functions $\pi : \mathcal{O} \rightarrow \Omega_{exe}$. Ω_{exe} defines the scope of pure activity the agent is capable of performing. Plans map situations, distinguished by their observation patterns, into activities. Intuitively, a course of action $\omega \in \Omega_{exe}$ specifies a set of primitive end-effector commands unconditional on any explicitly planned observations.

Figure 3.1 illustrates the relationship between the planning and execution modules in a complete agent. The plan to be executed, $\omega_{obs} = \pi(obs)$, specifies all planned activity, including any sensing operations that enable the agent to observe *obs*.

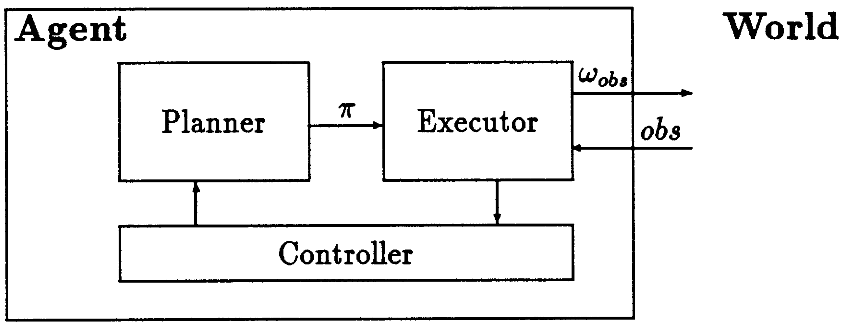


Figure 3.1: An integrated planning and execution system.

Time is handled implicitly in this functional notation. All temporal information must be encoded in the descriptions for observations and courses of action. This treatment is unsatisfactory because it cannot enforce fundamental temporal constraints, such as the prohibition on conditioning current activity on future events.[1] The per-

[1]For further discussion of SUDO-PLANNER's atemporality, see Section 9.2.2.

missiveness of this syntax implies that ultimately we must decide the legality of plans on semantic grounds. The class of legal plans, Ω_0, is given informally by

$$\Omega_0 = \{\pi \in \Omega \mid \pi \text{ satisfies semantic constraints}\}. \tag{3.1}$$

A more restricted plan language syntax, perhaps incorporating some of these temporal and other legality constraints, would be provided for any particular integrated planning and execution system. The context also dictates computability constraints on π, perhaps requiring real-time performance. REX, a language designed for situated planning [114], is an example of the kind of representation suitable for the plans passed to the executor in the framework of Figure 3.1.

3.1.2 Cascaded Planners

We can generalize this framework by decomposing the planning module into a series of submodules, each of which is itself a planner. As shown in Figure 3.2, the output of each module is a plan class, which serves as the root of the plan graph for the next planner in the sequence. Each planner has its own constraint language, determining the plan classes $\mathcal{P}_i \subseteq 2^{\Omega}$ expressible in its plan graph. The $\mathcal{P}_i$ also constrain intermodule communication: planner-j must output a plan class $\Pi_{j+1} \in \mathcal{P}_{j+1}$.

A variety of cascaded planning architectures can be defined by specifying conventions regarding the plan constraint languages. For example, the condition $\mathcal{P}_i \subseteq \mathcal{P}_{i+1}$, $1 \leq i \leq n-1$ entails a successive refinement of the plan spaces for the cascaded planners. On the other hand, if $\mathcal{P}_i \supseteq \mathcal{P}_{i+1}$ for that range, successive planners possess cruder distinguishing abilities. Such an architecture might be appropriate for a system in which high-level planners pass on partial results to lower-level control algorithms. In general, however, the $\mathcal{P}_i$ will overlap, reflecting divisions of labor according to expertise or other resources.

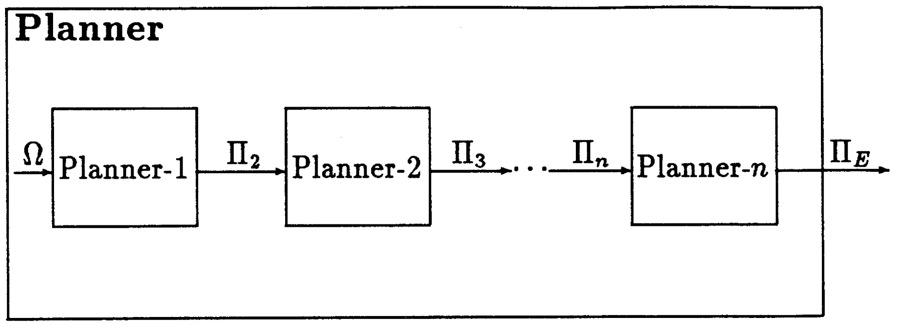


Figure 3.2: Cascaded planners. Planner-i maintains a plan graph with plan classes expressible in its constraint language, $\mathcal{P}_i$. The output of the system, Π_E, is the plan class passed to the executor.

The final planner in the chain, planner-n, produces a plan class Π_E for the executor. The framework of Figure 3.1 implicitly assumes that the constraint language for the executor admits only singleton plan classes.

$$\mathcal{P}_E = \{\{\pi\} \mid \pi \in \Omega_0\}.$$

This restriction on $\mathcal{P}_E$ is in no way essential to the view of planning and execution presented here.

In the cascaded planning framework, SUDO-PLANNER plays the role of planner-1. The constraint language described in Section 3.4 defines the expressible plan classes, $\mathcal{P}_1$. I assume planner-2's constraint language to be sufficiently expressive so that $\mathcal{P}_2$ includes the admissible plan spaces output by SUDO-PLANNER.

3.2 Action Representation

In contrast to the abstract plan language discussion above, this section describes concrete representations for actions as they appear in the SUDO-PLANNER knowledge base. Actions play a part in defining two components of SUDO-PLANNER:

1. Constraint language. Actions are the building blocks of plans. Plan classes are described by constraints on the actions making up their constituent plans.

2. Domain modeling language. The effects of plans are determined by the effects of their component actions.

This section presents the basic action representation, with an emphasis on features necessary for the constraint language. Issues in representing the effects of actions are the subject of Chapter 5.

3.2.1 Action Taxonomies

The knowledge base of actions is implemented in NIKL [66, 150], a terminological knowledge representation language based on KL-ONE [8]. Action types are represented by NIKL *concepts*, frame-like specifications of classes of individuals. Concepts are defined by their position in the concept lattice and restrictions on their *roles*, or associated features.[2] Roles are also taxonomized, according to the generality of their domains (concepts for which the feature is relevant) and ranges (possible values of the feature). The *extension* of an action type a, denoted $\mathcal{X}(a)$ is the set of actions satisfying its specification.

The NIKL taxonomy of action concepts specifies a multilevel description of actions that plans may include. Let **action** be the root of the taxonomy, a superconcept of every other action concept a_i. $\mathcal{A} = \mathcal{X}(\textbf{action})$ is the set of all actions. One way to specialize partial plans in the plan graph is to specialize individual action types included in that plan class. If Π_1 includes action $a_1 =$ **drug-therapy**, the class of plans Π_{1D} that replace a_1 with $a_D =$ **drug-therapy**(D) is a specialization of Π_1.

The dimension of specialization in the example above corresponds to the **drug**

[2]For an alternate action representation implemented in NIKL, see Swartout and Neches [142]. In their scheme, plans as well as actions are represented as NIKL concepts.

role of action a_1. The concept **drug-therapy**(D) is defined by a *value restriction* [8] on that role, limiting its range to D.[3] Additional specializations might restrict the drug further (perhaps D is a family of drugs) or may include other information, such as dosage or method of administration.[4] In realistic knowledge bases the axes of specialization are quite numerous.

Action-type abstraction is used in MOLGEN [137] and other constraint-posting planners [95]. Although MOLGEN lab operators constitute a simple two-level hierarchy of actions, in combination with the hierarchy of lab objects in their domain they form a rich, multiply hierarchical action structure. From this perspective the lab objects are merely further axes for specializing the lab operators.

As Tenenberg points out [145], this specialization form of abstraction is orthogonal to the step-components type of abstraction hierarchy used in NOAH [118]. In NOAH actions may be decomposed into sequences of lower-level actions, effectively viewing each action as a sub-plan. A decomposition hierarchy associates actions by a "part-of" rather than an "is-a" relation. SUDO-PLANNER does not explicitly support this sort of abstraction.

Another distinct kind of abstraction is the precondition hierarchy of plan spaces Sacerdoti introduced in ABSTRIPS [117]. Abstract actions are defined implicitly by temporarily ignoring preconditions. Note that preconditions refer to the implications of including an action in a plan, rather than to features intrinsic to the action. Ordering the preconditions imposes structure on the search space rather than on the actions themselves.

Basing the knowledge representation on a terminological reasoner like NIKL has

[3]I adopt the notation a_i(*value-restriction*) for convenience whenever the role involved is obvious in context.

[4]As in planning, this specialization process never ends. Actions exist in the knowledge base only as types; the action individuals specifying an actual plan π are required only if the planner is integrated with an execution capability as in Section 3.1.1.

the important advantage of dynamic flexibility. New action types may be created at planning time by adding constraints to, specifying new features of, or combining existing action concepts. These new action types can often be automatically classified [122]; that is, NIKL can determine the location of the new type in the existing taxonomy. This is important because

1. an action type's place in the taxonomy determines the properties it inherits from and supplies to other types, and

2. action subsumption is employed by the constraint language subsumption algorithm of Section 3.5 to classify plan classes.

Some simple examples of dynamic action creation in SUDO-PLANNER during dominance proving are described in Section 7.2.

3.2.2 Action Variables

As described in Section 3.4 below, the plan constraint language uses action types to restrict a plan class to plans including actions with certain features. It is sometimes convenient to refer to properties of action features directly, for example, in relating the dosage of a drug to other characteristics of events or plans. The SUDO-PLANNER objects denoting such features are called *action variables.*

Action variables are represented by pairs of NIKL concepts and roles. Drug dosage, for example, is the combination of the action **drug-therapy** with its role **dosage**. Other variables are constructed by pairing the concept with other roles, such as **duration** and **method** (whether the drug is given as a pill, intravenously, or in some other form). The notation $av \equiv role(a_i)$ makes the components of an action variable type av explicit. The *domain* of the action variable is the value restriction on its role component.

Like actions, SUDO-PLANNER action variables are types, not individuals. A taxonomy of action variables can be defined in terms of the concept and role taxonomies. One action variable specializes another if and only if both the action concept and role of the first specialize the corresponding components of the second.

For each action variable *instance* AV, there is a function $AV(\omega)$ that returns AV's value in the course of action $\omega \in \Omega_{exe}$. If ω does not commit to a value for AV, the function is undefined.

3.2.3 SUDO-Planner Actions

Figure 3.3 illustrates the form of action definitions in NIKL. The definition of **surgery** declares that it is a subconcept of **action** (the NIKL concept for SUDO-PLANNER's universal action class $\mathcal{A}$) and that it has a role named **route** with the value restriction **invasive-path-into-body**. The action **open-lung-biopsy** is defined similarly.

A **surgery** is an **action**
with one **route** which is an **invasive-path-into-body**.

An **open-lung-biopsy** is a **biopsy**
with one **route** which is an **open-lung-path**.

Figure 3.3: Stylized NIKL definitions for the actions **surgery** and **open-lung-biopsy**.

SUDO-PLANNER uses NIKL for two types of inference: classification and inheritance. Given that **biopsy** is a subconcept of **action**, and that **open-lung-path** is a kind of **invasive-path-into-body**, NIKL's classifier can determine that an **open-lung-biopsy** is a kind of **surgery**. Facts about surgical actions—that they require anesthesia, for example—are automatically inherited by **open-lung-biopsy** and other specializations of **surgery**.

The SUDO-PLANNER action knowledge base includes all of the actions for the running AAA example. A portion of this taxonomy appears in Figure 3.4.

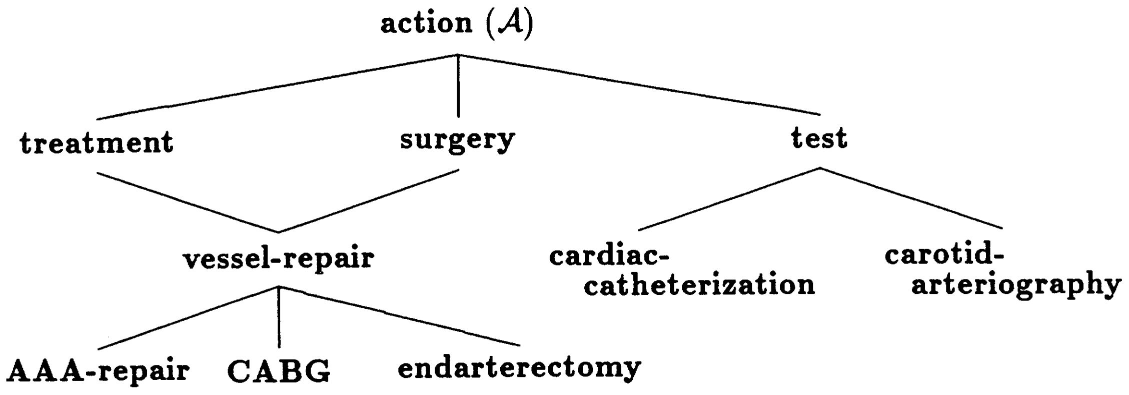


Figure 3.4: A fragment of SUDO-PLANNER's action taxonomy.

3.3 Events

The representational structures denoting events in the knowledge base are very much like those for actions. In fact, actions in SUDO-PLANNER are a special type of event, distinguished because they are controllable by the planner. That is, the root of the event taxonomy, **event** ($\mathcal{E}$), is a superconcept of **action** ($\mathcal{A}$). The primary role of events in the knowledge base is in the representation for the effects of actions (Chapter 5). Events also serve to describe observed conditions in plan class constraints.

The *event variables* representation is identical to that for action variables, described in Section 3.2.2. The extension of an event variable, $\mathcal{X}(ev)$, is equivalent to the cross product of the extensions of its concept and role. The function $EV(obs)$ returns the value taken by an observable event variable instance EV in the observation obs. If EV is unobservable, the function is undefined. By convention, action variables are accessible to the agent without explicit observation. That is why the AV functions are defined in terms of π rather than obs even though action variables are a special case of event variables.

The event variable **value**⟨**agent**⟩ corresponds to the special value node v in QPNs.

3.4 Constraint Language

The unconstrained plan class object denotes Ω, the universal plan class. Partial plans are constructed by adding constraints to Ω or to other existing plan classes. Representations and interpretations for the three types of SUDO-PLANNER constraints (action constraints, action policy constraints, and conditional constraints, introduced in Section 2.5) are presented in sequence below.

3.4.1 Action Constraints

An action constraint on a plan class asserts that an action of a specific type is in its plans. Action types are simply the NIKL concepts appearing in SUDO-PLANNER's action taxonomy. The constraint poster records a sequence of action constraints in an *in-list* describing the action types included in the partial plan.

An in-list consisting only of a collection of types can be ambiguous, however, due to the hierarchical nature of action types. For example, suppose the in-list is [**surgery**,**CABG**]. Because **CABG** is a subconcept of **surgery**, any plan that includes an instance of **CABG** automatically satisfies both constraints. Under this interpretation, we need some other way to express the constraint that a plan include a **CABG** *plus* some other unidentified surgical procedure.

The solution is to associate each separate action constraint with a unique identifier. An in-list of the form $[(a_1, id_1), \ldots, (a_n, id_n)]$ indicates that the plan must include actions of types $a_1, \ldots, a_n$ and that no single action can account for both a_i and a_j unless $id_i = id_j$. If $id_i = id_j$, then there must be a single action that is both an a_i and an a_j. When there is a known type a_k corresponding to the conjunction of a_i and a_j, the two in-list entries can be merged into one for a_k. For example, [(**surgery**, 1), (**CABG**, 1)] is equivalent to [(**CABG**, 1)], but [(**surgery**, 1), (**CABG**, 2)] cannot be simplified. Because NIKL can express conjoined concepts, we need not consider in-lists with more than one action type having the same identifier.[5]

Some action types offer little real constraint on a plan class. For example, asserting that an action of type **drug-therapy** is in the plan isn't saying much because the **dosage** could be zero. By this standard, all plans, medical or otherwise, include a **drug-therapy**. In fact, they also include **surgery**, which (like all SUDO-PLANNER actions) has a boolean **presence** role specifying whether the action is actually performed. Any action constraint whose type has the null action as an instance does not meaningfully restrict the executable plans.

In the cascaded planning architecture of Section 3.1.2, however, such action constraints can make a difference. In general, a plan class produced by the planner need not commit to a policy for every available action. An action constraint restricts that freedom, dictating that the plan class passed to the next module make some commitment about the action. For example, if **drug-therapy** appears on a class's in-list, any subclass passed on to the next module must specify something further about the value of the therapy variable. In this situation, we say that **drug-therapy** is "under consideration" in the plan class.

For example, in the plan graph of Figure 1.5, the partial plan "CABG?" refers to the class of plans where **CABG** is under consideration. It is formed by posting an action constraint on Ω. If our semantics did not supply any force to this constraint, the classifier should have merged the class with Ω and with any other class with such weak action constraints.

To assert that **CABG** should actually be performed, we must post an action

[5]In fact, this property renders identifiers superfluous for plan classes defined exclusively by action constraints. The *ids* are retained, however, because they are indispensable for representing conditional constraints (Section 3.4.3).

constraint for the type **CABG(true)**, the subconcept of **CABG** where the **presence** role is value-restricted to **true**. An action constraint for **CABG(false)** asserts that **CABG** is not performed in the plan.

3.4.2 Action Policy Constraints

An action policy constraint specifies an action variable, an event variable, and a relationship that must hold between them. For the constraint to be operational, the event variable must be observable at execution time. This condition is satisfied automatically when the event variable is also an action variable because the agent knows what actions it has performed.

The monotonicity conditions mentioned in previous chapters are the only action policy constraints employed by SUDO-PLANNER. A monotonicity constraint is written (δ, av, ev), with δ the direction of the relationship and av and ev the action and event variables, respectively. Adding such a constraint to a plan class Π yields the subclass Π', where

$$\Pi' = \{\pi \in \Pi \mid \forall o_1, o_2 \in \mathcal{O},\ \forall EV \in \mathcal{X}(ev).$$
$$EV(o_1) \geq EV(o_2) \Rightarrow \forall AV \in \mathcal{X}(av).\, AV(\pi(o_1))\ R_\delta\ AV(\pi(o_2))\}, \quad (3.2)$$

with $x\ R_+\ y$ defined to hold if either $x \geq y$ or one of the arguments is undefined. R_- is the inverse of R_+, and R_0 the conjunction of R_+ and R_-. In words, a plan π satisfies the constraint if for every action variable AV of type av, the policy for AV is monotonic (of the appropriate polarity) in every event variable of type ev.

Policy constraints where both variables refer to actions are split into two constraints, with av and ev taking on reversed roles. Because the event component of ev is an action, the first inequality in condition (3.2) is replaced by

$$EV(\pi(o_1)) \geq EV(\pi(o_2)).$$

A monotonicity constraint is ill-defined unless the action for av is under consideration in Π.

For example, the action policy constraint

$$(+, \textbf{presence}\langle\textbf{CABG}\rangle, \textbf{extent}\langle\textbf{cardiac-cath-result}\rangle) \tag{3.3}$$

asserts that the CABG policy is an increasing function of the CAD extent as revealed by cardiac catheterization.[6] Because **presence⟨CABG⟩** is a boolean variable, this implies that the plan is a threshold policy.

3.4.3 Conditional Constraints

Conditional constraints are action or policy constraints that have effect only under some observed condition. For example, we could assert that **CABG** should be performed if **catheterization** reveals three-vessel disease (**3vd**) by posting such a constraint. Although SUDO-PLANNER's dominance prover never directly derives results about plan classes defined by conditional constraints, such constraints could be generated if the inputs were conditioned externally.

Syntactically, constraints are conditioned by including an associated event type e_i. Unconditional constraints are implicitly associated with the universal event **event**. Let $e_i : \Pi$ denote a constraint that the plan must be in Π under condition e_i. The semantics of a conditional constraint can be defined in terms of the semantics for the corresponding unconditional constraint.

Let $\pi[e_i] : \mathcal{O} \rightarrow \Omega_{exe}$ be the partial function defined by

$$\pi[e_i](o) \equiv \begin{cases} \pi(o) & \text{if } o \in \mathcal{X}(e_i) \\ \text{undefined} & \text{otherwise,} \end{cases}$$

[6]The variable **extent⟨CAD-event⟩** is not directly observable. Its value is identical to **extent⟨cardiac-cath-result⟩** if and only if catheterization is a perfect test.

and let

$$\Pi[e_i] \equiv \{\pi[e_i] \mid \pi \in \Pi\}.$$

Note that $\pi[\textbf{event}] = \pi$ and $\Pi[\textbf{event}] = \Pi$.

If Π_1 is the plan class we obtain by posting a constraint on Π, then the class obtained by posting a conditional version of that constraint on Π is

$$\{\pi \in \Pi \mid \pi[e_i] \in \Pi_1[e_i]\}.$$

The SUDO-PLANNER event taxonomy induces a relationship among conditional constraints. If e_i subsumes e_j, constraints conditional on e_i are also in effect under e_j. As in Section 3.4.1, identifiers on the in-lists disambiguate the meaning of separate appearances of compatible actions. For example, the plan class

$$e_i : \ [(\textbf{surgery}, id_1)]; \ \ e_j : \ [(\textbf{CABG}, id_2)]$$

must have a **surgery** *in addition* to a **CABG** in event e_j, unless $id_1 = id_2$.

3.5 Computing Subsumption

This section presents the subsumption algorithm in three stages, extending the constraint language to include each type of constraint in turn. The version of Section 3.5.2, accounting for action and policy constraints, is sufficient for expressing all of the plan classes about which SUDO-PLANNER can derive dominance results.

3.5.1 Subsumption with Action Constraints

With only the first type of constraint, a partial plan consists of an in-list of action types. Because each identifier is unique in an in-list (see Section 3.4.1), we can ignore them in this part of the subsumption algorithm.

In general, one plan class subsumes another iff the constraints on the second are at least as strong as those on the first. Because individual action constraints restrict the role of exactly one action in the plan, we can test this subsumption condition action-by-action. However, we cannot tell immediately which actions constrained in the second partial plan correspond to each mentioned in the first. Instead, we must determine whether there is any mapping between the two in-lists such that each action type in the first is matched with a distinct action type in the second that is at least as specific.

For example, suppose the action taxonomy consists of action types $a_1, \ldots, a_6$ such that a_i subsumes a_j iff $i \leq j$. Let $\Pi_1 = [a_1, a_2, a_5]$ and $\Pi_2 = [a_3, a_4, a_6]$. Figure 3.5a illustrates these two classes, with a link between each pair such that the upper action type subsumes the lower. From this diagram, we can tell that Π_1 subsumes Π_2 because for each action in the former it is possible to select a distinct one in the latter connected to it.

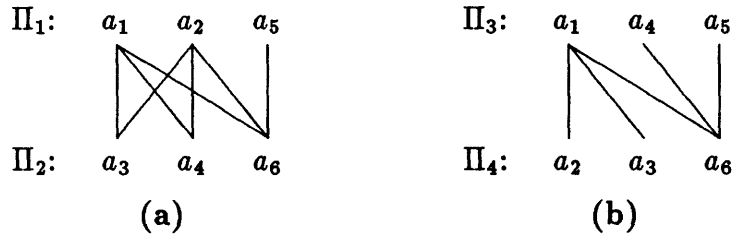


Figure 3.5: Plan class subsumption by bipartite matching. Links indicate that the upper action subsumes the lower. (a) Π_1 subsumes Π_2, but (b) Π_3 does not subsume Π_4.

On the other hand, Figure 3.5b demonstrates that $\Pi_3 = [a_1, a_4, a_5]$ does not subsume $\Pi_4 = [a_2, a_3, a_6]$, because there is no such one-to-one mapping. Action a_6 can be paired with at most one of a_4 and a_5, leaving the other unaccounted for.

To compute subsumption among plan classes Π_1 and Π_2 (represented by their in-lists), we first construct a graph of the sort illustrated in Figure 3.5. An action type in Π_1's in-list is linked to every type in Π_2's that it subsumes. Construction of this graph requires $O(n^2)$ action subsumption computations. This graph is *bipartite* because the vertices are partitioned into two groups such that all edges connect vertices in different groups. A *matching* on a bipartite graph is a collection of unconnected edges. Given a bipartite graph, Π_1 subsumes Π_2 iff there is a matching that uses all of the actions in Π_1's in-list.

We can compute a maximum matching in a bipartite graph in $O(n^{5/2})$ time using the algorithm of Hopcroft and Karp [58]. The subsumption holds iff the cardinality of this matching equals the cardinality of Π_1's in-list. Assuming that action subsumption takes constant time, the algorithm for plan class subsumption is $O(n^{5/2})$.

3.5.2 Subsumption with Policy Constraints

In this section, we extend the partial plan language to include, in addition to the in-list, a collection of monotonicity constraints of the form (δ, av, ev). Because the policy constraints do not interact with the action constraints, the subsumption algorithm is separable into two parts. Π_1 subsumes Π_2 iff both

1. the in-list for Π_1 subsumes that of Π_2 according to the algorithm of Section 3.5.1, and

2. for all monotonicity constraints $(\delta_{i,1}, av_{i,1}, ev_{i,1})$ in the description of Π_1, there exists a (not necessarily distinct) $(\delta_{j,2}, av_{j,2}, ev_{j,2})$ associated with Π_2 such that:

   (a) $\delta_{i,1} = \delta_{j,2}$,

   (b) $av_{j,2}$ subsumes $av_{i,1}$, and

   (c) $ev_{j,2}$ subsumes $ev_{i,1}$.

Note the reversal of subsumption polarity for the variables in monotonicity constraints. For action constraints, it is weaker to assert that a more general action type is included in the plan. Under the definition of monotonic policy constraints (3.2), in contrast, generality in the variables leads to stronger constraints. For example, asserting that **presence⟨vessel-repair⟩** increases in the test result variable is more restrictive than asserting that **presence⟨CABG⟩** does.

Pairwise comparison of monotonicity constraints is sufficient as long as it is not possible for one constraint to be subsumed by a group of others without being subsumed by a single member of the group. This condition would be violated if such a group subsumption were possible for action and event variables. For example, if $\mathcal{X}(av_1) \subseteq \mathcal{X}(av_2) \cup \mathcal{X}(av_3)$ but $\mathcal{X}(av_1) \not\subseteq \mathcal{X}(av_2)$ and $\mathcal{X}(av_1) \not\subseteq \mathcal{X}(av_3)$, pairwise comparison of monotonicity constraints would lead to an incomplete subsumption algorithm. Although this situation can be expressed with NIKL's *covers* relation [66], that facility is not used in the SUDO-PLANNER knowledge base.

Because the mapping of monotonicity constraints need not be one-to-one, the second stage of the algorithm requires only $O(n^2)$ event variable subsumptions. Taking these primitive subsumptions as constant, the overall plan class subsumption complexity remains $O(n^{5/2})$.

3.5.3 Subsumption with Conditionals

Π_1 subsumes Π_2 iff $\Pi_1[e_i]$ subsumes $\Pi_2[e_i]$ for each e_i appearing in an explicit conditional constraint on Π_1. Unfortunately, determining the effective constraints under each condition (the $\Pi[e_i]$) is difficult because of interactions between monotonicity and conditional constraints. A constraint conditional on e_i may influence $\Pi[e_j]$ even if e_i and e_j have no taxonomic relation.

For example, let $e_i =$ **cardiac-cath-result(3vd)**. The conditional constraint $e_i :$ **CABG(true)** asserts that **CABG** is performed if catheterization indicates three-vessel CAD. Suppose we post this constraint in conjunction with the monotonicity constraint that CABG policy be increasing in CAD extent (3.3). Because the monotonicity constraint implies a threshold policy and the conditional constraint tells us that **3vd** is above the threshold, we can infer the CABG policy for values of CAD extent more severe than **3vd**. **CABG(true)** must also be included, for instance, under $e_j =$ **cardiac-cath-result(left-main-disease)**.

The following algorithm ignores this potential interaction and is therefore incomplete. It assumes that the conditional constraints are arranged in a taxonomy according to the conditioning events. The classifier should update this taxonomy as the constraints are posted.

The algorithm constructs a description of the partial plan class $\Pi_1[e_i]$ by merging the in-lists and policy constraints associated with each event subsuming e_i. It merges in-list elements as described in Section 3.4.1 and simply conjoins the policy constraints. For each e_i relevant to Π_1, the subsumption algorithm computes $\Pi_2[e_i]$ by classifying e_i in Π_2's conditional event lattice and proceeding as above. If $\Pi_1[e_i]$ subsumes $\Pi_2[e_i]$ for all e_i, then Π_1 subsumes Π_2.

Despite the separability assumption, the algorithm is sound (all derived subsumptions are valid) because the constraints in Π_2 are individually stronger than those of Π_1. Therefore, any properties of plans holding by virtue of constraint interactions in Π_1 must also hold in Π_2.

The complexity of this algorithm is dominated by subsumption computations on the conditional partial plans; the time needed to construct them can be ignored in asymptotic analysis. In the worst case, the events are linearly arranged so that each constraint is inherited by all successors. The complexity of the subsumption algorithm in this situation is $O(n^{7/2})$.

3.5.4 Extensions

Further work should explore the development of a complete algorithm for plan class subsumption with conditional constraints. In addition, there are several extensions to the constraint language that would provide significantly greater expressive power.

The constraint language cannot express relations among actions in the plan. For example, it would be convenient to link the roles of several actions to assert that they share a common instrument. The language would also be enriched if it permitted expression of some kinds of temporal relations among actions.

Unfortunately, it appears that admitting any interactions or relations among different actions renders the problem combinatorial. The separability of action constraints led to a tractable algorithm because they could be matched action-by-action. The tractability of monotonicity constraints is due to their universality. In contrast, the intractability of nonlinear plan subsumption (see Section 2.7) lies in the addition of an operator for constraining a transitive relation. The relation in this case is interpreted as step ordering; other isomorphic interpretations lead to similar intractability. In extending the expressive power of the constraint language, caution is required to minimize the concomitant degradation of computational feasibility.

Chapter 4

Qualitative Probabilistic Networks

The qualitative probabilistic network formalism is part of SUDO-PLANNER's domain modeling language for representing the relationships among actions and events. This chapter is a self-contained description of the formalism. Chapters 5, 6, and 7 describe in detail how qualitative probabilistic networks support tradeoff formulation in SUDO-PLANNER.

4.1 Introduction

Many knowledge representation schemes, ranging from the various flavors of "causal networks" [100, 113, 153] to qualitative physical models [4] to belief networks [101], model the world as a collection of states, events, or other ontological primitives connected by links that describe their interrelationships. The representations differ widely in the nature of the fundamental objects and in the precision and expressiveness of the relationship links.

Qualitative probabilistic networks (QPNs) occupy a region in representation space where the objects are arbitrary variables, and the relationships are qualitative constraints on the joint probability distribution among them. This area is important for AI research because the relation among variables is often uncertain due to incomplete knowledge or modeling, and because strictly numeric representations are inappropriately precise for many applications. Excess precision leads to knowledge bases applicable in only narrow domains and to knowledge engineering difficulties due to diminished modularity [51, 161].

The qualitative relationships expressible in the QPN formalism are designed to afford robustness yet permit a reasoner to deduce useful properties about optimal assignments to the specially designated decision variables in the network. These "useful properties" are facts that enable a planner to reduce the search space of possible courses of action. The nature of these decision properties and the qualitative relationships leading to them are developed in the body of this chapter.

4.1.1 Motivation

The primary purpose of qualitative probabilistic networks is to support dominance-proving for tradeoff formulation in SUDO-PLANNER. In addition, the analysis of QPNs offers potential benefits in two other areas.

Probabilistic semantics for a common knowledge base construct. Relations similar in intent to those expressible in QPNs have been applied widely in AI knowledge bases without serious attempts at formalization, probabilistic or otherwise. The analysis below suggests how such constructs might be interpreted and in some cases dictates how they *must* be interpreted to justify inferences drawn by associated reasoners.

Qualitative reasoning methods for domains where directions are not guaranteed, and functional relations are not deterministically fixed. Many applications of qualitative reasoning are not faithful to the underlying assumptions behind a "qualitative differential equations" interpretation. Taking an explicit probabilistic approach reveals the possible pitfalls of such violations. This issue is discussed further in Section 4.8.3 below.

4.1.2 Preview of the Chapter

Section 4.2 formally introduces qualitative probabilistic networks, relates them to numeric graphical probabilistic representations, and presents an example from the domain of digitalis therapy. The digitalis example illustrates the use of *qualitative influences*, one of the two qualitative relationship types appearing in qualitative probabilistic networks.

The next four sections elaborate the semantics, properties, and application of qualitative influences. A formal probabilistic definition is motivated and developed in Section 4.3. Section 4.4 describes inference mechanisms that are sound with respect to this definition, and presents an efficient algorithm for answering queries about the qualitative influences holding among arbitrary variables in the network. Section 4.5 considers alternative probabilistic semantics and shows that the definition of Section 4.3 is the weakest validating the inference mechanisms of Section 4.4. Application of these techniques to the digitalis example is the subject of Section 4.6.

The second type of relationship, *qualitative synergy*, is defined, defended, and analyzed in Section 4.7. The section also presents graphical algorithms for reasoning about synergies in QPNs similar to those for qualitative influences. Analysis of the digitalis model enhanced with synergy assertions demonstrates that useful properties of the preferred therapy plan follow from purely qualitative assertions.

Section 4.8 contrasts the qualitative probabilistic network representation with related work in AI, decision theory, and statistics. The relevance of these results to previous qualitative reasoning applications is also discussed. A perspective on the significance of this work is offered in the final section.

4.2 Qualitative Probabilistic Networks

4.2.1 Network Models

A network model is a graph-like structure with nodes that represent variables and edges and hyper-edges that describe relationships among them. In a probabilistic model, the values of variables as well as their interrelationships are uncertain, defined by a probability distribution over the joint value space. Probabilistic network models have attracted much recent attention in AI, for example in Pearl's work on belief networks [101, 103] and related formalisms [14, 77, 134]. The network formalism developed here is accurately viewed as a qualitative abstraction of *influence diagrams* [61], which are belief networks with additional constructs to support decision-making. Some terminology, notation, and even solution concepts (by analogy) are borrowed from Shachter's work on influence diagram evaluation [123, 124].

Formally, a qualitative probabilistic network is a pair $G = (V, Q)$. V is the set of *variables*, or vertices of the graph. Variables are associated with a set of possible values: for example, boolean for propositional event variables, or real intervals for continuous parameters. Unlike most numeric schemes, there is no practical requirement to reformulate the value spaces into discrete, finite sets. Let $X(a)$ denote the domain of variable a. The domain of a tuple of variables is the product space of the individual domains, for example, $X(\langle a, b \rangle) = X(a) \times X(b)$. The tuple will be written as a set when the ordering is insignificant. Subscripted symbols denote values in the domain of a variable.

The variable set V may contain one special variable v, called the *value node*. The relationship between v and the other variables in the network express preferences among states.

It is also useful to distinguish a set $D \subseteq V - \{v\}$ of *decision variables*. A decision-making program takes variables in D to be under its control and therefore focuses on deriving the implications on v of choosing different values for them. The remainder of the variables in the network are random variables not under direct control of the decision maker.

Q is a set of *qualitative relationships* among the variables. Qualitative relationships express constraints on the joint probability distribution over the variables. Unlike the numeric conditional probabilities specified in belief networks and influence diagrams, they are not generally sufficient to determine the exact distribution. In fact, in a purely qualitative network the absolute likelihood of any joint event is completely unconstrained! Nevertheless, the qualitative relationships are carefully designed to justify the deduction of a class of *relative* likelihood conclusions that in turn imply useful decision-making properties. Note that nothing prevents us from building hybrid models combining qualitative relationships with those more precise, although the present work does not pursue that possibility.

There are two types of qualitative relationships in QPNs. *Qualitative influences* describe the direction of the relationship between two variables. *Qualitative synergies* describe interactions among influences. These concepts form the basis of the QPN formalism and are developed in detail below.

4.2.2 Example: The Digitalis Therapy Advisor

The development of QPN concepts is illustrated with a simple causal model taken from Swartout's program for digitalis therapy [141]. The model, shown in Figure 4.1, is a fragment of the knowledge base that Swartout used to re-implement the Digitalis Therapy Advisor [44] via an automatic programmer.

In the figure the circular nodes represent random variables. The rectangular node is a decision variable, in this case denoting the dosage of digitalis (*dig*) administered

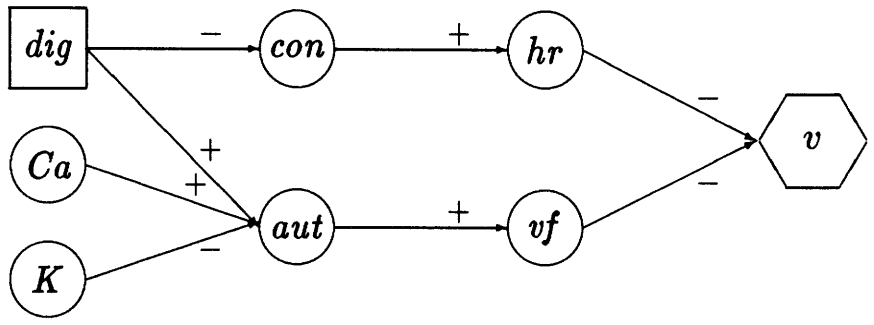


Figure 4.1: Part of the causal model for digitalis therapy. The direction on a link from a to b indicates the effect of an increase in a on b.

to the patient. The value node v is drawn within a hexagon and represents the utility of the outcome to the patient.

Qualitative influences among the variables are indicated by dependence links, annotated with a sign denoting the direction of the relationship.[1] Thus digitalis negatively influences conduction (*con*) and positively influences automaticity (*aut*). The former is the desired effect of the drug, because a decrease in conduction *decreases* the heart rate (*hr*), which is considered beneficial for patients with tachycardia, the population of interest here. The desirability of lower heart rates is represented by the negative influence on the value node, asserting that lower rates increase expected utility. The increase in automaticity is an undesired side-effect of digitalis because this variable is positively related to the probability of ventricular fibrillation (*vf*), a life-threatening cardiac state. Calcium (*Ca*) and potassium (*K*) levels also influence the level of automaticity.

There are no links into the decision variable because the digitalis dosage is considered by the model to be under direct control.

A qualitative encoding of this model is appropriate for the knowledge base of a general digitalis therapy program because a numeric description would require addi-

[1]Discussion of qualitative synergies holding in this example is deferred to Section 4.7.

tional context information or be inaccurate. While the exact probabilistic relationships among these variables vary from patient to patient, the *directions* of the relations are reliably taken as constant. Conclusions drawn from this model are therefore valid for a broad class of patients.

The conclusions we would like our programs to derive from the digitalis model are those taken for granted in the description above. For example, we unthinkingly assumed that the effects of digitalis on conduction and of conduction on heart rate would combine to imply that digitalis reduces the heart rate. Further, because lower heart rates are desirable, digitalis is *therapeutic* along the upper path. Conversely, it is *toxic* along its lower path to the value node. The tradeoff between therapy and toxicity cannot be resolved by mere qualitative influences.

The immediate task of this chapter is to develop a semantics for these qualitative influences that justifies the kinds of inferences we require while providing the maximum robustness. In the sections below, I provide such a semantics in terms of a probabilistic definition for qualitative influences. In Section 4.5 we will see that this definition is the weakest in a reasonable class that justifies the conclusions mentioned above.

4.3 Qualitative Influences Defined

4.3.1 Influence Notation

The qualitative links in the digitalis model above can be represented formally as edges in the graph annotated by sign. Let $S^\delta(a, b, G)$ denote the assertion that a qualitative influence of a on b in direction (that is, sign) δ holds in graph $G = (V, Q)$.

Definition 4.1 (Qualitative influence edges) $S^\delta(a, b, G) \equiv (a, b, \delta) \in Q$, *for all* $\delta \in \{+, -, 0, ?\}$.

By convention, S^0 links are left implicit in graphical displays of the network. They would also typically be left implicit—inferable via a closed-world assumption—in data structures representing qualitative networks. The *pred* function selects only the predecessors exerting nonzero influence on a variable.

Definition 4.2 (predecessors)

1. *$Pred_G(a,b)$ iff $(a,b,\delta) \in Q$, for some $\delta \in \{+,-,?\}$.*

2. *$pred_G(b) \equiv \{a \mid Pred_G(a,b)\}$.*

3. *$pred^*_G(b) \equiv \{a \mid pred^*_G(a,b)\}$, where $Pred^*_G$ is the transitive closure of the $Pred_G$ relation.*

Note that for all $d \in D$, $pred_G(d) = \emptyset$. The subscript G is omitted when its value is clear from context.

4.3.2 Probabilistic Semantics for Qualitative Influences

Consider two variables, a and b. Informally, when a and b denote boolean events, a qualitative influence is a statement of the form "a makes b more (or less) likely." This binary case is easy to capture in a probabilistic assertion. Let A and $\bar{A}$ denote the assertions $a =$ **true** and $a =$ **false**, respectively, and similarly, B and $\bar{B}$.

Definition 4.3 (binary S^+) *We say "a positively influences b" (stochastically) and write $S^+(a,b,G)$, iff for all $x \in X(pred_G(b) - \{a\})$ such that x is consistent with both A and $\bar{A}$,*[2]

$$\Pr(B|Ax) \geq \Pr(B|\bar{A}x). \tag{4.1}$$

[2]We can safely ignore cases where the conditional probabilities are undefined because these are impossible contexts.

In Definition 4.3, the *context* x ranges over all consistent assignments to the variables other than a that influence b. (Henceforth, x in a formula will be understood as universally quantified over the values of predecessor variables.) We need to include the *ceteris paribus* condition here and in the definitions below so that qualitative relations will be applicable in situations where x is partially or totally known. If we had stated the S^+ definition in marginal terms ("on average, a positively influences b"), it would not be valid to apply it in specific contexts.

Because of this context dependence, S^+ holds in a particular network; programs that alter the structure of the network may exhibit non-monotonicity in S^+ relative to its first two arguments [45]. In the following I omit the third argument only when the context is unambiguous or inessential.

Conditions analogous to (4.1) and those following define negative and zero influences; I omit them for brevity. S^0, an assertion that (4.1) holds with equality, is the familiar concept of conditional independence of a and b given b's direct influences. We could rule out the independent case with strict versions of S^+ and S^-, but discussion is limited to non-strict influences in this work.

$S^?$ always holds. It is included explicitly only so that we can represent S^0 implicitly in the lack of an influence assertion.

For binary variables, Bayes's rule implies that (4.1) is equivalent to

$$\Pr(A|Bx) \geq \Pr(A|\bar{B}x). \tag{4.2}$$

In the terminology of Bayesian revision, (4.1) is a condition on *posteriors*, while (4.2) is a condition on *likelihoods*. Notice that $S^+(a,b)$ is simply an assertion that the *likelihood ratio* is greater than or equal to unity.

Formalizing the intuitive idea that "higher values of a make higher values of b more likely" is not quite as straightforward when a and b take on more than two values. An obvious prerequisite for such statements is some interpretation of "higher."

Therefore, we require that each random variable appearing in an S^+ or S^- assertion be associated with an order $\geq$ on its values. This relation has the usual interpretation for numeric variables such as "potassium concentration"; for variables like "automaticity," a measurement scale and ordering relation must be contrived.

The more troublesome part of defining positive influences is specifying what it means to "make higher values of b more likely." Intuitively, we want a statement that the probability distribution for b shifts toward higher values as a increases. To make such a statement, we need an ordering on cumulative probability density functions (CDFs) F_b over b that captures the notion of "higher."

However, probability distributions cannot be straightforwardly ordered according to the size of the random variable. Different rankings result from comparison of distributions by median, mean, or mean-log, for example. We require an ordering that is robust to changes of these measures because the random variables need be described by merely *ordinal* scales [71]. An assertion that calcium concentration positively influences automaticity should hold whether calcium is measured on an absolute or logarithmic scale, and regardless of how we measure automaticity.

An ordering criterion with the robustness we desire is *first-order stochastic dominance* (FSD) [162]. FSD holds for CDFs F_b and F'_b iff for any given value b_0 of b, the probability of obtaining b_0 or less is smaller for F_b than for F'_b. That is, F_b FSD F'_b iff

$$\forall b_0 \; F_b(b_0) \leq F'_b(b_0). \tag{4.3}$$

A necessary and sufficient condition for (4.3) is that for all monotonically increasing (that is, order-preserving) functions ϕ,

$$\int \phi(b_0)dF_b(b_0) \geq \int \phi(b_0)dF'_b(b_0). \tag{4.4}$$

That is, the mean of F_b is greater than the mean of F'_b for *any* monotonic transform of b. For further discussion and a proof that (4.3) is equivalent to (4.4), see [33].

We are now ready to define qualitative influences.

Definition 4.4 (S^+) *Let $F_b(\cdot|a_i x)$ be the CDF for b given $a = a_i$ and context x. Then $S^+(a,b)$ iff*

$$\forall a_1, a_2 .\ a_1 \geq a_2 \Rightarrow F_b(\cdot|a_1 x) \ \text{FSD} \ F_b(\cdot|a_2 x). \tag{4.5}$$

Definition 4.4 is a generalization of Definition 4.3 under the convention that **true** > **false** for binary events.

Like (4.1), (4.5) is a condition on posteriors. To achieve the strength of (4.5), a definition of S^+ in terms of likelihoods must imply FSD of the posteriors for *any prior distribution* F_b. That is, we allow that there may be a context x inducing any distribution on b. Milgrom [90] proves that the following condition is necessary and sufficient for (4.5) to hold for any $F_b(\cdot|x)$.

$$\forall a_1, a_2, b_1, b_2 .\ a_1 \geq a_2 \wedge b_1 \geq b_2 \Rightarrow \frac{f_a(a_1|b_1 x)}{f_a(a_1|b_2 x)} \geq \frac{f_a(a_2|b_1 x)}{f_a(a_2|b_2 x)}. \tag{4.6}$$

In (4.6), $f_a(\cdot|b_i x)$ is the probability density function for a given b_i and x.

This condition is known in statistics as the Monotone Likelihood Ratio Property (MLRP) [3]. The necessity of MLRP for (4.5) is established by the special case of dichotomous events. That (4.6) is a generalization of (4.2) is more clearly seen by rewriting the latter as

$$\frac{\Pr(A|Bx)}{\Pr(A|\bar{B}x)} \geq 1 \geq \frac{\Pr(\bar{A}|Bx)}{\Pr(\bar{A}|\bar{B}x)}. \tag{4.7}$$

For a demonstration of the sufficiency of MLRP, see Milgrom [90].

It is convenient to adopt special notation for influences on the value node v. The value node is related to its predecessors by a *utility function* $u : X(\mathit{pred}(v)) \rightarrow \Re$ [69].

Definition 4.5 (U^+) *The variable a positively influences utility, $U^+(a)$, iff*

$$\forall a_1, a_2 .\ a_1 \geq a_2 \Rightarrow u(a_1, x) \geq u(a_2, x). \tag{4.8}$$

The definition of $U^+(a)$ is a special case of $S^+(a, v)$ taking into account the deterministic relation (a degenerate probability distribution) between v and its predecessors in the network.

4.4 Indirect Relationships

Edges in a graph of influence links constrain the direct relationship between pairs of variables. Our next step is to design inference mechanisms to derive the indirect relationships that follow from patterns of local influences.

First, let us define the *canonical direction* between two variables to be the strongest qualitative influence derivable from those explicitly appearing in Q. The canonical direction can be easily computed from Q by preferring an explicit 0 to the other δs (which are always consistent with 0 because the conditions are non-strict), preferring + or − to ?, and replacing the combination of + and − with 0.

Definition 4.6 (dir) *Let $S = \{\delta \mid (a, b, \delta) \in Q\}$. The canonical direction of influence of a on b, $dir(a, b, G)$, is given by*

$$dir(a,b,G) = \begin{cases} 0 & \textit{if } S = \emptyset,\ 0 \in S, \textit{ or } \{+,-\} \subseteq S \\ + & \textit{if } S = \{+\} \textit{ or } S = \{+,?\} \\ - & \textit{if } S = \{-\} \textit{ or } S = \{-,?\} \\ ? & \textit{otherwise.} \end{cases}$$

If $dir(a, b) = ?$ then a and b are dependent in an unknown, varying, or context-dependent direction.

4.4.1 Probabilistic Dependence in Graph Representations

Definition 4.7 (dep) *The dependency graph, $dep(G)$, of $G = (V, Q)$ is*

$$dep(G) = (V, E), \quad \textit{where } (a, b) \in E \textit{ iff } dir(a, b, G) \neq 0$$

The dependency graph simply encodes the pattern of nonzero influences without distinguishing the signs on the links. Pearl [104] has characterized the expressiveness of these graphs with respect to the dependency structure of probability distributions. Some results of this work and terminology developed there will prove useful in analyzing the properties of QPNs.

In a directed acyclic graph representation, two variables are conditionally independent given any set of other variables that *d-separates* them in the graph.

Definition 4.8 (d-separation, Pearl [101]) *Two variables a and b are* d-separated *by a set of variables S in a directed acyclic graph iff for every* undirected *path from a to b either:*

1. *there is a node $s \in S$ on the path with at least one of the incident edges leading out of s, or*

2. *there is a node t on the path with both incident edges leading in, and neither t nor any of its successors are in S.*

The concept of d-separation is illustrated by the network of Figure 4.2.

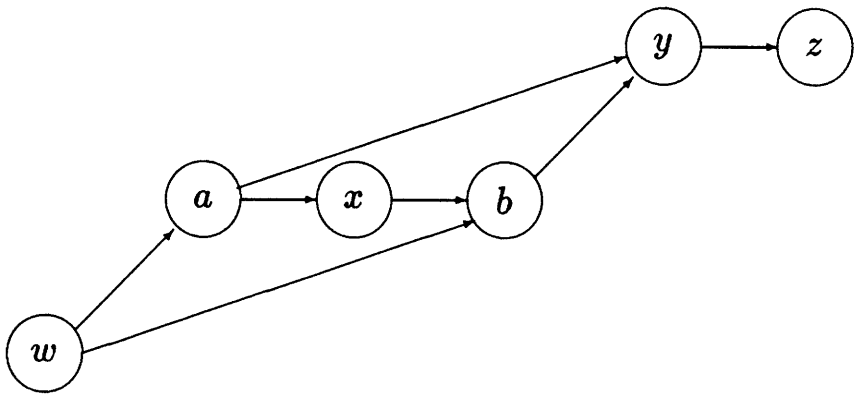


Figure 4.2: Variables a and b are *d-separated* by $\{w, x\}$ but by no other subset of $\{w, x, y, z\}$.

The following implication of Definition 4.8 is useful in justifying the inference rules for QPNs presented below.

Lemma 4.9 *If $b \notin pred^*_G(a)$ then a and b are d-separated in $dep(G)$ by any S such that $pred_G(b) \subseteq S \subseteq pred^*_G(b)$.*

Proofs of this lemma and subsequent results appear in Appendix B.

Taking $S = pred_G(b)$, this result is the basis for our closed-world assumption that $dir(a, b, G) = 0$ if there are no explicit influences in Q. If in addition there are no directed paths from b to a, we adopt the default influence $S^0(a, b)$. In Pearl's terminology, this assumption is valid when $dep(G)$ is an *I-map*—a graph for which all d-separations are true conditional independencies.

4.4.2 Network Transformations

We answer queries about relations among separated variables in a QPN by transforming the graph into one where the variables of interest are related directly. The method is based on Shachter's algorithm for evaluating numeric influence diagrams [123] by repeated reductions and arc reversals. Each manipulation preserves the probabilistic relationships—qualitative in our case—holding among variables in the possibly smaller set V. Shachter [124] shows that there is a sequence of manipulations to answer a query about the relationship among any subset of variables in the network.

The two basic network transformation operators are reduction (*red*) and reversal (*rev*). The reduced network $red(b, G)$ is the qualitative probabilistic network obtained by splicing variable b out of G and adjusting qualitative influences as dictated by Theorem 4.11 below. Network $rev(a, b, G)$ is obtained from G by replacing $(a, b, \delta) \in Q$ with the influence (b, a, δ) and updating other influences as specified in Theorem 4.12.

The exact reduction and reversal procedures are described in Section 4.4.3 below. Both the *red* and *rev* operations preserve essential properties of the networks. Let

$G' = (V', Q')$ be the result of one of these operations. Then:

- $dep(G')$ is acyclic.
- $dep(G')$ is an I-map.

4.4.3 Variable Reductions

It is straightforward to demonstrate for the binary case that, in the absence of direct links from a to b, $S^+(a,b,G) \wedge S^+(b,c,G) \Rightarrow S^+(a,c,red(b,G))$. The ability to perform inference across influence chains is an essential property of a qualitative algebra. From the digitalis model, for example, we would like to deduce that increasing the dose of digitalis decreases the heart rate but increases the likelihood of ventricular fibrillation. Indeed, most programs with models like this would make such an inference. Fortunately, the definition offered above for S^+ implies transitivity for multi-valued as well as binary variables.

Theorem 4.10

$$S^{\delta_1}(a,b,G) \wedge S^{\delta_2}(b,c,G) \wedge S^0(a,c,G) \Rightarrow S^{\delta_1 \otimes \delta_2}(a,c,red(b,G)),$$

where $\delta_i \in \{+,-,0,?\}$ *and* $\otimes$ *denotes sign multiplication, described by Table 4.1.*

$\otimes$	+	−	0	?
+	+	−	0	?
−	−	+	0	?
0	0	0	0	0
?	?	?	0	?

$\oplus$	+	−	0	?
+	+	?	+	?
−	?	−	−	?
0	+	−	0	?
?	?	?	?	?

Table 4.1: The $\otimes$ operator for combining influence chains and the $\oplus$ operator for combining parallel influences. For example, $+ \otimes - = -$.

Application of Theorem 4.10 requires that no direct influences exist between a and c. A more general specification of the result of variable reduction is the following:

Theorem 4.11

$$S^{\delta_1}(a,b,G) \wedge S^{\delta_2}(b,c,G) \wedge S^{\delta_3}(a,c,G) \Rightarrow S^{(\delta_1 \otimes \delta_2) \oplus \delta_3}(a,c,red(b,G)), \tag{4.9}$$

where $\oplus$ denotes sign addition, also described in Table 4.1.

Theorem 4.10 is really a corollary of Theorem 4.11 with $\delta_3 = 0$, the identity element for $\oplus$.

Using this result, we can reduce any variables from the network by computing the new direction for each pair of immediate predecessor and successor variables. To find the qualitative influence of a on b given a set of variables W for any $a \in pred^*(b)$ and $W \subseteq pred^*(b)$ we need only splice out all other variables in the network. Because each application of reduction rule (4.9) reduces the number of influence edges (including zeroes) in the network by one, the complexity of this procedure is $O(|V|^2)$.[3] This contrasts with the corresponding problem for numeric probabilistic networks, which is NP-hard [15]. Some sample reductions are displayed in Table 4.2.

4.4.4 Influence Reversals

The procedure developed above is valid when $a \in pred^*(b)$ and $W \subseteq pred^*(b)$. Sometimes, however, we may be interested in the qualitative influence of a on b when the paths in the network run in the other direction. In such cases we need to perform one or more reversals in the network before or after applying the methods of the previous section.

In reversing a qualitative influence link, we must preserve the essential properties mentioned in Section 4.4.2 above. To ensure acyclicity, we can reverse the influence from a to b only if there are no other directed paths between them. Reversal is also

[3]In the worst case, this algorithm requires $\Omega(|V|^2)$ operations for reducing a single variable. The overall complexity is therefore $\Theta(|V|^2)$.

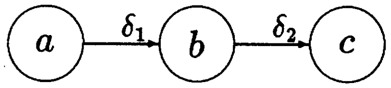
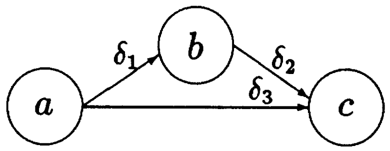
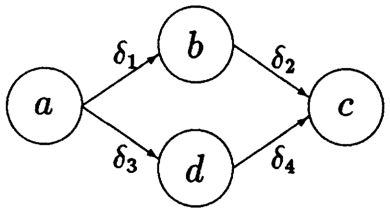
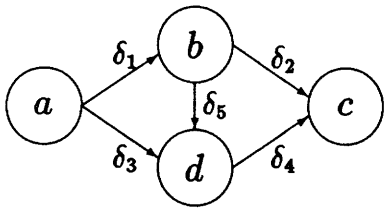
	original network	$dir(a,c)$
(1)		$\delta_1 \otimes \delta_2$
(2)		$(\delta_1 \otimes \delta_2) \oplus \delta_3$
(3)		$(\delta_1 \otimes \delta_2) \oplus (\delta_3 \otimes \delta_4)$
(4)	 a δ_1 b δ_2 δ_5 c δ_3 d δ_4	$[\delta_1 \otimes (\delta_2 \oplus (\delta_5 \otimes \delta_4))] \oplus (\delta_3 \otimes \delta_4)$

Table 4.2: Some sample reductions. The right column contains the expression for $dir(a,c)$ in the network obtained by removing nodes between a and c. Fragments (1) and (2) correspond to the situations of Theorems 4.10 and 4.11, respectively.

precluded if a is a decision variable. To guarantee that $G' = rev(a, b, G)$ is an I-map, each must gain the other's predecessors:

$$pred_{G'}(a) = pred_{G'}(b) = pred_G(a) \cup pred_G(b).$$

The definition for $S^\delta(a, b)$ (4.4) explicitly refers to the predecessors of b. Therefore, when the predecessor structure changes we need to recompute the influences that may be affected. The following result describes the influences holding after reversal.

Theorem 4.12 *Let $G' = rev(a, b, G)$. G' inherits all the qualitative influences of G except:*

1. *$dir(a, b, G')$ is undefined.*
2. *$dir(b, a, G') = dir(a, b, G)$.*
3. *$\forall w \in pred_{G'}(b)$,*
$$dir(w, b, G') = [dir(w, a, G) \otimes dir(a, b, G)] \oplus dir(w, b, G).$$
4. *$\forall w \in pred_{G'}(a) - \{b\}$,*
$$\begin{aligned} dir(w, a, G') &= \begin{cases} dir(w, a, G) & \textit{if } dir(w, b, G) = 0 \\ ? & \textit{otherwise} \end{cases} \\ &= dir(w, a, G) \oplus (dir(w, b, G) \otimes ?). \end{aligned}$$

This transformation is illustrated in Figure 4.3.

Some information is lost in the process of reversing influences. For example, let $G'' = rev(a, b, rev(a, b, G))$, the network obtained by reversing an influence then reversing it again. Application of Theorem 4.12 twice yields the result depicted in Figure 4.4. Although the link from a to b is correct, the reversal process weakens the other links. It may be advisable to store information about the state of influences before reversals so that it is recoverable for subsequent reasoning with the original relationships.

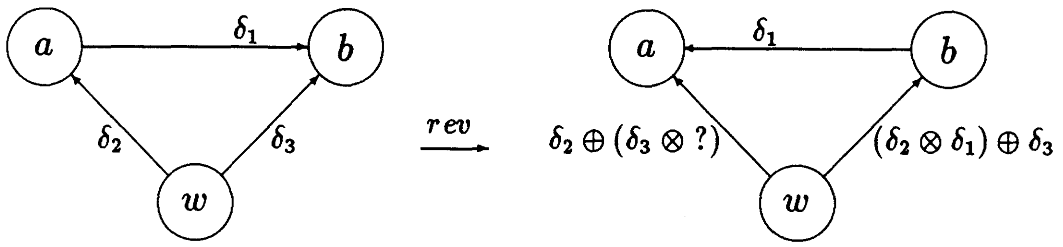

a
δ_1
b
δ_2
δ_3
w
rev
a
δ_1
b
$\delta_2 \oplus (\delta_3 \otimes ?)$
$(\delta_2 \otimes \delta_1) \oplus \delta_3$
w

Figure 4.3: Influence reversal.

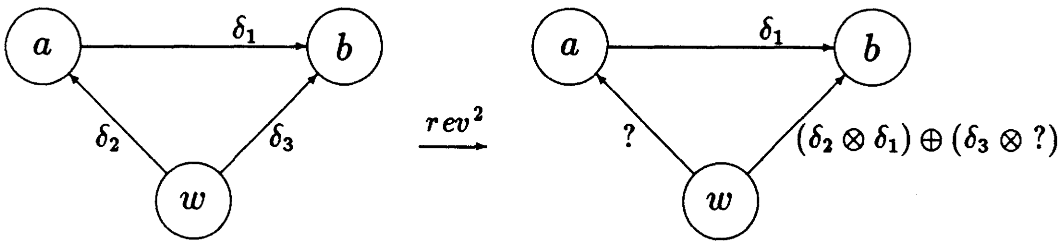


Figure 4.4: Information lost in a double reversal of the influence from a to b.

4.5 Weaker Conditions

4.5.1 Posterior Conditions

The preceding sections establish that the FSD condition for S^+ (Definition 4.4) is sufficient to support essential inferences such as the chaining of influences. In this section I present some simple desiderata for a qualitative influence definition that entail the *necessity* of FSD for these properties. I start by specifying the form such definitions must take. To capture the intent of "higher values of a make higher values of b more likely" in a probabilistic semantics, it seems reasonable to restrict attention to conditions on the posterior distribution of b for increasing values of a. Therefore, I postulate that a definition of $S^+(a, b)$ must be of the form

$$\forall a_1, a_2.\ a_1 \geq a_2 \Rightarrow F_b(\cdot|a_1x)\ R\ F_b(\cdot|a_2x), \tag{4.10}$$

where R is some relation on CDFs. This condition is exactly (4.5) with FSD replaced by the more abstract relation.

There are two basic desiderata that severely restrict the possible Rs. First, S^+ must satisfy Theorem 4.10. Without the ability to chain inferences, the qualitative influence formalism has little computational value. Second, the condition must be a generalization of the original specification of S^+ for dichotomous variables (Definition 4.3). With only two possible values this appears to be a minimal monotonicity condition. These criteria lead to a sharp conclusion.

Theorem 4.13 *Let $S^+(a,b)$ be defined by (4.10). Given the following conditions:*

1. *Theorem 4.10*
2. *For binary b, $a_1 \geq a_2$, and x,*

$$F_b(\cdot|a_1x)\ R\ F_b(\cdot|a_2x) \Leftrightarrow \Pr(B|a_1x) \geq \Pr(B|a_2x)$$

the weakest R is FSD.

The force of this result is weakened somewhat by the *a priori* restriction of definitions to those having the form of (4.10). Many statistical concepts of directional relation (based on correlation or joint expectations, for example) do not fit (4.10) yet appear to be plausible candidates for a definition of qualitative influence. *Quadrant dependence* [78] holds between a and b when[4]

$$\forall a_1, a_2.\ a_1 \geq a_2 \Rightarrow F_b(\cdot|a \leq a_1)\ \text{FSD}\ F_b(\cdot|a \leq a_2). \tag{4.11}$$

Lehmann proves that quadrant dependence is necessary but not sufficient for *regression dependence*, which is his terminology for (4.5) without the quantification over

[4]This is actually the condition Lehmann proposes as a strengthening of quadrant dependence. The basic quadrant dependence fixes a_1 at a's maximal value.

contexts x. As quadrant dependence is weaker, yet still exhibits transitivity,[5] it seems to be an attractive alternative to regression dependence. To justify our choice of the latter, we must consider the decision-making implications of probabilistic models.

4.5.2 Decision-Making with Qualitative Influences

The prime motivation for adopting a probabilistic semantics is so that the behavior of our programs can be justified by Bayesian decision theory [121]. A decision of d_1 over d_2 (that is, such a choice of assignments to decision variables) is valid with respect to a QPN if the network entails greater expected utility for the former. The most useful distinctions to make in designing a qualitative representation are those that will support inferences about properties of the valid decisions.

For example, if $U^+(a)$ and there are no *indirect* paths from decision variable a to the value node, then a choice of a_1 over a_2 is valid iff $a_1 \geq a_2$, by the definition of U^+ (4.5).[6] Decision-making power is enhanced if we can deduce new influences on utility from chains of influences in the network. Our definition of qualitative influence is necessary as well as sufficient for such inferences.

Theorem 4.14 *Suppose $U^{\delta_2}(b, G)$ and $U^0(a, G)$. A necessary and sufficient condition for $U^{\delta_1 \otimes \delta_2}(a, red(b, G))$ is $S^{\delta_1}(a, b, G)$ as in Definition 4.4.*

Figure 4.5 depicts this situation with $\delta_2 = +$.

Theorem 4.14 demonstrates that while conditions weaker than S^+, such as quadrant dependence, may be sufficient for propagating influences across chains, they are not adequate to justify *decisions* across chains. For choosing among alternatives, the

[5]For transitivity we need to quantify over contexts in (4.11). The proof parallels that for Theorem 4.10.

[6]The existence of other paths from a to utility would leave open the possibility that the net influence of a is negative. For example, we could summarize the therapeutic effect of digitalis through conduction and heart rate as a direct positive influence. But this might be outweighed by the indirect negative influence of digitalis via automaticity.

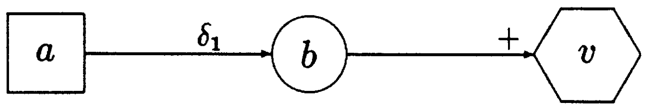


Figure 4.5: Chaining utility influences. The influence $\delta_1 = +$ in G is necessary and sufficient for $U^+(a, red(b, G))$.

relevant parameter is the utility function evaluated at a point; utilities conditioned on intervals of the decision variable (as in quadrant dependence) do not have the same decision-making import.

4.6 Back to the Digitalis Model

To summarize the discussion of qualitative influences thus far, let us return to the digitalis example presented in Section 4.2.2. We are interested in computing the effect of the decision variable, *dig*, on utility. The network of Figure 4.1 reduces to the one depicted in Figure 4.6a, which further reduces to that of 4.6b.

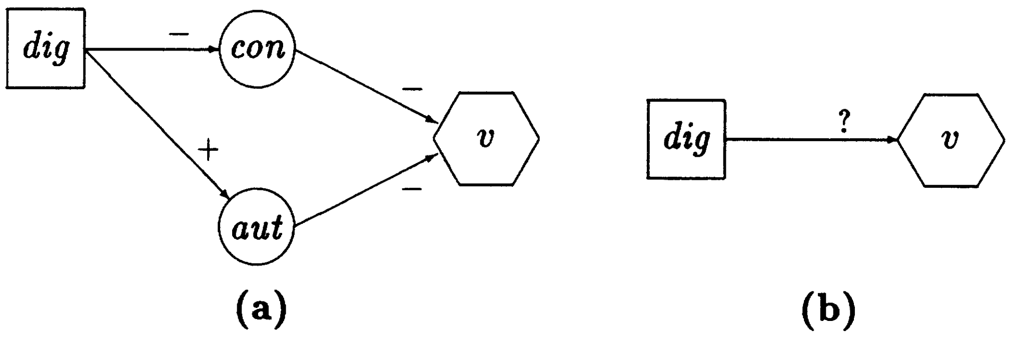


Figure 4.6: Reduction of the digitalis model. (a) Digitalis is therapeutic in its effect on conduction but toxic via the influence on automaticity. (b) The overall effect of digitalis cannot be resolved with qualitative influences.

The result, not surprisingly, is ambiguous. Purely qualitative influences are too weak to determine optimal decisions in the presence of true tradeoffs. Nevertheless, the QPN is sufficient to determine some influences (for example, *Ca* on *vf*), and uncovers the source of indeterminacy in others.

In the next section, a second type of qualitative relationship is introduced: qualitative synergy. Synergies complement influences by providing constraint on the interactions among probabilistic influences. Although synergies cannot resolve the tradeoff of Figure 4.6b, they can provide useful facts about the relation of the optimal digitalis dosage to other variables in the model.

4.7 Qualitative Synergy

Swartout's XPLAIN knowledge base includes the "domain principle" that if a state variable acts synergistically with the drug to induce toxicity, then smaller doses should be given for higher observed values of the variable [141]. This fact could be derived by a *domain-independent* inference procedure given a suitable definition for qualitative synergy. Two variables synergistically influence a third if their joint influence is greater (in the sense of FSD) than separate, statistically independent influences. In the digitalis example, we need to assert that digitalis acts at least independently with Ca and K deviations in increasing automaticity. For the desired result, we also need the fact that heart rate and ventricular fibrillation are synergistic in their influence on utility. (This synergy is due to our indifference to heart rate—indeed it is undefined—for patients in fibrillation. The relation of this indifference to synergy is clarified in Section 4.7.6 below.)

Figure 4.7 illustrates the QPN for digitalis enhanced by synergy assertions. Potassium (K) is omitted for simplicity; its implications are analogous (with sign reversal) to those for calcium. Qualitative synergies are indicated by a boxed sign with multiple inputs and a single output. The input variables are synergistic in the designated

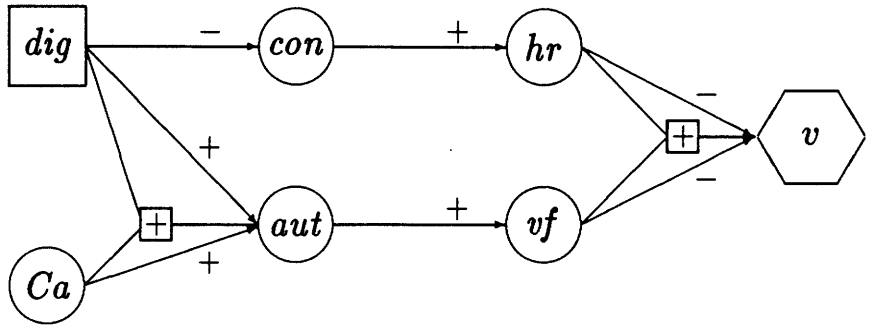


Figure 4.7: The digitalis model with synergy. A boxed sign indicates that the inputs are qualitatively synergistic in their influence on the output.

direction in their influence on the output variable.

4.7.1 Synergy Notation

Qualitative synergies are the second type of qualitative relationship represented in Q for a QPN $G = (V, Q)$. As qualitative influences are directed edges augmented by sign, qualitative synergies are directed hyper-edges with a sign label. A qualitative synergy assertion that the variables in $T \subset V$ are synergistic in direction δ on variable w is written $Y^{\delta}(T, w, G)$.

Definition 4.15 (Qualitative synergy hyper-edges) $Y^{\delta}(T,w,G) \equiv (T,w,\delta) \in Q$.

4.7.2 Qualitative Synergy Defined

A formal definition of qualitative synergy must capture the informal intuition expressed above that the "joint influence is greater than separate statistically independent influences." This will be the case when the effect of varying one variable is enhanced by simultaneous variation of the other.

The joint influence of two variables a and b on a third, c, is expressed by the conditional cumulative distribution for c, $F_c(\cdot|ab)$. To compare magnitude of "influence," we need some reference points. One way to measure a difference in influence is to take the difference of two conditional CDFs. Two variables are synergistic if the difference associated with raising one is greater (in the sense of FSD) for higher values of the second.

Definition 4.16 (qualitative synergy, Y^δ) *Variables a and b are synergistic on c in network G, written $Y^+(\{a,b\},c,G)$ iff*

$\forall a_1, a_2, b_1, b_2, c_0.\ a_1 \geq a_2 \wedge b_1 \geq b_2 \Rightarrow$

$$F_c(c_0|a_1b_1x) - F_c(c_0|a_2b_1x) \leq F_c(c_0|a_1b_2x) - F_c(c_0|a_2b_2x). \tag{4.12}$$

Replacing $\leq$ in condition (4.12) by $\geq$ or $=$ defines sub-synergy or zero synergy (Y^- and Y^0), respectively. If the variable set T in $Y^\delta(T,w,G)$ contains more than two elements, the condition above holds for all pairs *of variables in T.*

As usual, x ranges over assignments to the other predecessors of c.

The inequality (4.12) quantified over c_0 can be viewed as stochastic dominance of the respective distributions of CDF differences. The condition means that raising a from a_2 to a_1 has a greater effect for higher values of b. Note that the inequality is symmetric in a and b.

If $S^0(a,c)$, then $Y^0(\{a,w\},c)$ follows immediately for any variable $w \in \mathit{pred}^*(c)$ because of conditional independence. With conditional independence, $F_c(\cdot|a_1wx) = F_c(\cdot|a_2wx)$ for all w and x, therefore both sides of equation (4.12) are zero.

Lacking an explicit synergy assertion for two or more variables that are predecessors of another, the prudent closed-world assumption is $Y^?$: no constraint on their interaction.[7] Although it is reasonable to assume S^0 in the absence of knowledge to

[7]In the examples of this section, all synergies are specified explicitly.

the contrary, in this case, the variables are tied by a common immediate successor. They are *not* d-separated by this successor, and interactions in situations with this pattern are quite common.

Fortunately, there are several prototypical patterns of systematic interaction that might alleviate the burden of specifying qualitative synergies. One that has attracted some interest in the literature on numeric probabilistic networks is the "noisy OR gate" model proposed by Pearl [103, Chapter 4].

In the noisy OR model, the binary-valued predecessors of a binary "effect" variable are considered separate possible causes of the effect. Each "cause" variable is associated with a parameter p_i representing the probability of the effect given that this variable is true and all other predecessors are false. We can compute the rest of the conditional probabilities for y under the assumption that the "inhibiting events" that prevent Y given each Z_i are independent. For effect variable y with predecessors $z_1, \ldots, z_n$, the conditional probabilities are:

$$\Pr(Y|z_1 \ldots z_n) = 1 - \prod_{\{i|Z_i\}} (1 - p_i). \tag{4.13}$$

Regardless of the magnitudes of the p_is, the noisy OR model entails sub-synergy, Y^-. To see this, consider the Y^- condition ((4.12) with the inequality reversed) for the special case of binary variables. $Y^-(\{z_j, z_k\}, y)$ iff:

$$\forall x \in X(\{z_i | i \neq j \wedge i \neq k\})$$
$$\Pr(Y|Z_j Z_k x) - \Pr(Y|\bar{Z}_j Z_k x) \leq \Pr(Y|Z_j \bar{Z}_k x) - \Pr(Y|\bar{Z}_j \bar{Z}_k x). \tag{4.14}$$

Let

$$p_x = \prod_{Z(x)} (1 - p_i), \text{ where } Z(x) = \{i | Z_i \text{ in assignment } x\}.$$

Then from the noisy OR model (4.13),

$$\Pr(Y|Z_j Z_k x) - \Pr(Y|\bar{Z}_j Z_k x) = p_x p_j (1 - p_k), \text{ and} \tag{4.15}$$

$$\Pr(Y|Z_j \bar{Z}_k x) - \Pr(Y|\bar{Z}_j \bar{Z}_k x) = p_x p_j. \tag{4.16}$$

Because $0 \le p_k \le 1$, the expression in (4.15) is no greater than that of (4.16), satisfying the binary Y^- condition (4.14). Figure 4.8 illustrates the relation between a numeric probabilistic network using the noisy OR model and its corresponding QPN.

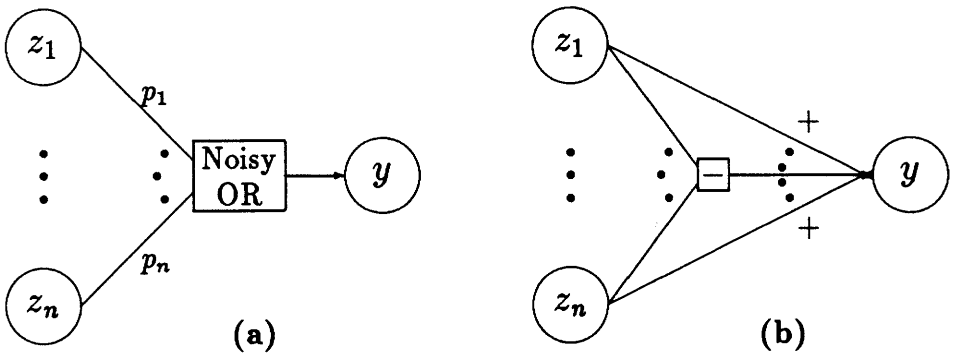


Figure 4.8: (a) The "noisy OR" model, and (b) its corresponding qualitative abstraction.

It is also easy to verify that Henrion's generalizations of the noisy OR model [52] entail Y^-. Intuitively, a noisy OR is sub-synergistic because, as with deterministic OR gates, raising an input has less effect when other inputs are already raised. In contrast, a model based on a probabilistic generalization of "gating conditions" (see Rieger and Grinberg [113]) would be synergistic because an increase in one variable enables the effect of the other. More generally, we should expect non-"?" synergy results from canonical models because any representation that specifies an n-way influence in terms of $O(n)$ parameters must employ some systematic assumption about interactions.[8]

4.7.3 Supermodularity, Y^δ, and Monotone Decisions

The Y^δ definition relates closely to the concept of *supermodular* functions [115, 146].

[8]Dempster's rule of combination is also sub-synergistic under an analogous definition of synergy in terms of belief functions [126]. A demonstration of this requires further assumptions regarding how to interpret conditioning as evidence combination.

Definition 4.17 (supermodularity, Ross [115]) *A function g such that, for all $a_1 \geq a_2$ and $b_1 \geq b_2$:*

$$g(a_1, b_1) + g(a_2, b_2) \geq g(a_1, b_2) + g(a_2, b_1) \tag{4.17}$$

is called supermodular. *If (4.17) holds with equality, then g is* modular, *and if the inequality is reversed, g is* submodular.

The most important property of supermodular functions, from our perspective, is that they imply *monotone decisions.* Let the function $a_g(b)$ choose the value of a that maximizes g for the given b.

$$a_g(b) = \arg\max_a g(a, b).$$

It can be shown that $a_g(\cdot)$ increases monotonically in b if g is supermodular (see Ross [115, p. 6]).

The following result clarifies the connection between Y^δ and supermodularity.

Lemma 4.18 $Y^+(\{a, b\}, c)$ *(respectively Y^- and Y^0) holds iff the function*

$$e_\phi(a, b|x) = \int \phi(c_0) f_c(c_0|abx) dc_0$$

is supermodular (submodular, modular) in a and b for all increasing functions ϕ and contexts x.

The function e_ϕ is the expectation of c under the monotonic transform ϕ. The equivalence between *sub*modularity for all c (Definition 4.16, the Y^+ condition) and *super*modularity of expectations for all ϕ is reminiscent of the correspondence between the FSD condition (4.3) and increasing expectations for all ϕ (4.4).

Once again, it is useful to define special notation for synergistic influences on the value node.

Definition 4.19 (Y_U^δ) *$Y_U^\delta(T,G) \equiv Y^\delta(T,v,G)$. The direction of synergy δ is $+$, $-$, or 0 as u is supermodular, submodular, or modular under all monotonic transforms.*

In the terminology of utility theory, δ-modularity expresses *multiattribute risk aversion*, *proneness*, or *neutrality* as δ is $-$, $+$, or 0, respectively [27, 112]. Multiattribute risk neutrality is equivalent to additive separability for u [32], as suggested by the form of the modularity condition (4.17).

The correspondence between Y_U^δ and supermodularity is useful because of the monotone decision property of supermodular functions. Consider the situation of Figure 4.9. There we have $Y_U^+(\{a,b\})$ even though $dir(a,v) = dir(b,v) = ?$. Qualitative influences alone tell us nothing about which value we should choose for the decision variable a. Positive synergy, on the other hand, implies that if b is observable, our policy should be to choose higher values of a for greater values of the observed b. While this still does not reveal the exact value of the optimal a, it dictates the *form* that our strategy should take.

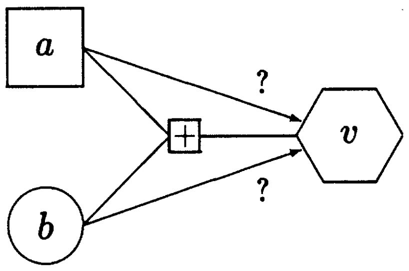


Figure 4.9: Synergistic influence on utility. Even though $U^?(a)$ and $U^?(b)$ we can deduce that the optimal choice of a is increasing in b.

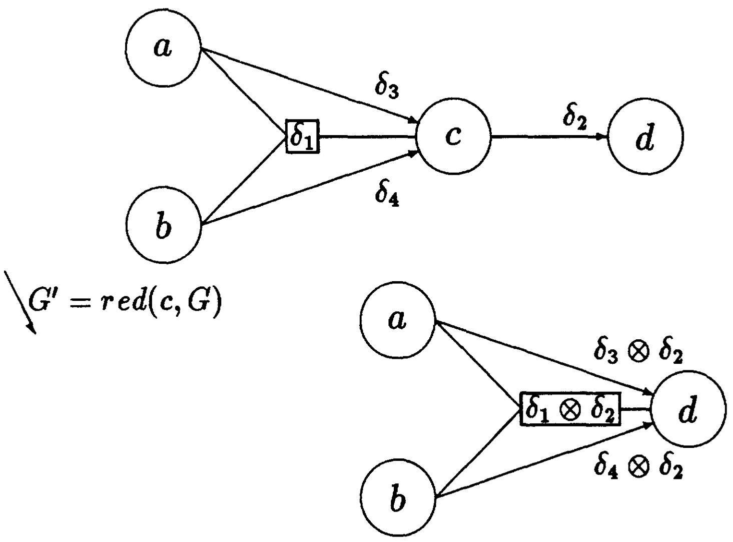
4.7.4 Propagation of Synergies in Networks

The mechanisms for deducing indirect synergies that hold in a QPN are analogous to the network transformation techniques for qualitative influences developed in Section 4.4. In particular, we can extend qualitative synergies through qualitative influences by variable reduction.

Theorem 4.20 *Synergies can be extended along qualitative influences by reduction according to the following:*

$$Y^{\delta_1}(\{a,b\},c,G) \wedge S^{\delta_2}(c,d,G) \wedge S^0(a,d,G) \wedge S^0(b,d,G) \Rightarrow$$
$$Y^{\delta_1 \otimes \delta_2}(\{a,b\},d,red(c,G)).$$

This reduction is depicted in Figure 4.10.



a
b
c
d
δ_1
δ_2
δ_3
δ_4
$G' = red(c,G)$
a
b
d
$\delta_1 \otimes \delta_2$
$\delta_3 \otimes \delta_2$
$\delta_4 \otimes \delta_2$

Figure 4.10: Propagation of synergy through qualitative influences. Values for $dir(a,d,G')$ and $dir(b,d,G')$ follow from Theorem 4.10. The new synergy $Y^{\delta_1 \otimes \delta_2}$ is the result of Theorem 4.20.

Like Theorem 4.10, Theorem 4.20 requires that there be no direct influences among the variables newly linked in the reduced QPN. The next result provides the reduction rule for the more general case.

Theorem 4.21

$$Y^{\delta_1}(\{a,b\},c,G) \wedge S^{\delta_2}(c,d,G) \wedge Y^{\delta_3}(\{a,c\},d,G) \wedge Y^{\delta_4}(\{b,c\},d,G)$$

$$\wedge S^{\delta_5}(a,c,G) \wedge S^{\delta_6}(b,c,G) \wedge Y^{\delta_7}(\{a,b\},d,G) \Rightarrow$$
$$Y^{(\delta_1 \otimes \delta_2)\oplus(\delta_3 \otimes \delta_6)\oplus(\delta_4 \otimes \delta_5)\oplus \delta_7}(\{a,b\},d,red(c,G)). \quad (4.18)$$

Theorem 4.21 generalizes Theorem 4.20 because $S^0(a,d,G) \wedge S^0(b,d,G) \Rightarrow \delta_3 = \delta_4 = \delta_7 = 0$ by conditional independence.

Note that the signs of direct influences from a and b to d do not affect the synergy propagation, though the signs of influences on c do. This more complicated situation is illustrated in Figure 4.11 below.

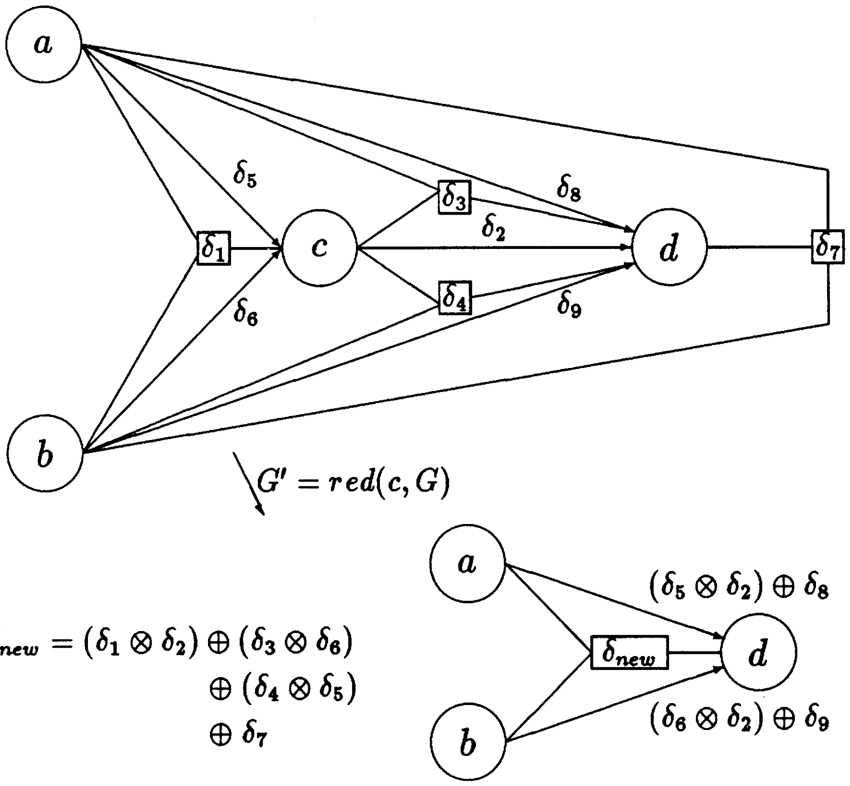

a
b
c
d
δ_1
δ_2
δ_3
δ_4
δ_5
δ_6
δ_7
δ_8
δ_9
$G' = red(c,G)$
$\delta_{new} = (\delta_1 \otimes \delta_2) \oplus (\delta_3 \otimes \delta_6) \oplus (\delta_4 \otimes \delta_5) \oplus \delta_7$
δ_{new}
$(\delta_5 \otimes \delta_2) \oplus \delta_8$
$(\delta_6 \otimes \delta_2) \oplus \delta_9$

Figure 4.11: Variable reduction with parallel synergies.

A special case of this result demonstrates how to propagate synergies *backwards* through qualitative influences. Upon reduction, a variable's predecessors assume its role in all synergies, with modified signs reflecting the direction of the predecessor's influence.

Corollary 4.22

$$Y^{\delta_3}(\{a,c\},d,G) \wedge S^{\delta_6}(b,c,G) \wedge Y^0(\{a,b\},c,G) \wedge S^0(a,c,G) \Rightarrow$$
$$Y^{\delta_3 \otimes \delta_6}(\{a,b\},d,red(c,G))$$

The result follows from the assignment $\delta_1 = \delta_5 = \delta_7 = 0$ in Theorem 4.21. Application of Corollary 4.22 is illustrated in Figure 4.12.

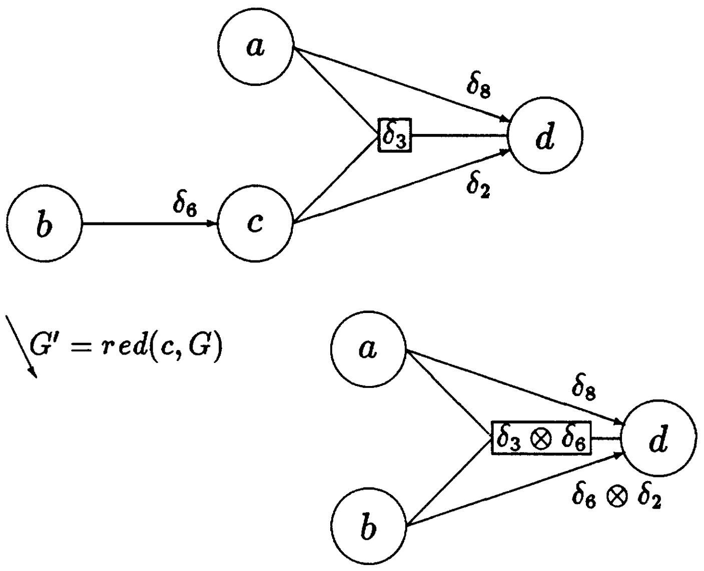


Figure 4.12: Backwards propagation of synergies through qualitative influences.

For an example of the use of backwards propagation, consider a synergy relation from the digitalis model. In the more detailed model of Figure 4.13, the effects of variables *dig* (digitalis dosage) and *Ca* (measured serum calcium) would be mediated by *dig'* and *Ca'*, the actual concentrations of digitalis and calcium in the bloodstream. Even though the synergy assertion is in terms of the physiological parameters, we can deduce synergy on the practically relevant proxy variables by reduction according to Corollary 4.22.

A canonical decision situation with the above form is the estimation problem from statistics. The problem is to choose an estimate *a* of the true "state of nature"

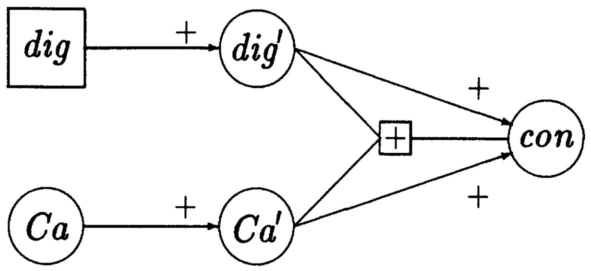


Figure 4.13: An elaboration of a digitalis model fragment. Variables *dig* and *Ca* represent dosage and measurement, respectively, while the primed versions are actual concentrations. The unprimed variables are synergistic by reduction of *dig'* and *Ca'*.

θ given only an observation z that is statistically related to θ. Karlin and Rubin [67] demonstrate that if

1. the optimal estimate is increasing in θ (the monotone decision property of Section 4.7.3),
2. utility decreases away from the optimum, and
3. z is related to θ by the MLRP (the likelihood condition for S^+ (4.6), Section 4.3.2),

then a and z also satisfy the monotone decision property.

By representing the estimation problem as the QPN of Figure 4.14, we see that Corollary 4.22 is a similar result, with the monotone decision property replaced by the stronger condition of qualitative synergy. Synergy seems justified for the estimation problem because the relative value of a higher estimate increases with the state of nature.

The applicability of the setup in Figure 4.14 goes well beyond estimation. Suppose the state of nature θ represents an unobservable disease severity and the decision variable a the aggressiveness of therapy. Choosing a therapy level is similar to estimating the severity of disease, as more serious conditions call for stricter treatments.

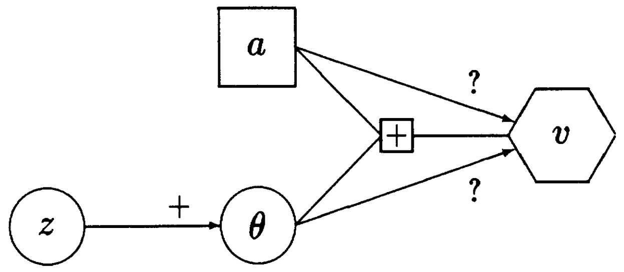


Figure 4.14: A qualitative probabilistic network for the estimation problem.

It is essential that a program be capable of inferring the qualitative implications for therapy of any symptom z related to disease severity in a known direction.

4.7.5 Landmark Values

The monotone decision property can be used to develop a concept of *landmark values* for QPNs analogous to the landmark value concept in qualitative simulation [73]. A landmark value is any distinguished point in the domain of a variable. Their usefulness to qualitative reasoning accrues when landmark values of several variables correspond in a meaningful way or the point has some other qualitative significance for the application.

In QPNs, the interesting landmarks are optimal values of decision variables and the corresponding values of observable non-decision variables. Suppose that in the disease-severity interpretation of Figure 4.14, the variable z represents an observable symptom with a specially designated "normal" value of z^*. There is a corresponding landmark value of the decision variable, a^*, representing the optimal level of therapy given $z = z^*$. The value of a^* may be known to the program, especially if there is documented experience with z-normal patients, everything else being equal. Even if its exact value is not known, or if it depends on other variables, the a^* concept has meaning as a landmark value in terms of its optimality property.

Suppose further that a patient presents with an elevated z-value of $z' > z^*$. The qualitative implication drawn from our model is that the corresponding optimal therapy a' is increased, $a' \geq a^*$, all else being equal. As correspondences in the quantity space [34] are known in finer detail, the program can determine optimal strategies with increasing precision.

4.7.6 Synergy in the Digitalis Example

To complete our discussion of qualitative synergy, let us return to the digitalis model of Figure 4.7. As promised, I start by justifying the synergy relation between *hr* and *vf*.

Consider two heart rates, $hr_1 \geq hr_2$, and the two values of the binary variable *vf*. The synergy condition, $Y_U^+(\{hr, vf\})$, is an instantiation of Definition 4.19:

$$u(hr_1, VF) - u(hr_2, VF) \geq u(hr_1, \overline{VF}) - u(hr_2, \overline{VF}). \tag{4.19}$$

Given *VF*, the heart rate is irrelevant (and ill-defined because ventricular fibrillation is a state where the heart is not contracting regularly). Therefore, the left-hand side of (4.19) is zero. For patients *not* in fibrillation, lower heart rates are preferable, by $U^-(hr)$, at least within the range considered here. This implies that the right-hand side of (4.19) is negative, satisfying the inequality.

By applying the results of Section 4.7.4, we can successively reduce any variables positioned between the ones of interest. Figure 4.15 shows the result of removing all but *dig*, *Ca*, and *v*. The final step, transformation from the fragment of Figure 4.15a to that of 4.15b, requires parallel combination of synergies using Theorem 4.21.

The final result of the exercise is that while the value of administering digitalis is ambiguous, by $U^?(dig)$, we can deduce that the optimal dosage is a decreasing function of calcium, by $Y_U^-(\{dig, Ca\})$. The more detailed model of Figure 4.13 showed us that this result holds whether we are talking about the actual substance concentrations

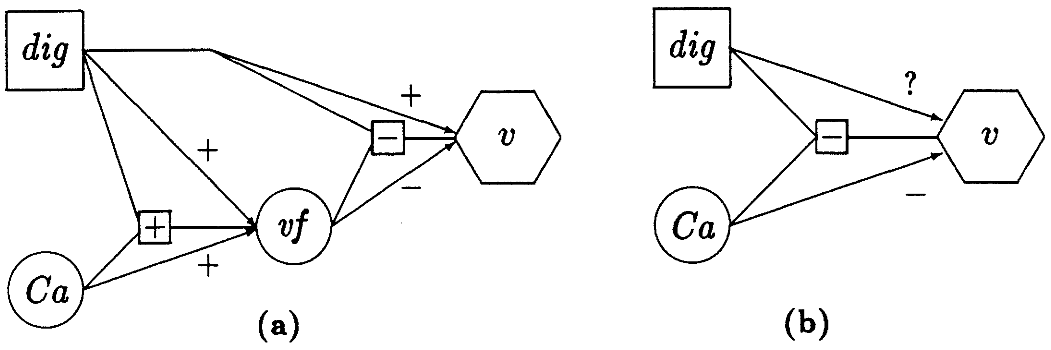


Figure 4.15: Transformation of the digitalis model with synergy: (a) collapsing the therapeutic pathway and consolidating the toxic, (b) final situation after reduction of *vf.*

in the bloodstream or about the amounts administered and measured by imperfect means.

Inferences of this sort play a central role in therapy planning and in the development of consultation systems via automatic programming [94, 141]. For planning, this type of result is a constraint on the class of admissible plans, significantly pruning the search space [158]. This is an especially useful kind of constraint for the automatic generation of a consultation system because the qualitative form of the solution corresponds to the structure of part of the target code.

The digitalis dosage d^* for patients with normal calcium—a distinguished point in the quantity space for *Ca*—is a landmark value as described in Section 4.7.5. Subsynergy implies that the dosage for a patient with calcium above normal should be lower than d^*. This is essentially the strategy of the digitalis program produced by Swartout's XPLAIN system [141], where a domain principle mandates that dosage should be adjusted according to "drug sensitivities." QPNs provide a more general and principled language for encoding domain knowledge, from which policies such as this can be derived.

4.8 Related Work

The QPN representation and reasoning techniques presented here borrow many concepts from other work in AI and decision theory. The most obvious debt is to research in numeric probabilistic networks, especially that of Pearl [103] and Shachter [123]. This work also relates to other efforts by similarity of purpose. In the following sections I compare it with research in qualitative probability, ordering relations on random variables, and qualitative reasoning.

4.8.1 Qualitative Probability

The central task in designing a qualitative probability representation—indeed in the design of a qualitative representation for anything—is choosing the important qualitative distinctions to make. For example, a straightforward mapping of techniques from qualitative physics might suggest that we carve up the $[0,1]$ probability scale into a quantity space by choosing a small set of designated reference points. For example, the set of points $\{.01, .05, .5, .95, .99\}$ might be chosen as especially significant.

Such a scheme is a non-starter because it is only by coincidence that the important qualitative thresholds for any problem will align themselves with the fixed boundaries in the probabilistic quantity space. Furthermore, it is not clear that the types of manipulations typically performed on probabilities will respect these boundaries in a systematic fashion. Though I do not know of any serious proposal along these simplistic lines, attempts to construct qualitative notions of *absolute* probability (see, for example, the work of Halpern and colleagues [47, 48]) are likely to encounter similar problems. Unlike the scales of physical parameters, the probability interval does not appear to have values (except the endpoints) that are universally interesting or even of special significance within a domain. And the *qualification problem* [130] is inevitably important here because one can almost always think of conditions that would bring the probability of any non-analytic event outside any given non-universal range.

This suggests that it might be more appropriate to base qualitative probability concepts on *relative* likelihoods. A relative likelihood logic permits statements that one formula is more likely than another [28, 39]. Absolute probability is subsumed by a scheme of this type given a set of special formulas corresponding to canonical chance situations (such as experiments with an idealized coin) of all probabilities.

The qualitative relationships presented here can be viewed as a special case of relative likelihood where only assertions about the comparative probability of particular conditional events are permitted. Both S^δ and Y^δ are limited to comparisons of the likelihood of a given event under different conditions. For the binary case, S^δ induces a quantity space on the *likelihood ratio* (4.7) with a distinguished value of one.

There are three primary advantages to restricting the formalism to these special likelihood comparisons. First, the class is closed under the reduction operations presented in Sections 4.4 and 4.7, and thereby is supportive of tractable inference procedures. Second, the ability to deduce decision properties suggests that these comparisons are making some of the significant qualitative distinctions. And third, the *ceteris paribus* condition in the definitions reduces the impact of the qualification problem, as does the embedding of the formalism in closed-world networks.

The enterprise of qualitative probability is not necessarily hostile to quantitative probability. In Savage's axiomatization of Bayesian decision theory [121, Chapter 3], the qualitative likelihood ordering logically precedes development of quantitative probability measures.[9] The existence of a numeric representation for likelihood is only a convenient fact that simplifies much of the theory and supports some direct ap-

[9]Strictly speaking, the qualitative theory is more general than the quantitative one, which typically requires some sort of additivity axiom. This is not, however, a motivation for the present work (indeed, the proofs assume additivity), which stresses advantages for knowledge representation and computation.

plications. The emphasis to date on numerical probability representations in applied decision theory and AI is due more to technological history than to any fundamental requirement that we focus exclusively on the precise extreme of the representation spectrum.

4.8.2 Relations on Random Variables

Philosophers have long attempted to develop mathematical definitions of causality, occasionally producing probabilistic interpretations. Motivated by a more limited set of concerns, I have ignored in this treatment temporal properties, mechanisms, and other issues salient to causality. These matters aside, Suppes [140] proposes a probabilistic condition for binary events that is equivalent to S^+ (4.1) without the context quantification. For multi-valued variables, Suppes suggests quadrant dependence (4.11). A cause is considered spurious if the probabilistic relation can be explained by a prior common cause. The concept of spuriousness can be partially captured in QPNs by distinguishing qualitative influences inferred via arc reversals (spurious) from those derivable solely from reductions along influence chains (genuine). This is similar in spirit to the approach of Simon [131], and is equivalent to the distinction emphasized by Pearl [102] between causal and evidential support.

As suggested previously, ordering of random variables has also attracted considerable interest in statistics [3, 78, 115] and decision theory [162]. Milgrom [90] demonstrates the application of MLRP to theoretical problems in informational economics.

The key difference between the S^+ definition proposed here and previous work is that we obtain transitivity by requiring the condition to hold in all contexts. Humphreys [64] proves a special case of Theorem 4.10 to the effect that binary qualitative influences along Markov chains (graphs where each node has a single predecessor, thereby eliminating context) can be combined by sign multiplication. In contrast, Suppes demonstrates that the causal algebra induced by his condition—defined only at the margin—does not possess the transitive property. A causal algebra either lacking sound reduction rules like those of Section 4.4.3 or restricted to simple Markov chains would have little value for knowledge representation.

Considerably less attention has been devoted to relations of probabilistic synergy. The supermodularity concept of Section 4.7.3 has not, to my knowledge, previously been interpreted in a probabilistic context. However, a constraint similar in spirit to sub-synergy was exploited by NESTOR [14, page 102], a diagnostic program based on probabilistic inequalities. (NESTOR used qualitative influences to bound probability intervals as well.) And we saw in Section 4.7.2 that several canonical probabilistic models proposed for AI programs are special cases of Y^{δ}.

4.8.3 Qualitative Reasoning

It might appear at first glance that the very imprecision sanctioned by qualitative mechanisms obviates the need to consider explicitly uncertainty underlying the models. This position, however, confounds the weakness of inferences and input specifications with other kinds of variability in the model. The distinction is crucial because the latter might undermine the soundness of conclusions drawn from qualitative knowledge bases.

The interpretation of a set of qualitative physical relationships as "qualitative differential equations" (see Kuipers [73], for example) treats each relationship as a constraint on some "true" functional relationship that holds over time. To assert that $b = M^{+}(a)$ (in Kuipers's notation) is to claim that there exists an increasing function f such that $b_t = f(a_t)$ for all t. This is incompatible with a probabilistic interpretation, even though f is only loosely constrained. A qualitative influence assertion of $S^+(a,b)$, on the other hand, leaves open the possibility that the relationship is non-deterministic (f might vary over time) and does not prohibit an increase in a from coinciding with a decrease in b.

Application of qualitative-physics inference mechanisms in a probabilistic environment is dangerous because they tend to take as impossible what is merely unlikely. For example, Forbus's measurement interpretation algorithm for Qualitative Process theory [35] prunes away the qualitative behaviors that are inconsistent with observations of the system. If the dynamics of the system are really probabilistic (I do not claim that this is the case for Forbus's application), then this step is not valid because no behaviors are truly inconsistent. In such a situation, measurements serve to change the likelihoods of various behaviors but never to rule them out. This difference is vital in a critical application because some highly unlikely behaviors may nevertheless be important enough to demand attention from the reasoner.

Though we cannot prune measurement interpretations, we might be able to perform some pruning on the plan space using the techniques presented above. A particular measurement does not in general reveal any facts about the other model variables with certainty, yet it may allow us to deductively conclude that some decision variables (perhaps dials in the control room) should be adjusted in particular directions.

4.9 Conclusion

4.9.1 Summary

A QPN model represents qualitative constraints on the probabilistic relationships among a set of variables. In this chapter I have defined and analyzed two basic constraint types: qualitative influences that express direct relationships between variables, and qualitative synergies that express interactions among influences. The probabilistic definitions justify sound graph-based inference procedures that answer queries about the qualitative relationship of any subset of variables in the model. Qualitative relationships involving the special value variable v dictate structural properties of the optimal assignment to decision variables.

Despite the ubiquity of constructs similar to qualitative influences in knowledge representation mechanisms, there has been little study of the semantics of these statements. Previous work either denies the probabilistic nature of the relationships among variables in the model or takes for granted the ability to draw inferences by chaining influences in the network. I have defined a positive qualitative influence of a on b as an assertion that, in all contexts, the posterior probability distribution for b given a is stochastically increasing (in the sense of FSD) in a. A series of results provides theoretical support for this S^{δ} definition:

- S^{δ} justifies reduction of variables by influence chaining. Reduction of any subset of variables can be performed in $O(|V|^2)$ time.
- S^{δ} permits some nontrivial conclusions upon influence reversal.
- S^{δ} is the weakest posterior condition that justifies chaining of influences.
- S^{δ} is necessary and sufficient for chaining *decisions* across influences.

Two variables a and b are positively synergistic on c if the posterior distribution for c is increased more (in the sense of FSD) upon a positive change in a for higher values of b. This Y^{δ} definition has several computationally and decision-theoretically useful properties:

- Canonical models such as the "noisy OR" often entail Y^{δ}.
- Y^{δ} is equivalent to supermodularity on expectation with respect to all monotonic transformations.

- Y_U^δ implies the monotone decision property.
- Synergies may be propagated forwards or backwards along qualitative influences.

Together, the two qualitative relationships provide a simple yet powerful modeling language. A planner is often able to derive important facts about the qualitative structure of optimal strategies from only weak premises on the qualitative relationships in the domain.

4.9.2 Discussion

Though powerful in some respects, the qualitative relationships are also quite limited. As we saw in Section 4.6, qualitative influences are unable to resolve true tradeoffs because parallel influences of different sign are indeterminate in combination ($+ \oplus - = ?$). Indeed, "unresolvable in a QPN" might be the best available formal definition of a tradeoff situation.

Thus, a QPN knowledge base can support planning "up to tradeoffs." To proceed beyond that point, we need more precise knowledge of the domain. I see no insurmountable barriers to the development of hybrid representations that augment QPNs with stronger constraints, up to and including constraint to exact numeric values. As mentioned above, features of such a hybrid scheme were explored by Cooper in the NESTOR project [14]. While NESTOR's basic representation was probability intervals, it applied constraints similar to qualitative influences and synergies to bound the result of certain combination operations.

Finally, evaluation of QPNs as a knowledge representation must also take into account the feasibility of constructing knowledge bases of reasonable complexity. For reasons discussed above, QPNs should be substantially easier to generate than their numeric counterparts. This prospect is explored further in the next two chapters, which describe SUDO-PLANNER's knowledge base and its procedure for constructing QPNs from the KB.

Chapter 5

Effects of Actions

The representation of effects is a central issue in planning research and is perhaps the factor that most distinguishes different designs. The topic requires special attention in this work because traditional representations cannot accommodate actions with uncertain effects. Mechanisms from previous work in planning under uncertainty have limited applicability for SUDO-PLANNER because the descriptions of effects were not specifically designed to support dominance-proving.

The first three sections in this chapter address basic issues that loom large in efforts to design a representation for the effects of actions. The distinction between terminological and assertional knowledge has implications for the role of effects in characterizing actions in a knowledge base. The frame problem remains an obstacle in the quest for tractable action representations that are formally sound. Finally, a realistic action representation must provide for context-dependent effects, though AI planners have traditionally lacked this facility.

The discussion of basic issues prepares the way for a presentation of SUDO-PLANNER's representation. Effects in SUDO-PLANNER are based on the qualitative relations of Chapter 4. Section 5.4 describes the knowledge base constructs for specifying effects and discusses some difficulties in relating them precisely to QPN semantics. Section 5.5 introduces the Markov influence, a special construct that facilitates specification of simple temporal effect patterns. Inheritance of effect assertions in the event taxonomy is the subject of Section 5.6. Finally, Section 5.7 describes SUDO-PLANNER's mechanism for handling the information-gathering behavior of actions. SUDO-PLANNER's procedure for generating QPNs from the knowledge base of effect assertions is presented in Chapter 6.

5.1 Terminological and Assertional Knowledge

Philosophers and knowledge representation theorists sometimes emphasize the distinction between knowledge that is *definitional* for a concept and *factual* information true only by happenstance. On the AI side, for example, Woods contrasts structural (definitional) and assertional (factual) features of semantic network representations [165, Section III.J]. The KRYPTON knowledge representation system [6] realizes this distinction by explicitly separating its terminological (definitional) reasoning component ("T-box") from the assertional reasoner ("A-box").

SUDO-PLANNER's representation for actions (Section 3.2) is based on NIKL, the T-box of KL-TWO [150]. Its taxonomy of action types, therefore, is based entirely on terminological knowledge.[1] Conversely, SUDO-PLANNER uses the action definitions exclusively to generate the taxonomic relations among action types. Mixing assertional knowledge with the definitions of actions in a terminological system like NIKL interferes with classification, thereby degrading the effectiveness of taxonomic reasoning. Haimowitz et al. [46] recount other undesirable consequences of T-box abuse.

In principle, actions could be defined in part by their effects. For example, we could define **treatment** as an action that alleviates some disease, and its specializa-

[1]There is substantial room for philosophical controversy about whether the information in NIKL concept specifications is truly definitional. All that matters for this discussion, however, is that the reasoner interprets the specifications as definitions.

tion **CAD-treatment** by value-restricting the **disease** role to **CAD**. Alternatively, actions can be defined *operationally*, that is, in terms of their objects, instruments, and procedures.

Permitting both types of definition presents a difficulty for terminological reasoning because the effects of operationally defined actions are typically not derivable from their specifications. For example, **CABG** is in fact a **CAD-treatment**, but this does not follow from its operational definition. It is unclear whether **CABG** should be classified as such in the taxonomy of Figure 5.1 because this would inhibit proper classification of its operationally defined subtypes, depending on the specification conventions and the classification algorithm.

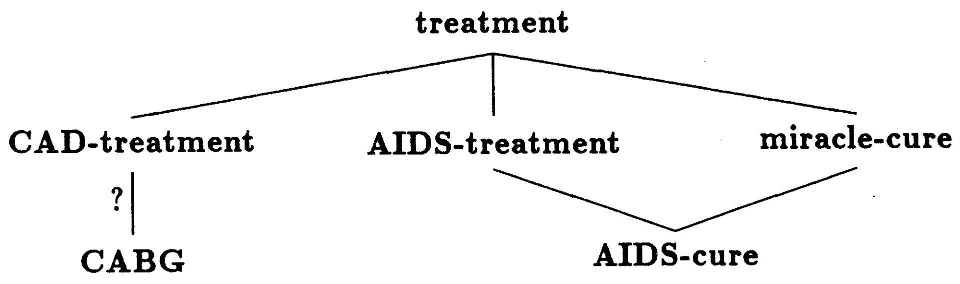


Figure 5.1: A pseudo taxonomy where actions are defined by their effects.

Defining actions by their effects makes it difficult to ensure that the agent can actually execute actions prescribed by the planner. The action type **AIDS-cure** in Figure 5.1, defined as a "treatment that cures AIDS," is perfectly coherent, but it cannot be implemented because it has no known instances (at the time of this writing). Unless the planner realizes this, it will offer vacuous recommendations whenever it encounters AIDS patients.

To prevent this potential fallacy, all information about the effects of actions in SUDO-PLANNER is treated as assertional knowledge. Definitions of actions must be "ineffective," referring only to operational features, as described above. Although nothing in SUDO-PLANNER's representation *guarantees* that definitions are operational or that action types are instantiable and implementable, banishing effects ensures that SUDO-PLANNER is not subject to the wishful-thinking delusion illustrated by the **AIDS-cure** action.

This is not to say that any scheme using "effective" definitions will produce practically worthless plans. Given a fixed knowledge base of implementable actions defined by their effects (as in the system of Swartout and Neches [142]), the planner will produce legitimate plans. However, if we permit the planner to construct its own action definitions dynamically from other concepts in the knowledge base (as in SUDO-PLANNER), including effect constructs can lead to non-implementable action types as above.

An alternative approach to the problem would be to explicitly represent and consider the *knowledge prerequisites* associated with each action [93]. Given a description of the agent's knowledge, the planner could recognize its inability to implement **AIDS-cure**. The "ineffective definition" principle I am advocating here is a simpler, global convention that achieves the purpose of knowledge prerequisites for an important special case.

5.2 The STRIPS Assumption

5.2.1 The Frame Problem

The classic dilemma in representing and reasoning about the effects of actions is the *frame problem* [88, 130]. The frame problem has come to stand for a variety of computational and notational complexities arising from the apparent necessity of considering the possible change in status of every proposition for each action. Actual planners circumvent these difficulties by restricting attention to propositions explicitly mentioned in action specifications, a convention first applied by STRIPS [30].

Waldinger has named this policy the "STRIPS assumption" [152]. Characterizing such policies in a formal logic has proven to be a difficult task for AI theorists [49].

The frame problem is just as important and difficult when actions have uncertain effects. My solution approach in the uncertain case is similar in form to traditional techniques, with results of comparable adequacy. Actions are presumed to affect directly only those variables explicitly referenced in their specifications. Of course, the uncertain nature of effects requires representations quite different from those of categorical planners, and non-effects likewise require a probabilistic interpretation. While add and delete lists that specify the propositions changing truth value as a result of an action are sufficient for categorical planning,[2] actions with uncertain effects must describe changes in probabilistic relations that occur when the action is performed.

5.2.2 The STRIPS Assumption for Uncertain Effects

Generally stated, the STRIPS assumption dictates that the effect of an action on the world model be completely determined by the direct effects specified in its description. We can characterize the implications of this assumption for planning under uncertainty in terms of dependency graph concepts (see Section 4.4.1). Let S_a be the set of event variables that action variable a directly affects. The STRIPS assumption for planning under uncertainty is that there exists an I-map G such that for all event variables e,

$$a \in \mathit{pred}_G(e) \Rightarrow e \in S_a. \tag{5.1}$$

Condition (5.1) has direct implications for probabilistic independence. By virtue of G's I-mapness and Lemma 4.9, a is conditionally independent of any $e \notin S_a$ given

[2]That is, the truth value changes are sufficient for defining modifications to the world model. Specifying and implementing truth value change is tricky, however, because changes in status for the sentences mentioned imply changes in their logical consequents. Lifschitz demonstrates how to account for this correctly within the STRIPS framework [82].

$pred_G(e)$. Each variable $d \in pred_G(e)$, in turn, is either a direct effect of a or is conditionally independent given its predecessors, $pred_G(d)$. Ultimately, the effect of a on e is completely determined by a's direct effects and e's relation to them. Note that we still need to describe the interaction, if any, between a and e in their joint effects. Section 5.3 discusses this issue further.

The probabilistic STRIPS assumption does *not* require that a be conditionally independent of e given the direct effects S_a, or even by any subset of S_a. In Figure 5.2, for example, a and e are d-separated (Definition 4.8) by $\{s,y\}$ but by no other variable set. The predecessor y is necessary for conditional independence of a and e even though y itself is unconditionally independent of a.

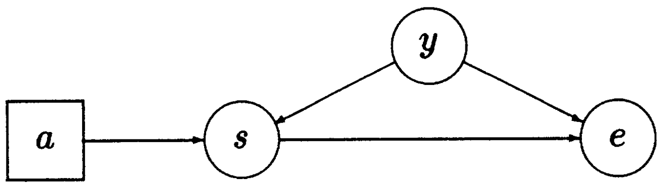


Figure 5.2: Action a is conditionally independent of e given $S'_a = \{s,y\}$ but not given any subset of its direct effects $S_a = \{s\}$.

If we enlarge the conditioning set to include predecessors of a's direct effects, however, we get another valid independence condition. Let S'_a be the set of a's direct effects plus the other variables that affect those effects:

$$S'_a = S_a \cup \bigcup_{s \in S_a} pred_G(s) - \{a\}.$$

The graphical condition (5.1) implies that a is conditionally independent of e given S'_a, as long as $e \notin S'_a$. In the graph of Figure 5.2, for example, $S'_a = \{s,y\}$.

In summary, we can formalize the STRIPS assumption under uncertainty in terms of probabilistic conditional independence. In particular, given a dependency graph of variables in the world model, any variable e not specified as an effect of action a must not be directly connected to a. Under this condition there may be a probabilistic dependency between a and e in some situations, but this can always be described in terms of a's and e's relations to S_a.

5.3 Context-Dependent Effects

It is not possible in general to describe the effect of an action on a single event in isolation. Typically, the effect will depend on the context in which the action is performed, including the other actions in the plan and the values of other event variables in the world model. Consideration of context requires an effect representation expressive enough to capture interaction among actions and events in their joint effects.

Traditional planning representations do not provide for context-dependent effects, except through the use of preconditions. Preconditions provide only a gross form of context dependence where the action is prohibited in designated situations. Formally, preconditions are a special case of more general specifications of effects as a function of the action in conjunction with the background situation.

Rather than saying an action cannot be performed in situations failing to meet a set of preconditions, we could specify that it does not have particular effects (usually the desired ones) unless the conditions hold. Under this interpretation, the planner can apply $\textit{put-on}(a, b)$, for instance, in any situation, but the result $\textit{on}(a, b)$ is conditional on $\textit{cleartop}(a)$ and $\textit{cleartop}(b)$. Otherwise, the effects of *put-on* are undefined. By placing the conditions on effects rather than on the action, we can employ actions that may have many contingent effects. In addition, the planner is free to introduce such actions into the plan without guaranteeing that the preconditions are satisfied.

Pednault has recently generalized traditional planning representations to include "secondary preconditions" for specifying context-dependent effects [106]. While "primary" preconditions can be treated as special cases of these, there may be some utility to keeping them conceptually distinct in a practical planning system.

5.4 Specification of Effects

The effects of actions in SUDO-PLANNER are encoded as qualitative relations among action and event variables. Qualitative influences describe the direct effects of an action, and qualitative synergies describe interactions among influences, or context-dependence (Section 5.3). Effects are specified simply by S^δ (qualitative influence) and Y^δ (qualitative synergy) assertions, with $\delta \in \{+,-,0,?\}$. S^0 assertions are implicit, by the probabilistic STRIPS assumption (5.1) of Section 5.2.2.

5.4.1 Semantics of Effect Assertions

An assertion of the form $S^\delta(av, ev)$ or $Y^\delta(av, ev_1, ev_2)$ means that action and event variables of types av and ev participate in the designated relation. Effects of events on other events are similarly specified. The knowledge base construct S^δ is distinguished from the QPN predicate S^δ in two ways:

1. There is no third argument. In QPNs, S^δ is always evaluated with respect to some network G. S^δ assertions in the knowledge base cannot refer to particular QPNs because they are, in fact, the source of QPNs assembled during planning.

2. The variable arguments are types, not instances.

A similar distinction applies to Y^δ.

Despite the rigorous probabilistic definition of qualitative relations in QPNs, the semantics of these assertions in the knowledge base are not immediately clear. The S^δ definition (4.4), for example, refers explicitly to the context x composed of the "other predecessors" of the affected variable. Without a G argument, the assertions must be given a context-independent interpretation. And because probability distributions over event variable types are undefined, the meaning of the qualitative relations must derive from the process of instantiating the types to individual variables.

A complete declarative semantics for effects in the SUDO-PLANNER knowledge base is a topic for further research. Some properties of these assertions can be presented in declarative form, however, and the description of the QPN construction algorithm in Chapter 6 provides further constraint on their interpretation. Nevertheless, the current absence of a full semantic account complicates knowledge engineering and hinders the formal analysis of SUDO-PLANNER's model construction techniques.

5.4.2 Special Constructs

It is possible, and often useful, to introduce special knowledge base constructs to capture regularities in the S^δ and Y^δ relations among instantiated event variable types. Such constructs offer notational economy by specifying patterns of effects with a single assertion. In addition, they can be used to direct the global course of model development through special handlers in the model construction procedure. An example of a special effect construct (the only one implemented in SUDO-PLANNER) is the "Markov influence" introduced in Section 5.5. By accounting for a simple but common pattern of effect over time, Markov influences help to redress the lack of temporal structure in SUDO-PLANNER's event representation.

5.5 Markov Influences

A Markov influence represents an effect that depends on the prior state of the affected variable. In other words, the value of the affected event variable after an action depends on its value before the action as well as on the value of the action variable.

For example, *CABG* alleviates *CAD*, but the distribution of final *CAD* states also depends on the initial *CAD* value. To represent this directly in the knowledge base, however, we would need separate concepts for "CAD before CABG" and "CAD after

CABG." Such a scheme is infeasible because it requires distinct concepts for each event variable in all potentially relevant temporal relationships to the action variable instances.

Markov influences specify state change for a variable across time periods without explicitly referring to the times involved. The form of a Markov influence assertion of *av* on *ev* is $K^{\delta_1,\delta_2}(av, ev)$, where δ_1 is the direction of influence and δ_2 is the direction of synergy between *av* and the prior value of *ev*. For example, the Markov influence of *CABG* on *CAD* is

$$K^{-,-}(\textbf{presence}\langle\textbf{CABG}\rangle, \textbf{extent}\langle\textbf{CAD-event}\rangle). \tag{5.2}$$

The first "−" indicates that *CABG* stochastically decreases *CAD*, and the second asserts that the effect is more negative the greater the extent of the patient's original coronary disease. Figure 5.3 displays the QPN fragment generated by the Markov influence in this example.

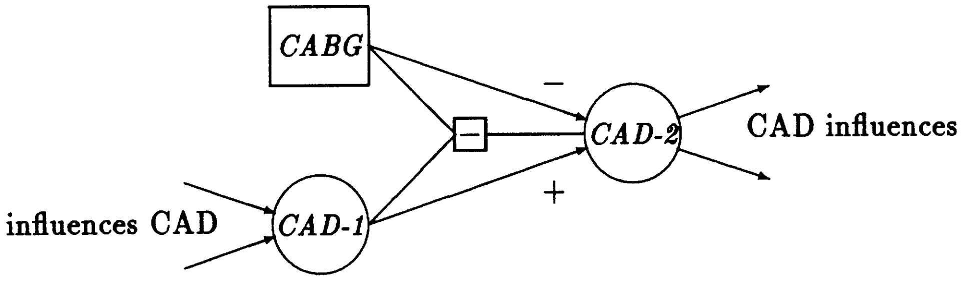


Figure 5.3: The negative Markov influence of CABG on CAD.

In interpreting K^{δ_1,δ_2} during QPN construction, SUDO-PLANNER modifies event variables and their relationships and introduces any new structure required. (Section 6.3.3 describes the interpretation procedure in detail.) For the CABG/CAD influence (5.2), SUDO-PLANNER splits **extent⟨CAD-event⟩** into two QPN variables, denoted *CAD-1* and *CAD-2* in the figure. It adds an influence of sign δ_1 (−) from *CABG* to *CAD-2*, along with a qualitative synergy of sign δ_2 connecting *CABG* and

CAD-1 to *CAD-2*. SUDO-PLANNER also introduces a positive influence from *CAD-1* to *CAD-2*, based on the general assumption that the value of the affected event after the action, ev_2, is positively related to its prior value ev_1.

The $K^{\delta_1,\delta_2}(av, ev)$ relationship is called a Markov influence because the split event variable enforces a Markov independence property in the network. Specifically, influencers of the original variable ev are independent of ev_2 given the prior state variable ev_1 and the new influencer av. Similarly, variables affected by ev do not depend on the prior value ev_1 given the posterior value ev_2 and the variables that affect ev. This partitioning is demonstrated in Figure 5.3 by the delegation of all influencers to *CAD-1* and all influences to *CAD-2*.

5.6 Inheritance of Effects

As mentioned in Section 5.4.1, effect assertions in the knowledge base refer to event variable types, while effects in QPNs relate event variable instances. The correspondence between the two is determined in part by the implications of an effect assertion on effects among taxonomically related types. That is, some of the semantical questions raised above would be answered by an account of effect inheritance.

An effect assertion of the form $S^{\delta}(ev_1, ev_2)$ has the intended interpretation that *all* event variables of type ev_1 have a δ influence on *some* event variable of type ev_2. Therefore, the asserted effect also holds for any subtype of ev_1 on any supertype of ev_2. Effects associated with their antecedent type are inherited downward in the event variable taxonomy.

Defining the effect relation to have a universally quantified antecedent and an existentially quantified consequent preserves the important property of closure under transitivity. This feature is essential given the chaining inference rules for QPNs developed in Chapter 4. An effect of ev_1 on ev_n derived from paths of effects from the former to the latter will have the universal/existential form if all of the component effects are of that form.

The details of inheritance and its uses in QPN construction are deferred to Chapter 6. Practical considerations in model construction as well as a set of general desiderata for reasoning at multiple levels of abstraction (Section 6.1) justify the interpretation for effect assertions described here.

5.7 Creating Observables

Although the emphasis in this chapter is on effects that change the world, it is also important to consider the information value of performing actions. SUDO-PLANNER actions generate information by creating *observable* event variables. An observable event variable EV has a corresponding function $EV(obs)$ defined for $obs \in \mathcal{O}$. A plan π cannot specify differential action on EV unless EV is observable.

An observable creation assertion of the form $CO(a_i, ev)$ specifies that variables of type ev are observable in plans that include a_i. For example,

$$CO(\textbf{cardiac-catheterization(true)}, \textbf{extent}\langle\textbf{cardiac-cath-result}\rangle)$$

asserts that performing catheterization reveals the value of the test result to the agent. The result variable exists conceptually whether the test is performed or not, but without the test the agent cannot see the result or use it to condition subsequent action.

To SUDO-PLANNER's dominance prover, observability makes an event variable eligible to appear in action policy and conditional plan class constraints. Constraints of these types are ill-defined unless the event in question can be observed.

Observable creation places a semantic constraint on plans beyond the syntactic specification of Section 3.1 (see the informal Ω_0 definition (3.1)). Specifically, the plan must order the actions so that all observables are created before they are used as conditions. This constraint ensures that we are not implicitly conditioning an observation on the value of the observation itself.

Observable-creating actions are called *tests.* The treatment of observables here differs somewhat from the representation of information dependencies in influence diagrams [61]. There the observability of events does not change but tests influence the values of the observables. One way to implement this (Ross Shachter, personal communication) is to make the observed variable EV a deterministic function of an underlying physiologic state s and a binary variable indicating whether the test was performed. The result is $EV = s$ if the test is performed, with EV a noninformative constant otherwise. In the scheme employed by SUDO-PLANNER, the test action directly causes s to be observable. The result is formally equivalent, but maintaining the distinction in the knowledge base leads to a uniform interpretation of EV. Furthermore, the separation of information-gathering and world-affecting permits the dominance prover to apply constraints on the value of information (for example, its non-negativity [43]) in determining plan class admissibility.

Chapter 6

Model Construction

Customized construction of decision models at multiple levels of abstraction is SUDO-PLANNER's solution to the computational problems of adhering to decision-theoretic principles (see Sections 1.5 and 1.6). This chapter explores these issues further and describes the implemented model construction techniques. I start by outlining a challenging set of desiderata for multilevel reasoning in the general case. Although it does not satisfy the desiderata completely, SUDO-PLANNER constitutes a first step towards automatic model construction from a multilevel knowledge base.

6.1 Desiderata for Reasoning at Multiple Levels of Abstraction

As argued in Section 1.6, the interconnectivity of concepts in realistic knowledge bases dictates that we employ abstraction to avoid the requirement for exhaustive consideration of the KB before model analysis. Many researchers have proposed and developed techniques for reasoning at multiple levels of abstraction (see Section 6.5.2 for a partial review of this work). In this section, I outline a set of desiderata for multilevel reasoning schemes in the general case. Subsequent sections describe SUDO-

PLANNER's multilevel representation and model generation procedure and evaluate them in terms of these desiderata.

The following list contains some key features of an ideal multilevel reasoning system.

- *Non-Reductionism.* Lower levels need not be strict refinements of upper levels. The strength of conclusions can increase in either direction.
- *Fluidity.* The knowledge base is not strictly layered. Relationships need not respect levels.
- *Multilevel Operationality.* Substantive reasoning can occur at any level. The higher levels are not merely for control and explanation.
- *Definitional Clarity.* The representation has a clear semantics and the inference procedure, a transparent description.
- *Coherence.* Conclusions are consistent across abstraction levels.

Conceptually, these desiderata are orthogonal. Practically, they interact considerably, in that satisfying some may render others more or less easy to achieve. I discuss the desiderata and their interdependencies in the sections below. I place particular emphasis on non-reductionism and fluidity because these have had the largest influence on the design of SUDO-PLANNER.

6.1.1 Non-Reductionism

The simplest theoretical approach to abstraction is to regard the lower levels as refinements of the upper, thereby defining upper level concepts as versions of those below that ignore certain distinctions (see Hobbs [55], for example). Indeed, the submergence of detail is the usual meaning of the term "abstraction." Enforcing this criterion in practice, however, can lead to infeasible information requirements and fails to capture some known phenomena in multilevel reasoning.

Reductionism imposes a strong constraint on the knowledge in a multilevel system: conclusions derivable at a given level must also follow from models at deeper levels. While this property may hold in some cases, in others the local information associated with low-level concepts will not entail conclusions as strong as those obtained from the higher-level model. This weakness can be caused by several factors, including:

- Difficulty in expressing the stronger relationships with the "natural" vocabulary for the more specific concepts. For example, breaking an aggregate concept into descriptions of its components may render global properties inaccessible.
- Lack of information for instantiating the more specific concepts to obtain conclusions of comparable strength. Typically, lower-level concepts are more difficult to instantiate because they require more detail. Even though incomplete knowledge may be sufficient for the conclusions, the program may require concrete instances for tractable, "vivid" reasoning [81].
- Absence of a theory describing the relation between properties of the specific and general concepts.
- Absence of a theory of the specific concepts themselves.

A second argument against reductionism in multilevel modeling is that it is violated in the current state of human knowledge. In medical reasoning, for example, physiological knowledge can often resolve conflicts arising from application of empirical associations at the higher, clinical level in the presence of multiple disorders [143]. But in some areas of medicine our knowledge of physiological mechanisms is quite weak, perhaps supporting confident prediction of only qualitative relationships [75]. These may lead to useful conclusions in some cases, but not necessarily conclusions as strong as those obtained from shallower models. (They may not even reproduce all qualitative conclusions known at the empirical level). Proceeding downwards in the abstraction hierarchy to biochemistry almost always makes things worse because biochemical theory and our knowledge of the biochemistry underlying physiological processes is not strong enough to explain higher-level effects.

This phenomenon is by no means unique to medicine. Successful macroeconomic models typically cannot be grounded in microeconomic foundations. Even in physics—the pinnacle of reductionist science—more detailed descriptions do not always lead to stronger conclusions. For example, we typically cannot derive useful predictions from a description of a system at the quantum level.

The foregoing argument is not a philosophical objection to reductionism in principle. Until we carry out the reduction in actuality, however, a purely reductionist computational mechanism will not suffice for representing and reasoning about the body of human knowledge.

6.1.2 Fluidity

The second desideratum for multilevel reasoning is that the notion of "level" be fluid as opposed to rigid. A rigid multilevel system is one where all reasoning respects fixed level boundaries defined by the allowable relationships between concepts.

To clarify this, let us define more precisely the components of a multilevel representation. We call the basic elements *concepts* and presume that concepts are arranged in some taxonomic structure. Two concepts are *taxonomically related* if one is a descendant of the other in the taxonomy. Taxonomic relativity is the basis for statements that one concept is at a higher or lower abstraction level than another. Interpretation of the taxonomic relation may vary in different representation schemes. In NIKL (and hence SUDO-PLANNER), the taxonomies represent specialization hierarchies, where concept subsumption is defined with respect to an extensional semantics.

In addition to taxonomic relationships, a multilevel knowledge representation provides a set of *domain relations*. Depending on the purpose of the KB, these might express causal, functional, or any other useful type of information about the concepts. In SUDO-PLANNER, for example, I employ qualitative probabilistic relationships to represent the effects of actions.

A multilevel KB is *rigid* if we can partition the concepts into a set of fixed levels, such that:

1. All domain relations connect concepts at the same level.
2. No concepts at the same level are taxonomically related.
3. Taxonomic relations among concepts at different levels induce a total order on the levels.

The rigidity of a given multilevel KB can be decided by an efficient algorithm. Figure 6.1 illustrates a rigid multilevel KB. Relations among concepts respect the level boundaries as dictated above. An example of a system with a rigid KB is Patil's ABEL program for multilevel physiological reasoning [98].

A *fluid* multilevel KB (see Figure 6.2) does not enforce these restrictions. Domain relations between two taxonomically related pairs may cross, and a concept may appear in a domain relation with two or more others that are themselves taxonomically related. A fluid scheme may impose some discipline of its own on patterns of relations, but the restrictions are not as sweeping as those given above. Pople's CADUCEUS [107] is an example of a fluid multilevel KB architecture.

While we can speak of *relative* levels in a fluid KB, the notion of an *absolute* level is ill-defined. One concept is at a higher level than another if the two are taxonomically related, otherwise they are incomparable. Concepts a and b are at the same level

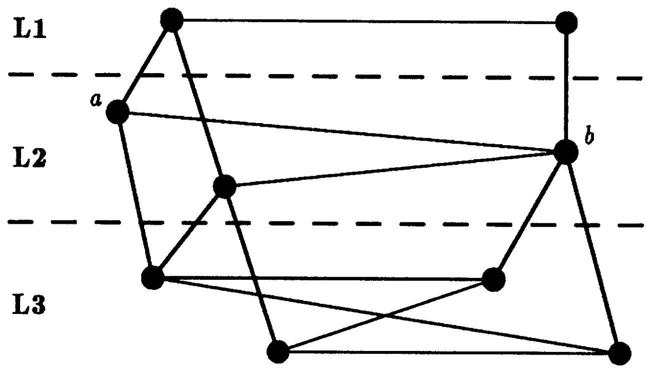


Figure 6.1: A rigid multilevel knowledge base. Thick lines indicate taxonomic relations; thinner ones denote domain relations. We can partition the KB into levels (L1, L2, L3) such that there are no inter-level domain relations and no intra-level taxonomic relations.

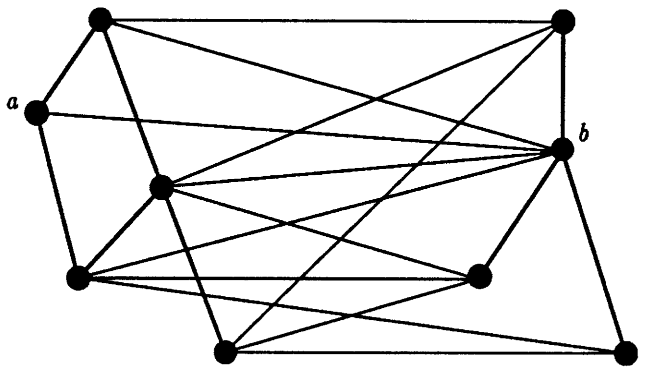


Figure 6.2: A multilevel system with a fluid knowledge base. No partitioning into levels is possible.

(L2) in the rigid KB of Figure 6.1, but there is no basis for such a statement in the fluid case of Figure 6.2.

One advantage of a rigid design is the ease of controlling multilevel reasoning with uniform levels. A rigid KB necessitates choice of only a single global level, in contrast to the selection of levels required for each local region of a fluid KB.

Although it complicates the control problem, the flexibility of fluid KBs has important advantages. The appropriate depth for reasoning about individual concepts depends on the goals and characteristics of particular problems, and there is no reason to expect that the same depth is appropriate throughout the knowledge base. For example, a reasonable medical reasoning strategy might be to explore in great detail the concepts relating to the patient's chief complaint while examining peripheral concepts at a higher level. A rigid multilevel reasoner can pursue this strategy by switching between levels for different parts of the analysis. However, the rigid reasoner can never mix concepts from distinct levels *in the same model* because it cannot express inter-level relationships directly. Therefore, it cannot consider the chief complaint and the peripheral concepts *together*, unless it is willing to represent them at the same abstraction level.

Fluidity is central to SUDO-PLANNER's ability to derive useful results before exhaustively examining the knowledge base. The example model of Figure 1.8 includes specific effects of **AAA-repair** as well as high-level effects associated with its ancestor concept, **treatment**. If the level of detail of a decision model were constrained to be uniform, reasoning could not commence until the model was translated to some common denominator. For large KBs, this requirement can impose unacceptable delays on model analysis.

6.1.3 Multilevel Operationality

In some multilevel reasoning systems, all conclusions are expressed in terms of the lowest-level concepts. Structure at higher levels is used for explanation, control of reasoning, and perhaps other purposes, but it is not operational in the same sense as knowledge at the lower fringe.

For true multilevel behavior, all levels should enjoy the same operational status. The full advantages of abstraction are realized only if it is possible to avoid some low-level concepts entirely; merely postponing attention to detail provides only limited savings. The requirement that the reasoner perform all substantive inference at the lowest level smacks of reductionism and compromises the flexibility of fluid representations.

There are two senses in which SUDO-PLANNER's high-level reasoning is substantive. First, the plan graph can express dominance results about plan classes at high levels of abstraction. In fact, these results are more valuable than those about lower-level plan classes because they prune more of the search space.

Second, the benefit of fluidity cited above would not be possible if substantive reasoning were not permitted at higher levels. Reasoning about some facets of **AAA-repair** at the generality of **treatment**—even though more specific knowledge about the same effects is available—leads to useful dominance results at an earlier stage of the modeling process than would otherwise be possible.

6.1.4 Definitional Clarity

The desirability of definitional clarity in a knowledge representation is self-evident, regardless of its number of levels. Unfortunately, a precise semantics for knowledge representation schemes is sometimes difficult to produce, as Section 5.4.1 illustrates. In the case of multilevel reasoners, our other desiderata further complicate the task. A reductionist scheme is theoretically much simpler than one that violates monotonicity of conclusion strength with refinement, and rigidity in level boundaries eliminates a large class of interactions that would present definitional difficulties. In my view, the flexibility advantages of fluidity and the unreality of reductionism outweigh the current lack of clear theoretical accounts of non-reductionist, fluid multilevel reasoning systems.

6.1.5 Coherence

The final desideratum is that the KB be *coherent.* A multilevel KB is coherent if the conclusions it sanctions at different levels are logically consistent with one another. The conclusions derivable at different abstractions can be different—weaker, stronger, or incomparable because they refer to distinct domains—but they should not be contradictory.

Certain controlled forms of incoherence might be tolerable if the reasoner has special facilities to handle inter-level conflicts. For instance, some nonmonotonic inference mechanisms are designed expressly to prefer specificity, overriding the conclusions it derives from high-level premises in favor of more specific results (see especially research on inheritance systems and formalisms [120, 147]).

Inter-level coherence is usually difficult to guarantee. A purely reductionist KB is coherent by definition, though it may not always be feasible to verify reductionism. Definitional clarity makes it easier to maintain coherence, but as mentioned above, there is a strong tension between clarity and fluidity. Tension also exists between definitional clarity and nonmonotonicity [5, 49, 148], so admitting even regular incoherence is likely to compromise other desiderata.

6.2 Event Variable Knowledge Base

The action and event taxonomies of Chapter 3 constitute the taxonomic skeleton of SUDO-PLANNER's multilevel knowledge base. The KB is fleshed out with effect assertions (Section 5.4) relating actions and events across taxonomic levels.

Figure 6.3 presents a view of part of the KB supporting the running example. Effect arcs relate the simple taxonomy of primary surgical actions at the left of the diagram to the major event variables of interest. Among these are *MI* and *stroke* presence, a small cluster of *disease severity* variables, and *mortality*. All paths eventually lead to the special utility variable, *value*. The qualitative relation assertions specifying Figure 6.3 and the rest of the KB are listed in Appendix C.

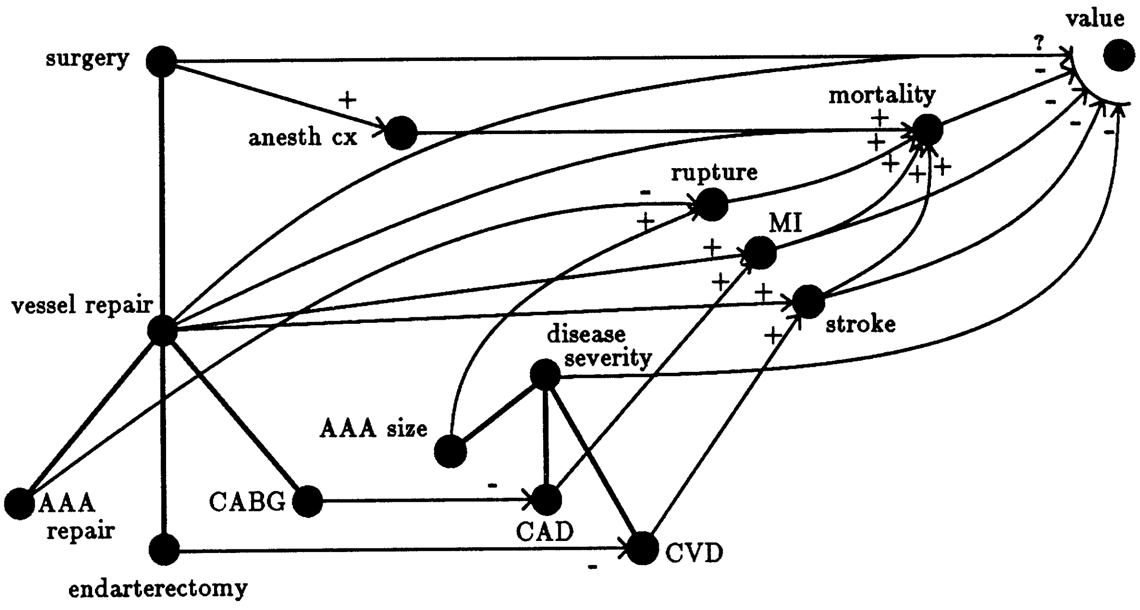


Figure 6.3: Part of SUDO-PLANNER's multilevel event variable KB.

As in previous figures, thick lines represent taxonomic relationships, and thinner ones, domain relations. Domain relation links in Figure 6.3 correspond to qualitative influence (S^{δ}) assertions in the KB. Qualitative synergy and Markov influence assertions are not shown. To avoid confusion between the two uses of qualitative relations, the diagrammatical conventions for KB graphs like Figure 6.3 differ significantly from those used in QPN figures.

Inspection of Figure 6.3 reveals that SUDO-PLANNER's KB is fluid, as we cannot partition the nodes into levels in a manner satisfying the rigidity conditions of Section 6.1.2. For instance, any partitioning would require that *surgery* and *vessel repair* be on different levels because of their taxonomic relation, and on the same level because of their mutual links to value.

Close inspection of the KB also reveals that it is incoherent (and therefore trivially non-reductionist). For example, there is a direct positive link from *vessel repair* to *stroke*, yet its subconcept, *endarterectomy*, has a negative effect on *stroke* through its influence on *CVD*. The apparent contradiction causes no difficulty in this case, since SUDO-PLANNER's model construction algorithm prefers the more specific path when considering *endarterectomy*. Incoherence with respect to *value*, on the other hand, would present a serious problem, causing SUDO-PLANNER's dominance prover to generate inconsistencies in the plan graph. The "?" links to *value* from *surgery* and *vessel repair* are in the KB specifically to avoid this potential error.

A full discussion of coherence requires an examination of how the KB is interpreted by SUDO-PLANNER's model construction procedure, described in Section 6.3 below. One important characteristic of the construction procedure is its treatment of inheritance. As described in Section 5.6, the intended meaning of an effect link from ev_1 to ev_2 is that each variable of type ev_1 affects some variable of type ev_2 (the universal/existential interpretation). Therefore, event variables inherit outgoing relation links from their taxonomic ancestors.

Figure 6.4 illustrates the use of inheritance in a fragment of the SUDO-PLANNER KB. In the linear taxonomy at the left, *aneurysm size* is a kind of *disease severity* variable because size is an indicator of severity for the disease "aneurysm presence."

The variable is further specialized by restricting the location of the aneurysm to the abdominal aorta (AAA). The same concept specialization relates the two rupture variables. All links in the figure appear as explicit assertions in the knowledge base.

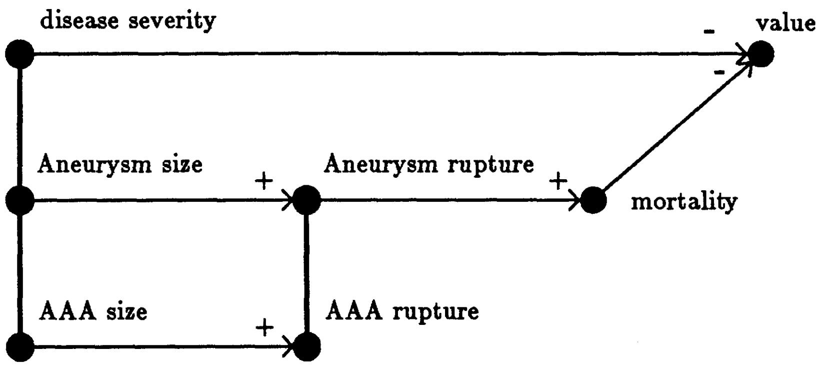


Figure 6.4: Fragment of the KB relating AAA size and value. Effect links are inherited downward in the antecedent taxonomy.

Further effect relations are implicit in the taxonomic relationships. For example, *AAA rupture* positively influences mortality, by inheritance from its parent, aneurysm rupture. In this case, inheritance provides everything that is known about the effects of *AAA rupture*.

In other cases, more specific knowledge supplements or replaces inherited information. *AAA size* positively influences *aneurysm rupture* by virtue of being an *aneurysm size*, but more specifically it influences the rupture of a particular type of aneurysm, AAA. Given the universal/existential interpretation, the local assertion is strictly stronger than that inherited from *aneurysm size*.

From the perspective of the destinations of effect links, variables inherit relations *upwards* in the taxonomy. For example, the link from *AAA size* to *aneurysm rupture* could also be inherited (at the destination end) upward from *AAA rupture*. The inheritance is redundant in this case, but in general could make a difference, for instance, if the link from *aneurysm size* did not exist.

A final possibility is that a direct link at one level could correspond to a more complex set of paths at another. For example, *aneurysm size* (and *AAA size* as well) exhibits a negative influence on *value* by virtue of being a *disease severity*. At a more specific level, *aneurysm size* influences *value* via a path through *aneurysm rupture* and *mortality*. Although the more detailed path leads to the same conclusion in this case, the direct relation leads to simpler and more efficient models. On the other hand, the detail is necessary for reasoning about interactions with other variables that share with *aneurysm size* segments of their influence path to *value*.

6.3 Model Construction in SUDO-Planner

SUDO-PLANNER's high-level behavior is a cycle of model construction and dominance proving, as illustrated by Figure 1.10. Figure 6.5 elaborates that view, revealing that the model construction cycle consists of the incremental evolution of a central QPN. From the perspective of the model constructor, the KB is an event variable graph in the general form of Figure 6.3. At each iteration, the constructor modifies the current QPN according to relations in the KB. The dominance prover analyzes the modified QPN, which then forms the basis for the next cycle of model construction. The process continues until no QPN modification operators are applicable (that is, the KB is exhausted) or it is explicitly halted by its invoker. The invoker can interrupt the model construction cycle at any time to inspect the plan graph, which is continually maintained by the plan space manager.

This section explains the left-hand half of Figure 6.5 involving the process of constructing and modifying QPNs based on KB relations. Chapter 7 explains the dominance-proving right-hand side.

To assemble a QPN from relations in an event variable KB, the model constructor selects a subset of the variables and links from the KB graph. Selection of this subset

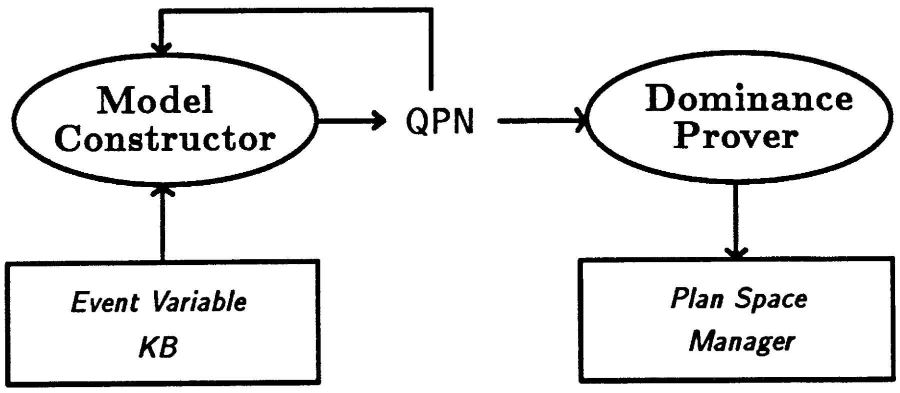


Figure 6.5: SUDO-PLANNER's model construction cycle. The QPN evolves through a sequence of incremental modifications.

is complicated by several factors, which form some of the central issues in model construction and modification.

Choosing variables. In a large knowledge base, it is imperative that selectivity be exercised in assembling the set of variables. For proper focus, we require some notion of relevance. I discuss the design of justified focus mechanisms in Section 6.4.

Choosing abstractions. Given that a particular concept is of interest, the constructor must decide at which taxonomic level to represent it with a QPN variable.

Changing abstractions. One way to modify a QPN is to change the abstraction level of some of its components. SUDO-PLANNER's elaboration mechanism (Section 6.3.1) attempts to update QPN relationships appropriately when changing levels.

Multiple inheritance. An event variable may have more than one source for inheriting effect relationships. The constructor must include mechanisms for selecting, merging, and resolving conflicts among inherited relationships.

Recording and merging conclusions from different models. In an iterative process of model modification, some of the conclusions drawn from the models might be useful in decisions about the subsequent evolutionary path. SUDO-PLANNER exploits this opportunity in a limited fashion, described in Section 6.3.2.

Because the dominance prover continually analyzes the QPN throughout its evolutionary process, each modification the model constructor applies must preserve validity. This requirement places strong constraints on the modification operators and the KB, and in fact, it is not completely met by SUDO-PLANNER's model construction mechanisms.

The model construction procedure includes two basic operations for modifying QPNs. SUDO-PLANNER alternates between *elaboration* steps that replace existing relationships with more detailed pathways, and *backward chaining* steps that extend the model to include additional related variables. The sections below present the procedures for these steps.

6.3.1 Elaboration

Elaboration introduces detail to a QPN. The multilevel operationality of SUDO-PLANNER's knowledge representation permits it to reason initially at high levels of abstraction, then refine the model to consider more specific concepts. Thanks to the KB's fluidity, refinement need not occur all at once; instead, the constructor can refine the QPN incrementally by elaborating individual links.

The elaboration process comprises three stages:

1. Choose a QPN link to elaborate.

2. Find a set of elaborating paths in the KB.

3. Merge the new structure into the QPN.

The starting point for the model construction process in our running example is the QPN of Figure 6.6a.[1] The initial QPN relates *AAA size* to *value* via the most general route in the KB. In this instance, the negative link derives from the KB relation between *disease severity* (an ancestor of *AAA size*) and *value*, as shown in Figure 6.6b.

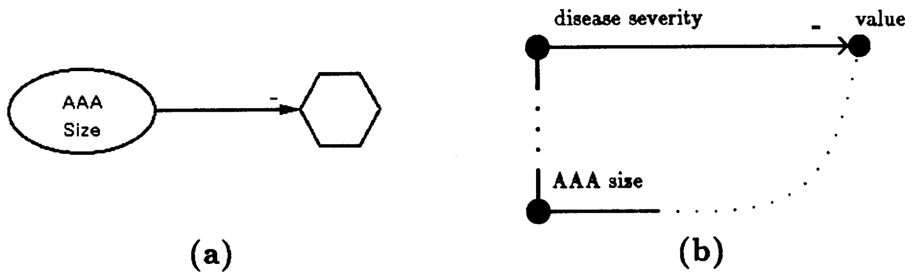


Figure 6.6: (a) Initial QPN for the running example. (b) The link corresponds to the most general effect of *AAA size* found in the event variable KB.

The rest of this section describes the three stages of elaboration, illustrating the process by application to this simple QPN and to more complex networks.

Stage 1: Choose Link to Elaborate

SUDO-PLANNER selects the QPN link to elaborate in a first-in/first-out order. It places links on a FIFO queue upon their addition to the QPN. If the queue is empty, the elaboration operator is inapplicable.

Stage 2: Find Elaborating Paths

Given a link between two QPN variables, an *elaboration* is a mutually compatible collection of paths connecting the corresponding variables in the KB. SUDO-PLANNER considers paths of length one or two only, thereby limiting the rapidity of model refinement. The individual paths may include both explicit and inherited KB relations.

[1]This QPN is identical to that of Figure 1.11. SUDO-PLANNER's graphical interface drew all QPN pictures for the running example appearing in this chapter (including Figure 6.6a).

Two paths are *compatible* if their relations are valid when combined in a QPN. For example, SUDO-PLANNER regards as incompatible combinations that contain versions of the same variable at different taxonomic levels. True semantic compatibility is not verifiable by the model constructor.

SUDO-PLANNER tags each link in the QPN with a pointer to the event variable in the KB from which it was derived. The link in Figure 6.6a, for example, originated from *disease severity* in the KB of Figure 6.6b. To elaborate this link, SUDO-PLANNER searches for paths from *AAA size* to *value* that derive from origins more specific than *disease severity*.

As Figure 6.7 illustrates, the search for elaborating paths in the KB graph is bounded by the origin variable (in this case, *disease severity*) and the KB variable currently appearing in the QPN (*AAA size*). The source of an elaborating path must be a descendant of the origin, so that it specializes the model, and an ancestor of the QPN variable, so the relations are inherited. The path may terminate at the current destination or at any of its descendants.

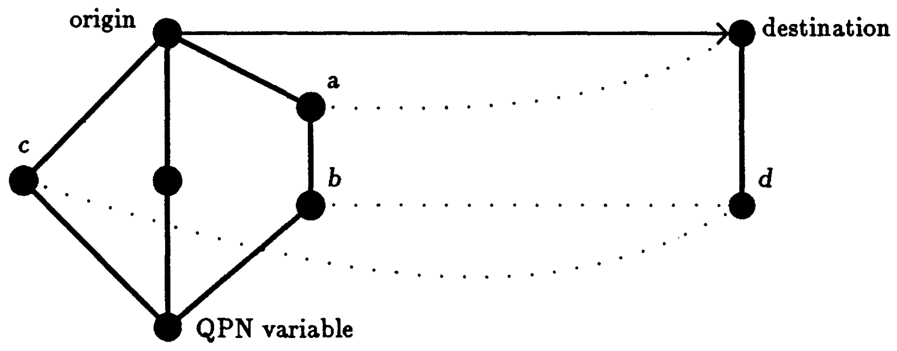


Figure 6.7: Search for elaborating paths. The dotted lines indicate that variables *a*, *b*, and *c* are sources of candidate elaborations.

There may be several candidate elaborations. Suppose, for example, that in the KB of Figure 6.7 elaborating paths emanate from nodes *a*, *b*, and *c*. SUDO-PLANNER prefers the most general elaborations because the more specific ones remain reachable in subsequent elaboration steps. Therefore, it removes the paths from b from consideration. But since a and c are not taxonomically related, SUDO-PLANNER includes the union of their elaborating paths.

This strategy is a form of multiple inheritance. Because there is no guarantee that the relations inherited from different paths are consistent or compatible, this merge operation (as well as the process of merging the elaboration into the QPN, stage three) is a source of potential error in SUDO-PLANNER's model construction algorithm.

The path selected to elaborate the link in our initial QPN (Figure 6.6a) is the chain

$$\textit{aneurysm size} \longrightarrow \textit{aneurysm rupture} \longrightarrow \textit{value}. \tag{6.1}$$

Aneurysm size is the most general variable that is a descendant of *disease severity*, an ancestor of *AAA size*, and has a path of length at most two to *value*. (The direct link from *aneurysm rupture* to *value* is inherited from an ancestor, not shown in the KB fragment of Figure 6.4.) The elaboration can be interpreted as an explanation of the original link: *AAA size* is undesirable because it positively influences *aneurysm rupture*, which is undesirable.

Stage 3: Merge New Structure

If SUDO-PLANNER finds an elaboration, it removes the original link from the current QPN. It then proceeds to the final stage of elaboration: merging the chosen elaborated paths into the model. The merge task is to determine how to modify the variables and relationships in the QPN to include the new structure.

For our initial QPN, merging is easy. A new variable, *aneurysm rupture*, is introduced to the QPN, and the elaboration path (6.1) simply replaces the original link.

The KB variable *aneurysm size* is recorded as the origin of the first link, though the QPN variable remains *AAA size*. SUDO-PLANNER adds the two new links to the FIFO queue for possible further elaboration. Figure 6.8 displays the final result of the elaboration step.

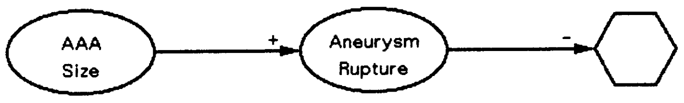


Figure 6.8: An elaboration of the initial QPN.

In the general case, merging is more complicated. When introducing a new variable, SUDO-PLANNER must connect it to all existing QPN variables—not just those on the new paths—according to relations encoded in the KB. Performing the update correctly is tricky because the new variable may be connected to existing ones via complex and possibly redundant pathways.

When a new variable is introduced to the network, SUDO-PLANNER connects it with all existing QPN variables that are directly related to it, explicitly or via inheritance, in the KB. For example, the variable z is linked to current variable x upon introduction to the QPN of Figure 6.9a. Variables z and y remain unconnected because there is no direct link between them in the KB graph.

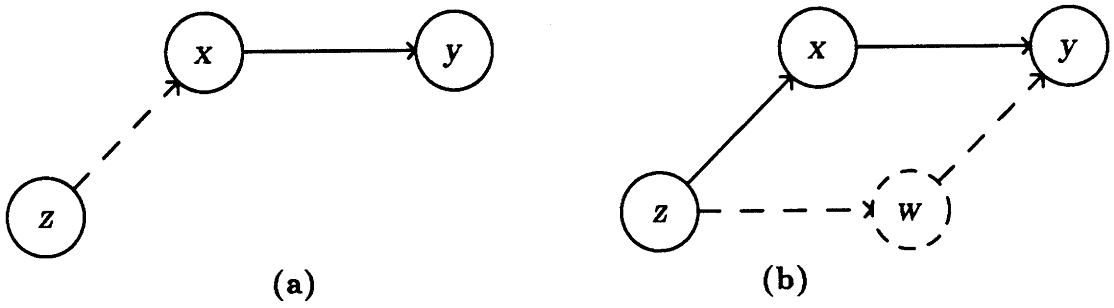


Figure 6.9: (a) Introducing z to the network. (b) The KB contains no direct relation between z and y, but includes a path through w, not currently in the QPN.

Suppose, however, that the KB contains a path from z to y via variable w, which is not represented in the current QPN. The network of Figure 6.9a then leads to the false conclusion that y is independent of z given x. We can remedy this by introducing w to the QPN along with z, adding the appropriate links as illustrated in Figure 6.9b. Although safer, this policy is unsatisfactory because introduction of a single variable is liable to trigger the migration of a large fraction of the KB into the QPN.

SUDO-PLANNER preserves the incrementalism of QPN modification by ignoring indirect connections of the sort displayed in Figure 6.9b. Typically (and necessarily to avoid unsound inferences), pathways in the KB have corresponding one-step summaries at higher taxonomic levels. In this case, we would expect z to inherit a direct link to y from some ancestor. To be conservative, we could add these *spanning links* [107] automatically, although the implemented SUDO-PLANNER includes no KB pre-processing. The spurious links included due to such a policy are of little consequence because if truly spurious they will be removed upon subsequent elaboration.[2]

When possible, SUDO-PLANNER combines taxonomically related variables. If the newly introduced variable has an ancestor in the QPN, the merge procedure tests whether the existing relations are compatible with the specialized variable. Because outgoing links are inherited, compatibility holds exactly when the links into the existing ancestor variable hold for the new variable. If compatible, the new replaces the old. Otherwise, SUDO-PLANNER attempts to merge the new relations with the old variable. Here, compatibility depends only on the outgoing relations. If the merge fails in this direction as well, SUDO-PLANNER maintains both variables in the QPN. The two will be merged eventually if subsequent QPN modification presents the opportunity.

For example, the elaboration search situation of Figure 6.7 yielded paths from a to the existing variable *destination*, and from c to its descendant, d. SUDO-PLANNER

[2]Elaboration results in simple removal if the elaborating structure is already present in the QPN.

first tries to replace *destination* with d, succeeding if all incoming links to *destination* hold for d as well. Failing that, it attempts the converse merge. In this case, d is replaceable by *destination* because it is a new variable with no outgoing links. Subsequent elaboration steps will trigger the reintroduction of d, in particular when the path from a is specialized to originate from b.

Finally, qualitative synergies are also added to the QPN during the merge stage. Whenever a node has two or more incoming links, SUDO-PLANNER checks the knowledge base for an asserted or inherited synergy among the predecessor variables. Recall that SUDO-PLANNER interprets the absence of an explicit synergy link in the QPN as "?" by default.

6.3.2 Backward Chaining

The QPN for the running example as developed to this point (Figure 6.8) is not very interesting from the perspective of planning. The semantics of qualitative influences entail that small AAAs are preferable to large ones, but tell us nothing about what we should do about them. Furthermore, no amount of elaboration applied to this relation sheds any light on action.

The second basic QPN modification operator, *backward chaining*, extends the scope of the model. In a backward chaining step, SUDO-PLANNER searches for variables in the KB that affect a particular existing QPN variable. The mechanics of this step are straightforward: choose a variable to extend back, find its predecessors in the KB, and merge the new structure according to the merge procedure described above.

The first attempt at backward chaining on the running example produces no modification because the chosen variable, *AAA size*, has no predecessors. SUDO-PLANNER next applies an elaboration step, which replaces *aneurysm rupture* with

AAA rupture. Backward chaining on *AAA rupture* finally yields a significant QPN modification, shown in Figure 6.10.

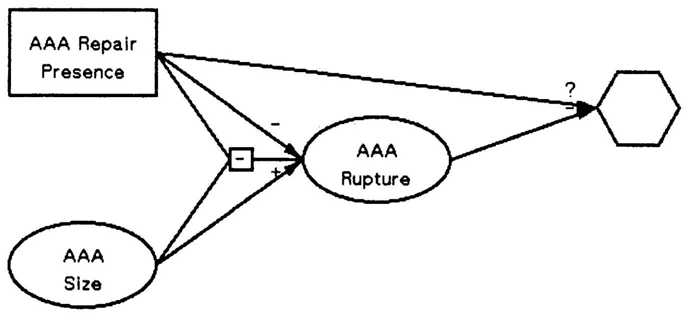


Figure 6.10: QPN after backward chaining on *AAA rupture*.

The new QPN has the same variables as that of Figure 1.8, though it lacks the positive synergy between *AAA repair* and *AAA rupture* on *value*. Given this synergy, the dominance prover could establish that *AAA repair* is positively synergistic with *AAA size*, and therefore that the optimal repair policy is to fix the aneurysm iff it is larger than some threshold size. However, the synergy is unavailable to SUDO-PLANNER at this stage in QPN development. In the current model, *AAA repair*'s relation to *value* is inherited from its ancestor *surgery* (see the KB graph of Figure 6.3). The synergy is simply not valid at this high level. Further elaboration of the model combined with QPN inference yields the desired result later in the model construction process.

The initial stage of backward chaining is the choice of a variable to extend back. SUDO-PLANNER selects arbitrarily among the eligible, or *extensible*, variables. If no variables are extensible, the backward chaining operator is inapplicable. A variable is considered extensible if:

1. it is not a decision variable (these have no predecessors),

2. it has not already been extended back, and

3. it is one of the specially designated *focus variables*, it is a successor of a focus variable, or it is $\pm$ synergistic with a focus variable on *value*.

I describe the concept of a focus variable and justify its use in Section 6.4.2. Informally, focus variables are those deemed relevant based on their role in the case description or in plans under consideration. The heuristic application of focus variables to restrict the variables eligible for backward chaining directs the course of QPN evolution.[3] Although this control is not necessary to run the main example, a focus mechanism of some sort is crucial for tractable model construction from large KBs.

To implement the extensibility test, SUDO-PLANNER records instances of synergy with a focus variable as it derives them in intermediate computations of the dominance prover. This is the only semantic information about domain relations (and the only type of QPN inference) that the model construction procedure exploits. More sophisticated application of intermediate results might improve efficiency and control, but would also exacerbate the undesirable consequences of incoherence in the multilevel KB.

The second stage of backward chaining is to retrieve the QPN variable's predecessors from the KB. SUDO-PLANNER simply traces back links in the KB graph. The back extension does not consider inherited predecessor relations because these will be uncovered in subsequent elaboration.

The final stage is to introduce the predecessors to the QPN. The procedure to merge structure from backward chaining is equivalent to the merge stage for elaboration described in the previous section.

The backward chaining strategy adopted by the model construction procedure imparts special significance to the directionality of influence relations in the event variable KB. Whether we assert that a affects b or that b affects a in the KB strongly

[3]The application of focus variables is similar in spirit to the use of *focus objects* in McAllester's ONTIC theorem prover [87].

influences the shape of QPNs produced by SUDO-PLANNER, even though the semantics of qualitative influences does not dictate this choice. As Section 4.4.4 demonstrates, QPN influences can be reversed without affecting their sign, although the reversal may change other relations in the network. SUDO-PLANNER fails to exploit this flexibility—in fact, the implementation never attempts to reverse influences—despite its demonstrated usefulness in traditional decision modeling [125].

6.3.3 Variable Mitosis for Markov Influences

An important type of backward chaining occurs when the predecessor exerts a Markov influence on the current QPN variable. As Section 5.5 describes, a Markov influence represents an effect over time, inducing a relationship between the action and the values of its affected variable before and after execution. To represent this situation in a QPN, we require separate variables for the "before" and "after" values. SUDO-PLANNER modifies the QPN by dividing the affected variable in two and updating relations as appropriate.

The variable division process, or *mitosis*, is illustrated in Figure 6.11. SUDO-PLANNER encounters a Markov influence in the process of backward chaining on *CAD extent*. The QPN fragment of Figure 6.11a displays the variables connected to *CAD extent* prior to interpreting the Markov influence.

Figure 6.11b illustrates the Markov influence itself. *CABG* influences *CAD extent*, in a manner synergistic with the prior value of the variable. The mitosis process translates this implicit distinction among the two *CAD* variables to an explicit separation in the QPN. The resulting network fragment, shown in Figure 6.11c, contains the variables *CAD-1* and *CAD-2*, denoting the *CAD extent* before and after *CABG*, respectively. The previous predecessors of *CAD*—*CAD history* and *catheterization result*—are attached to *CAD-1*, and the previous successor—*MI*—to *CAD-2*.

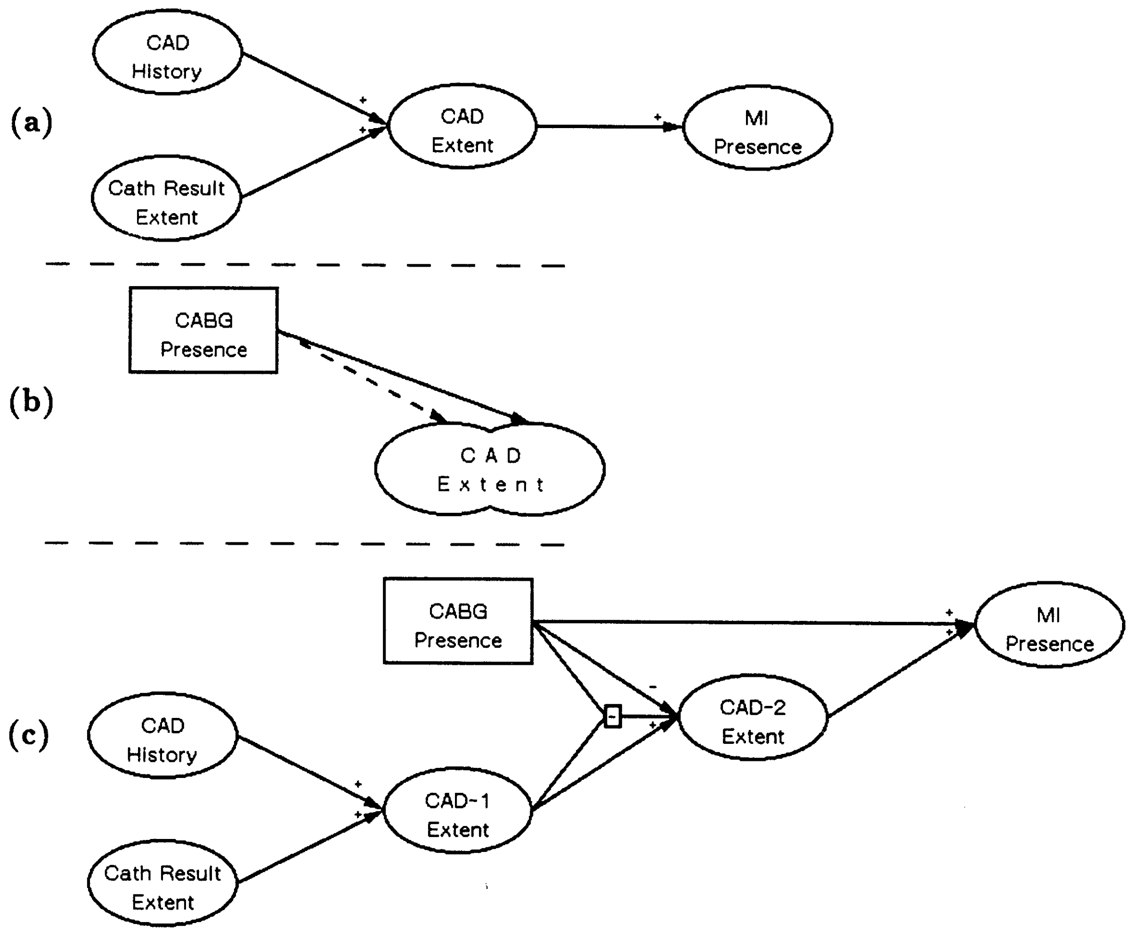


Figure 6.11: Variable mitosis. (a) The situation before mitosis. (b) The Markov influence of *CABG* on *CAD* implicitly refers to two *CAD* variables. (c) The final result: *CAD* is split into *CAD-1* and *CAD-2*, and relationships are appropriately distributed among the two.

The Markov influence explicitly specifies the influence of *CABG* on *CAD-2*, and its synergy with *CAD-1*. Implicit in all Markov influences is the positive relationship between the "before" and "after" variables.

The split variables are divided further if they are the object of other Markov influences. In general, the QPN may contain an arbitrary Markov chain reflecting a temporal sequence of values of the same event variable.

The presence of multiple QPN variables corresponding to a single KB variable complicates the model construction process slightly. When merging structure (during elaboration or backward chaining) involving links *into* such a variable, SUDO-PLANNER selects the head of the Markov chain. Structure containing paths *out of* a multiply represented variable is referred to the tail variable. These conventions preserve the Markovian character of the QPN's dependency graph.

6.3.4 The Model Construction Procedure: Conclusion

Having completed the description of SUDO-PLANNER's model construction process, we are now in a position to evaluate it in terms of the five basic desiderata for multilevel reasoning presented in Section 6.1. Of the five, SUDO-PLANNER scores better on non-reductionism, fluidity, and multilevel operationality than it does on definitional clarity and coherence. A strict judge, however, would find the system wanting with respect to all these criteria.

The event variable KB is clearly non-reductionist and fluid. However, in adopting a style of top-down refinement, SUDO-PLANNER's model construction procedure does not take full advantage of the flexibility offered. Because it generally does not record intermediate results, the dominance prover is incapable of combining partial conclusions from separate decision models. Thus SUDO-PLANNER will miss a result following from a set of KB relations unless it incorporates those relations together in a QPN.

The addition of high-level spanning links as a conservative knowledge engineering policy compromises the goal of multilevel operationality. Because SUDO-PLANNER removes the spurious relations only upon elaboration, it ends up deriving conclusions at a lower level than is strictly necessary.

As mentioned above, SUDO-PLANNER's event variable KB is neither well-defined nor guaranteed to be coherent. Further study of the semantics of KB assertions would advance definitional clarity (see Section 5.4.1), as would a more formal characterization of the model construction procedure. A better understanding of these issues is a prerequisite for an analysis of KB coherence.

The practical implication of this evaluation is that knowledge engineering is difficult. In my experience developing SUDO-PLANNER, the effect of KB modifications on the sequence of models generated was often unpredictable. Rules of thumb for debugging the KB, such as adding high-level single-step summaries of complex pathways, were discovered by trial-and-error. Only later was I able to rationalize the practice in terms of its interaction with the model construction procedure.

Some of the shortcomings of the model construction procedure are probably amenable to incremental solutions. For example, we could increase fluidity and coherence by permitting multiple, mutually exclusive relation sets at a single abstraction level. Such an extension requires a generalization (and a formal tightening) of the notion of compatibility regulating the merging of structure into QPNs. Section 7.5.1 discusses the possible use of spanning relations, a special case of this idea, in a qualitative approach to tradeoff resolution.

This critical view of SUDO-PLANNER's model construction performance does not reflect discouragement about the task of constructing decision models from large multilevel KBs. SUDO-PLANNER represents a first attempt to automate this task, which was intentionally framed in terms of highly ambitious (and somewhat conflicting) desiderata in order to uncover the performance boundaries. By identifying and emphasizing its limitations, I aim to stimulate research on improvements to SUDO-PLANNER's approach to knowledge representation and model synthesis.

6.4 Focus of Attention

For large KBs, controlling model construction is a significant and difficult problem. Directing model synthesis effort toward the most relevant regions of the KB requires some sort of *focus* mechanism. Unfortunately, it appears to be difficult to justify a focusing strategy based on the decision-theoretic principles underlying SUDO-PLANNER. The sections below discuss the basic issue of justified focus and present the rationalized, albeit unjustified, focus mechanism implemented in SUDO-PLANNER.

6.4.1 Justified Focus

In the most straightforward implementation of the model construction procedure, SUDO-PLANNER would start from a QPN containing only the value node and proceed to apply backward chaining and elaboration steps until it exhausted the KB or its computational resources. This unfocused approach proves unsatisfactory for any real problem. A planner with a moderately large medical knowledge base would be obliged to consider the advisability of actions ranging from taking blood pressure to prophylactic heart transplants for every patient. We need a method that focuses attention without recklessly overlooking valid therapeutic opportunities.

Ideally, we would like to ground our focus mechanism in relevance principles that justify selectivity in the application of knowledge. A sound procedure for focus of attention ignores only facts that are *irrelevant* to the task at hand [139]. For the planning task, knowledge is irrelevant iff it has no bearing on the optimal strategy.

One approach to justified focus assumes that at any time the agent's plan is optimal given the information available at that time. Under this assumption, the planner need consider only strategy modifications warranted by changes in its information state. In a medical context, the planner in effect assumes that it has been the patient's physician up to the current time, so the current plan must be what it would have recommended given the current information. When new information arrives, it is necessary to reevaluate only those previous conclusions that depend on changed facts [85].

This "status quo optimality" heuristic meshes well with the planning and execution model of Section 3.1. If the planner has converged on a singleton plan class specifying the appropriate action in every contingency, then planning is finished and only execution remains. More realistically, the planner produces plan classes specifying action for only the immediate contingencies, necessitating further planning for the unanticipated observation patterns that make up the majority of possible futures. When an unanticipated situation occurs, the planner needs consider only its difference from an anticipated situation in adapting its corresponding plan to meet the new situation.

Unfortunately, the focusing power of this heuristic is often disappointingly weak. The smallest changes in situation may dictate arbitrary alterations in strategy, requiring the modification of actions seemingly unrelated to the situation change. In medicine, for example, the advisability of a therapeutic action typically depends on the broad concept of overall state of health. For instance, consider a patient who is a candidate for a heart transplant. Suppose that in the current situation, the planner determines that the patient narrowly qualifies for this dangerous and expensive procedure. Next consider a new finding that is unrelated to heart disease or to any other cardiovascular concept but that has a slight negative influence on the patient's life expectancy. Even though its health risk is unrelated, the new finding reduces the potential benefit of a heart transplant because it lessens the survival time to be gained. Because the patient is a marginal transplant candidate and the new finding has no effect on the cost of the procedure, the observation may be grounds for reversal of the original decision.

These kinds of situations are ubiquitous in medicine, and I suspect they are common in other domains as well. In a comprehensive KB all events are related, if only because they all have some connection to *value*. This observation suggests that the opportunities to focus a planner based on true irrelevance are rare and therefore we need to explore other grounds for allocating reasoning resources in the planning task.

6.4.2 Focus Variables

SUDO-PLANNER employs a simple focus mechanism based partly on the status quo optimality heuristic discussed above. The method designates particular QPN variables as *focus variables*, thereby controlling the invocation of backward chaining as described in Section 6.3.2.

Initially, the set of focus variables comprises those marked as *changed* in the input problem description. This is a direct reflection of the assumption described above: if the current strategy is optimal, only changes in the status quo are relevant to planning. SUDO-PLANNER maintains focus by adding to the list any variable that is mentioned in dominance conditions derived in the course of planning. These variables are worthy of further inference because new dominance results about actions and observed events that appear in undominated plan classes refine the plan graph.

A QPN variable is eligible for backward chaining if it is synergistic with a focus variable on *value*. To understand the rationale for this criterion, consider the QPN of Figure 6.12. There, f is the sole focus variable and c is a candidate for backward chaining. The two variables are synergistic in direction δ_1 on *value* in this reduced

QPN, that is, $Y_U^{\delta_1}(\{c,f\})$. Let d be some predecessor of c in the event variable KB, not in the current QPN. The backward chaining criterion dictates that c be extensible if $\delta_1 \in \{+,-\}$, but not if $\delta_1 \in \{0,?\}$.

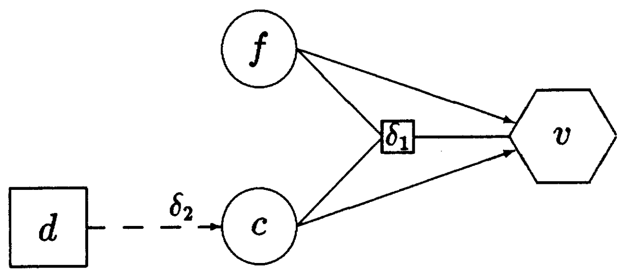


Figure 6.12: Synergy with a focus variable. $Y_U^{\delta_1}(\{c,f\})$ holds in the QPN containing variables c, f, and v. Backward chaining on c to introduce d to the network can lead to interesting results only if $\delta_1 \in \{+,-\}$.

If $\delta_1 \in \{+,-\}$, then both c and its predecessor d are of interest to the planner. The synergy entails a nontrivial monotone decision relation for f and c, and for f and d as well if $\delta_2 \neq ?$. This result could be significant for one of two reasons:

1. Variable f has changed, therefore synergy with a decision variable dictates the direction to alter the status quo plan.

2. Variable f appears in the description of an active plan class, therefore the monotone decision property dictates a refinement to the plan graph.

Because f is a focus variable, one of these reasons must be in force.

Conversely, if $\delta_1 \notin \{+,-\}$, we gain nothing (with respect to f) by backward chaining on c. If $\delta_1 = 0$, then c is irrelevant to f, as is d. This is a strong case of justified focus. On the other hand, if $\delta_1 = ?$ then c and its predecessors generally are relevant. However, the ambiguous sign prevents SUDO-PLANNER from determining in what way they are relevant, and further backward chaining will never resolve the ambiguity. Hence, the model construction procedure may as well not pursue this path in the KB. To SUDO-PLANNER, futility serves as a focus justification on par with irrelevance.

To implement the extensibility criterion, SUDO-PLANNER records all known synergies with focus variables (on *value*) as they are derived in the course of dominance proving. The situation Figure 6.12 depicts holds for the QPN in some state of reduction, not necessarily in any constructed QPN or in the KB directly.

6.5 Related Work

Section 6.3.4 includes an evaluation of SUDO-PLANNER's model construction procedure in terms of the desiderata for multilevel reasoning presented in Section 6.1. In this section, I take a broader perspective and contrast the SUDO-PLANNER approach with related work on model construction and multilevel reasoning. The two topics are reviewed separately below, since SUDO-PLANNER is the first program designed to construct decision models from a multilevel KB.

6.5.1 Decision Model Construction

Most work on decision model structuring has been carried out with the intent of aiding human decision analysts (see Humphreys and McFadden [63], Keeney [68], and von Winterfeldt [151] for a mixed sample). There is a significant demand for such aids, especially as the availability of inexpensive, user-friendly software packages for personal computers makes DA technology accessible to a widening group of potential analysts. The need is particularly critical for those analysts who have less experience with the methodology and a lower level of familiarity with the underlying decision-theoretic concepts. In fact, we have incorporated some of the qualitative dominance ideas from this dissertation into BUNYAN, a program for critiquing human-generated decision models [160]. But despite some common concerns, the issues faced in attempting to fully automate the model construction task differ significantly from the problems of aiding human modelers. In this section I discuss the few projects that have specifically addressed themselves to automated knowledge-based formulation of decision models.

In Holtzman's "intelligent decision systems" [57], the domain knowledge is primarily in the form of a general decision model, encoded as an influence diagram with assessment functions for each node of the graph. Constructing a model for a particular decision is largely a matter of reducing the template model that is built into the program. This is essentially the decision-model-as-KB approach that I rejected in Section 1.5 for reasons of scalability.

The alternative advocated in this work calls for dynamic assembly of decision models from a knowledge base of primitive components. An early example of this kind of behavior was exhibited by a program for Hodgkins disease, described by Rutherford et al. [116], that dynamically constructs decision trees from a KB specifying a variety of diagnostic tests and treatments. Its flexibility lies in the possibility of modifying the set of available tests and treatments on a case-specific basis. Hollenberg's Decision Tree Builder (DTB) [56] also generates decision trees using a medical knowledge base of diseases, tests, and treatments. Unlike the Hodgkins program, DTB is intended to handle a broad range of medical decision problems. Consequently, its representations are considerably more general, and its tree generation, correspondingly more flexible. A disease may be parameterized by *attributes*, which in turn may influence the applicability of various actions as well as the values of probabilities and utilities in the model. Tests and treatments may indicate or modify the values of disease attributes.[4] A simple control structure directs tree construction, employing a model

[4]Incidentally, difficulties with this mechanism first brought my attention to the problem of distinguishing the values of event variables in different temporal relation to their affecting actions (J. P. Hollenberg, personal communication), handled by Markov influences and variable mitosis in

of patient states for bookkeeping purposes.

With this kind of generation approach, it is extremely difficult to escape from exhaustive consideration of a combinatorial space of plans and events. The programs must construct strategies that include every action identified as potentially beneficial, and model every event identified as potentially relevant. The desire to avoid this behavior was the primary motivation for developing abstraction mechanisms for SUDO-PLANNER (see Section 1.6). Control of model construction remains important even with a multilevel representation, as discussed in Section 6.4.

The model constructor closest in spirit to SUDO-PLANNER is Breese's ALTERID system [9, 10]. ALTERID constructs numeric influence diagrams from a KB of logical and probabilistic assertions in response to specific queries. Aside from the contrast in precision between qualitative and numeric probabilistic relationships, the KB language ALTERID uses differs from SUDO-PLANNER's in two important respects:

1. SUDO-PLANNER's KB has an explicit taxonomic dimension. Relations at different levels of abstraction may coexist in the ALTERID KB, but they are not treated specially by the reasoner.

2. ALTERID bundles predecessors of a variable together and permits multiple exclusive groupings. Backward chaining retrieves one of the specified predecessor sets, rather than a collection of individually asserted predecessors as in SUDO-PLANNER.

Because the predecessors are specified and retrieved as a group, ALTERID avoids some of the compatibility problems faced by SUDO-PLANNER. Manipulation of these compound units provides the knowledge engineer with greater control over the networks ultimately produced, at the expense of flexibility obtained by reasoning about the individual predecessors. ALTERID's provision for multiple exclusive predecessor sets

SUDO-PLANNER (see Section 6.3.3).

is not expressible in SUDO-PLANNER, although some exclusivity can be represented by separating the structures by taxonomic levels. Section 7.5.1 discusses the possibility of extending SUDO-PLANNER's event variable knowledge representation to allow alternate relation patterns at the same level of abstraction.

Finally, in previous research I considered special mechanisms for constructing the outcome and preference portions of a decision model [156, Chapter 8]. Although this work is largely compatible with SUDO-PLANNER, I have not included any of it in the implementation. In particular, the facilities for choosing representations for health outcomes based on terminological transformations [157] should be applicable to model construction in general.

6.5.2 Abstraction

Abstraction has been studied in a variety of AI contexts. Section 3.2.1 discussed research in planning with abstraction and its relation to SUDO-PLANNER's knowledge representations for plans and actions. Researchers have also developed formal theories of abstraction *per se* [55], intended to support a variety of AI tasks. In this section I briefly review (in terms of the desiderata presented in Section 6.1) some approaches toward multilevel knowledge representation within the "causal" modeling paradigm for diagnostic reasoning. The commonality of these systems with SUDO-PLANNER lies in their explicit use of taxonomic and domain relations; the causal nature of the domain relation is inessential to this discussion. This appears to be a minimal requirement for evaluation in terms of the multilevel reasoning desiderata. Direct comparison of these programs to SUDO-PLANNER is not possible, however, as they were designed for significantly different tasks.

In a *compiled knowledge* approach [11], all of the relations encoded at high levels could in principle be derived from deeper knowledge about low-level concepts. Compilation (typically a manual operation) improves reasoning efficiency by replacing common, complicated inference patterns by direct associations. A truly compiled KB is necessarily reductionist. The system as a whole is not really multilevel unless the reasoner also has access to the deep knowledge.

In Section 6.1.2, I cited ABEL [98] and CADUCEUS [107] as examples of rigid and fluid systems, respectively. ABEL is also reductionist; it ultimately performs all of its causal reasoning at the lowest available level of abstraction.[5] Partitioning the KB into levels improves definitional clarity and thereby enhances the prospects for coherence. CADUCEUS, like SUDO-PLANNER, is non-reductionist, fluid, and operational at all levels. Although the semantics of its KB are not perfectly clear, and the KB itself is not guaranteed to be coherent, CADUCEUS appears not to suffer as badly in these respects as SUDO-PLANNER. Because it reasons directly from the KB, CADUCEUS imposes the closed-world assumption globally, rather than locally on models constructed from the KB. Finally, the hybrid ABEL/CADUCEUS approach that Patil and Senyk propose [99] appears to inherit the desirable characteristics of CADUCEUS listed above.

6.6 On Constructed Models

In this section, I step back from the specifics of QPNs and SUDO-PLANNER's methods for constructing them to examine some fundamental questions about synthetic decision models. In particular, I look at the nature of decision models, their relation to normative decision theory, and the implications for the model construction enterprise.

[5]However, given incomplete information, ABEL's higher levels apply knowledge based on implicit assumptions not accessible to levels below. In this situation, the program can exhibit non-reductionist behavior.

6.6.1 Models and Closed Worlds

One plausible operational definition for "model" is "a structure on which a closed-world assumption is invoked." In Hewitt's terminology, decision and other models are *microtheories*, the closed components of an "open system" [54]. All deduction in open systems is applied to microtheories, which are constructed by extralogical mechanisms.

Whether or not their designers employ the terminology of models or microtheories, systems that interleave inference under closed-world assumptions with non-logical processing perform what I have been calling "model construction." An example is Forbus's Qualitative Process Engine (QPE) [36], a program that constructs qualitative process models from a KB of process descriptions and a specification of potentially active processes. As in QPNs, derived relations among variables in qualitative process models depend on an assumption that the set of influences is completely known. QPE accumulates active relations in *closed-world tables* before assuming the closure required to compute the net result.

6.6.2 Decision Models

The decision-theoretic formulation of a decision problem postulates an abstract set of available acts, corresponding to the set of all plans Ω, and a set of possible states of nature Θ. The axioms of expected utility [108, 121] imply the existence of a utility function $u : \Omega \times \Theta \rightarrow \Re$, such that for all $\pi_1, \pi_2 \in \Omega$,

$$\pi_1 \succeq \pi_2 \Leftrightarrow E_\theta[u(\pi_1, \theta)] \geq E_\theta[u(\pi_2, \theta)]. \tag{6.2}$$

The expected utility property (6.2) is of fundamental theoretical importance for several reasons, including:

- It grounds the theory of subjective probability in rational decision making.

- The existence of numerical representations is analytically convenient for mathematical decision theorists.

Practitioners of applied decision theory, or *decision analysis* (DA) [62, 109], cite another implication of (6.2):

- The theory can be directly implemented in prescriptive aids for practical decision making through the assessment of formal probabilistic models and utility functions.

A *decision model* is a literal translation of the theoretical entities of (6.2) to concrete symbolic structures. In specifying decision models, decision analysts usually separate the probability and utility components by identifying a special outcome descriptor c (the *consequence*, a function of π and θ) sufficient for describing preferences. The utility model is then $u(c)$ and the probabilistic model is expressed as $f_c(\cdot|\pi)$ without explicit reference to θ.

Although modeling schemes differ in form and expressive power, all represent a choice situation in terms of the basic decision-theoretic concepts of alternative acts, uncertain events, and preferences. In standard DA methodology, analysts assess these elements by interpreting real and hypothetical choices of the decision maker in terms of (6.2). As Tversky [149] and others have noted, this procedure presumes the theory has some descriptive validity, a matter deserving considerable skepticism.

Stated in the abstract, the components of decision theory are completely general, and its arguments normatively compelling.[6] Despite their structural correspondence, however, the generality and normativeness of the basic decision-theoretic formulation

[6]This is not to deny that the tenets of Bayesian decision theory have been and continue to be highly controversial—in statistics and philosophy as well as AI. I will not recount the dispute here; nevertheless, my opinion is that the majority of the objections apply to narrow applications of the theory rather than to the fundamentals themselves. The majority of objectors, of course, have differing opinions. For elaboration, see the discussion below and previous writings [161].

does not automatically transfer to particular decision models. The rational agent axiomatized by decision theory is an idealization; decision models can only approximate its preferences and beliefs.

The next section explores the nature of this idealization, using Savage's concept of a "small world." Section 6.6.4 examines the implications of these observations for the enterprise of building computational agents based on decision-theoretic principles.

6.6.3 Small Worlds

Decision models fall short of capturing the ideal agent for two related reasons. First, as *models*, they are imperfectly related to the reality they are intended to represent [133]. Second, the decision problems they model constitute only a slice of the overall decision situation faced by the agent. This latter issue is the problem of small worlds.

Savage points out that in the decision-theoretic formulation, the ideal rational agent [121, page 83]

> ...has only one decision to make in his whole life. He must, namely, decide how to live, and this he might in principle do once and for all.

Choice of this lifelong policy is the agent's *grand-world* decision problem. Decision theorists and analysts following (and including) Savage rightfully regard grand-world decision modeling as unrealistic, and instead focus on isolated decision situations called *small worlds*. The problem of small worlds is to justify this focus—to determine when it is legitimate to apply the machinery of decision theory to isolated slices of the grand-world decision.

Savage attacked this problem formally by describing the correspondence between the small and grand worlds. Because the grand world is a refinement of the small world, states in the latter map to classes of states in the former. Thus, small-world consequences correspond to distributions over grand-world consequences. A small world that obeys the axioms of expected utility is called a *pseudo-microcosm.* Decisions based on a pseudo-microcosm are valid when the utility of small-world consequences is equal to their expected utility in the grand world, and the probability of small-world states is the same as the grand-world probability of the corresponding class. If these correspondences hold, then the small world is called a *microcosm.* Savage shows that pseudo-microcosms need not be microcosms, though he expresses the opinion that "the possibility of being taken in by a pseudo-microcosm that is not a real microcosm is remote" [121, page 90].

Details of the technical criteria for determining whether a small world is microcosmic are inessential to the present discussion (in fact they have not been completely characterized).[7] Examination of decision modeling experience suggests that even if the small worlds can be formally cast as microcosms, the myopic view from the small world is likely to result in models that fail to respect features that are expressible only in more refined worlds. For example, failure to account for portfolio effects is one way to get "taken in by a pseudo-microcosm." Decomposing a grand-world decision into a series of small-world choices often leads to models that do not reflect the reduction in risk due to diversification from the perspective of the global portfolio choice.

Whether we attribute the portfolio effect distortions to pseudo-microcosmic effects or simply to inaccurate modeling, the source of the problem lies in the isolated nature of small-world decision making. As Section 6.6.1 notes, reasoning with decision models requires a closed-world assumption; closing a small world is tantamount to ignoring features relevant only in the grand world. Thus, choosing the scope of small worlds to avoid bias is critical to the legitimacy of decision modeling. Unfortunately, the basis for such a choice is not well-understood by decision theorists.

SUDO-PLANNER's dominance-proving architecture is unique in embracing the con-

[7]For a lucid presentation and a concrete example of a non-microcosmic pseudo-microcosm, see Shafer's reexamination of Savage's arguments, as well as the commentaries on Shafer by Lindley and Dawid [128].

cept of grand-world decision making. A dominance-proving planner, as described in Section 2.4.1, is a grand-world decision maker if its plan space comprises life-long policies rather than short-term strategies. Such a plan space is conceivable within a constraint-posting framework because—unlike the basic decision-theoretic formulation—acts are composite objects described as combinations of primitive acts. Decisions are distinctions among plan classes rather than selections of particular acts. In decision theory, acts are atomic and decisions are selections. Hence, it is not possible in decision theory to make a near-term decision without either committing to long-term decisions or proceeding as though the near-term is the only term via a small-world construction. A constraint-posting planner, in contrast, can make a near-term decision without the pretense that it has solved its ultimate decision problem.

We can consider SUDO-PLANNER a grand-world planner in this sense because its universal plan class (Section 3.1) does not limit the scope of plans. In domain modeling, however, SUDO-PLANNER is just as dependent on small worlds as traditional DA systems. As Figure 6.13 illustrates, the models SUDO-PLANNER constructs to evaluate the effects of actions are small worlds. Specifically, they represent consequences at a level of abstraction coarser than the ultra-fine grand-world ideal. It appears to me that this relation will inevitably be approximate, and thus that small-world decision models are unavoidable.

The tenuous relationship between decision models and the decision-theoretic ideal is transparent when we view our models as imperfect representations of worlds that are small to begin with. Those recognizing the problem have drawn differing conclusions about its implications for decision-theoretic applications, in both DA and AI. In the next section, I examine some of these arguments and present the view behind the design of SUDO-PLANNER and its mechanisms for constructing decision models.

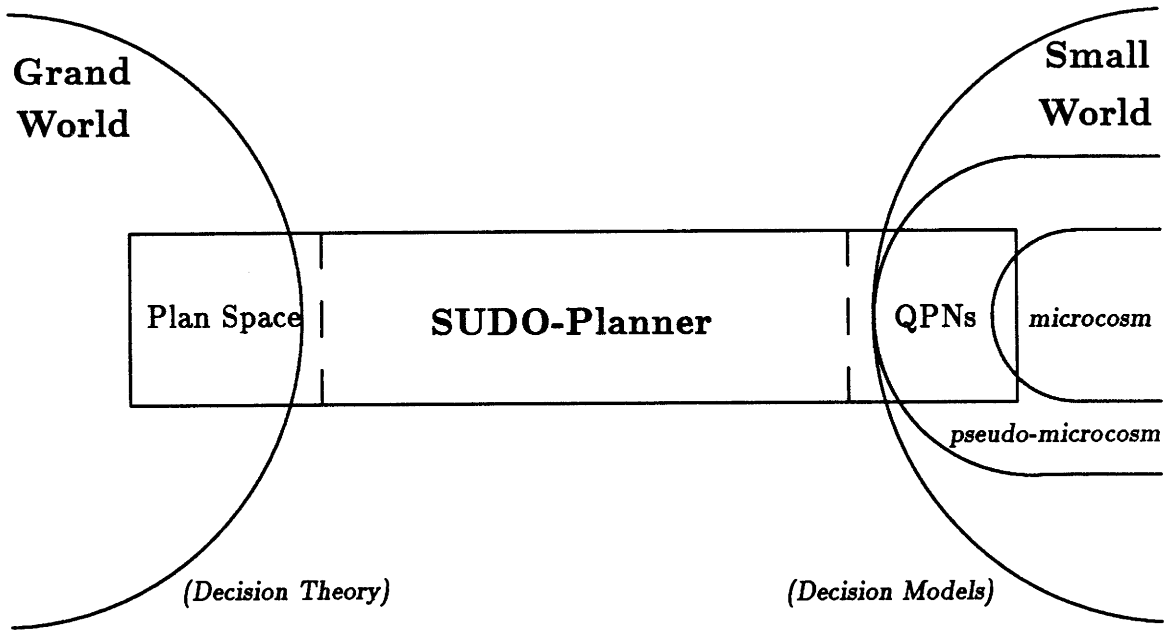


Figure 6.13: SUDO-PLANNER plans in the grand world using small-world decision models. The QPNs it generates will generally be pseudo-microcosms, faithful to the grand world only if they are also microcosms.

6.6.4 Constructive Decision Theory

Shafer has argued persuasively that the value of a theory of probability judgment depends on its *constructive* utility, that is, on factors that determine its usefulness for developing mental arguments, evaluating evidence, and expressing and explaining beliefs [127, 129]. Shafer's objections to Bayesian decision theory stem in part from its treatment of preferences and beliefs as conceptually innate and its consequent emphasis on elicitation rather than synthesis. Moreover, in his view [128, page 485], "the problem of small worlds serves as a demonstration of how far [Savage's] normative approach was from a sensible, constructive approach to decision."

My conclusion in Section 1.5 that decision models are not suitable knowledge bases is an endorsement for the constructive point of view. Once we undertake to design agents without explicit *a priori* decision models, it is clear that constructive issues will be influential. The lack of an established set of principles for decision model construction, a *constructive decision theory*, increases the difficulty of this endeavor.

Although I agree with Shafer about the goals of constructive decision theory, we diverge sharply in approach. Where Shafer has emphasized comparison of alternative probability calculi (in particular, Bayesian probability versus Dempster-Shafer belief functions [126]), I would focus on more comprehensive representation issues and procedures for constructing decision models from KBs. Furthermore, there is no need to cast this effort as a new foundation for normative behavior; the rational agent of decision theory remains a useful idealization for anchoring our constructive concepts [161].

Perhaps the most appealing path to a constructive decision theory is to broaden our rationality concepts to incorporate constructive factors. This is essentially the approach of Good, who introduces "type II" rationality as the extension of Bayesian rationality where the cost of computation is taken into account [41]. This principle is difficult to apply, however, as we are usually unable to precisely characterize the necessary computation. The full analysis requires consideration of an agent's decisions about its own cognitive make-up, which can lead to infinite regress if we are not careful. (See Doyle for a discussion of some issues in what he calls *rational self-government* [25].)

6.6.5 Constructed Models: Conclusion

Once we take into account the dynamic process of synthesizing decision models, the overall procedure is no longer guaranteed to be normative. Nevertheless, the relation to Bayesian decision theory is valuable as a comparative standard. Principles of model construction can be evaluated on this basis, to the extent that we can measure the potential distortion of alternative construction strategies.

Little research to date has addressed constructive decision-modeling issues, especially from a computational perspective. In this chapter, I have identified some basic problems in representing and reasoning with the knowledge SUDO-PLANNER uses for constructing QPNs. The description of SUDO-PLANNER's mechanisms represents a first step toward solving these problems and developing a set of principles for automated decision model synthesis.

Chapter 7

Dominance Proving

This chapter completes the description of SUDO-PLANNER by describing its dominance prover (the right-hand half of Figure 6.5). This module (1) derives and (2) records dominance conditions entailed by a given QPN. The first two sections of this chapter describe these tasks. SUDO-PLANNER's dominance prover supports planning up to tradeoffs, but its qualitative techniques are insufficient for resolving the tradeoffs identified. The remainder of the chapter considers approaches toward surmounting the tradeoff barrier in ways compatible with SUDO-PLANNER's dominance-proving architecture and its existing qualitative methods.

7.1 Reducing the QPN

The relation of QPN variables to the value node determines their decision-theoretic properties. The dominance prover derives these properties by transforming the original QPN to one where these relationships are direct. The reduction operator of Chapter 4 renders indirect relations direct by removing intermediate variables from the QPN and updating the remaining qualitative relations according to Theorems 4.11 and 4.21. Reduction is the sole QPN inference rule applied by SUDO-PLANNER's dominance prover.

The reduction process proceeds backwards from the value node. At each iteration, SUDO-PLANNER chooses a predecessor w of *value* such that $pred(w) \neq \emptyset$. It then constructs $red(w, G)$ by splicing w from the graph and recomputing the qualitative relations for each combination of predecessors and successors of w using formulas (4.9) and (4.18).[1] After each reduction, the dominance prover inspects the QPN for decision-theoretic implications and records the results as described below. The procedure terminates when all remaining variables have *value* as sole successor.

For example, at one point the model construction process produces the QPN of Figure 7.1a. The candidates for reduction are *long-term morbidity* and *mortality*. SUDO-PLANNER chooses one of these and reduces it, adding its predecessors to the potential reduction candidates. The program proceeds to reduce nodes one-by-one until it reaches the configuration of Figure 7.1b.

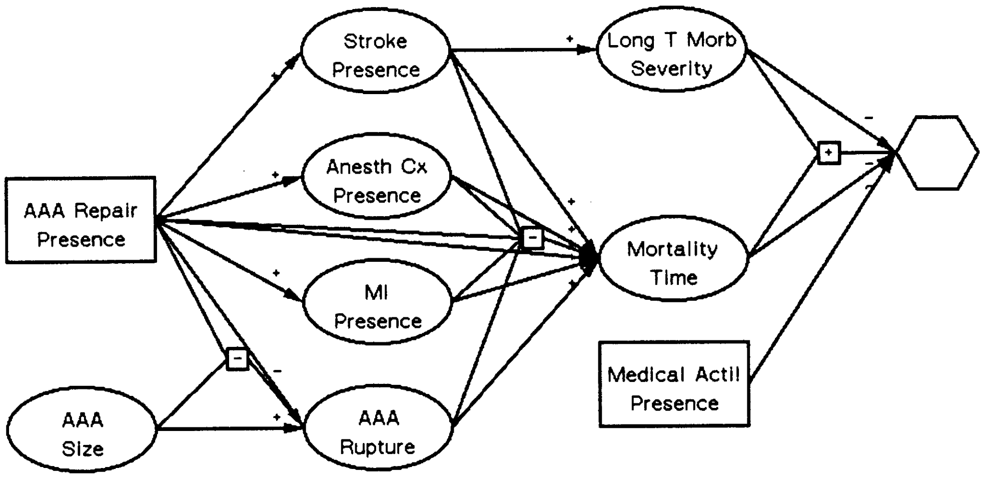
As shown in Section 4.4.3, reducing an arbitrary subset of the variables in a QPN can be performed with $\Theta(|V|^2)$ applications of the rule for updating qualitative influences (4.9), where $|V|$ is the number of variables in the network. Each influence update may require up to $O(|V|)$ applications of (4.18) to compute the new synergies (one for each potentially synergistic partner), bringing the overall worst-case complexity to $O(|V|^3)$.

7.2 Recording Dominance Results

Between reductions, SUDO-PLANNER inspects the QPN for relations implying dominance among plan classes. There are two ways that a QPN can manifest such implications, both involving qualitative relations with the value node. First, an action variable av may be directly related to *value*, $U^{\delta}(av)$ for $\delta \in \{+, -\}$. If av has no other

[1]The implementation of these reductions is non-destructive; the original QPN is required for modification in the model construction cycle of Chapter 6.

(a)



(b)

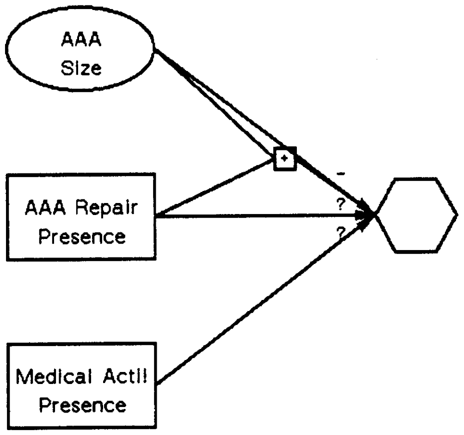


Figure 7.1: (a) The QPN before reduction. (b) The completely reduced QPN.

paths to *value*, this implies that the optimal value of av in the plan is its maximum or minimum value, depending on δ (see Section 4.5.2). Second, av may be synergistic on *value* with a potentially observable event variable ev, $Y_U^\delta(\{av, ev\})$. In this case, the optimal av policy is monotone in ev if ev is observable, by the monotone decision property (Section 4.7.3).

Let $av = r_j\langle a_i\rangle$, the action variable describing role r_j of action concept a_i. The first case can be expressed as a dominance condition by constraining plans with actions of type a_i to those with its subtype where r_j is restricted to the extreme value of its range. Let r_j^δ denote this extreme value, r_j^+ for the maximum and r_j^- the minimum. For example, **presence**$^+$ = **true** and **dosage**$^-$ = 0. The subtype of a_i, written $a_i(r_j^\delta)$, is obtained by value restricting r_j to r_j^δ. SUDO-PLANNER creates this concept in NIKL, which classifies it automatically in the action taxonomy.

To express the dominance result $D(\Pi_2, \Pi_1)$, SUDO-PLANNER must create the two plan classes. The dominance prover defines Π_1 by posting an action constraint (see Section 3.4.1) of type a_i on the universal plan class Ω. Π_2 is similarly defined by an action constraint of type $a_i(r_j^\delta)$. The plan space manager classifies both classes in the plan graph. The $U^\delta(r_j\langle a_i\rangle)$ condition implies that any plan in Π_1 can be improved (in a non-strict sense) by revising $r_j\langle a_i\rangle$ toward its extremum, therefore Π_2 dominates Π_1 by Definition 2.1. $D(\Pi_2, \Pi_1)$ is a restriction because $\Pi_2 \subseteq \Pi_1$.

The second type of dominance result is the monotone policy constraint justified by qualitative synergy. For example, in the reduced QPN of Figure 7.1b, *AAA repair* is positively synergistic on *value* with *AAA size*. (The $Y^?$ synergies holding between other pairs are implicit in the QPN graph.) $Y_U^+(\{AAA\ repair, AAA\ size\})$ implies that the optimal *AAA repair* policy is increasing in *AAA size*, which is asserted to be observable in the description of this particular case.

The method for recording this fact on the plan graph is analogous to the action constraint procedure described above. Let Π_1 be the class of plans where **AAA-**

repair is under consideration, defined by posting an action constraint of type **AAA-repair** on Ω. SUDO-PLANNER creates Π_2 from Π_1 by posting an additional monotone policy constraint of the form[2]

$$(+, \textbf{presence}\langle\textbf{AAA-repair}\rangle, \textbf{size}\langle\textbf{AAA-event}\rangle).$$

The dominance condition $D(\Pi_2, \Pi_1)$ asserts that plans where **AAA-repair** is under consideration can be restricted to threshold policies on **size⟨AAA-event⟩**.

SUDO-PLANNER must also ensure that the event variable in a monotone policy constraint is observable in the plan class restricted by the dominance result. An event variable is *freely observable* (observable without an explicit test action) if it is an action variable or, like *AAA size* in the example above, it is asserted to be so in a particular case. If the event variable *ev* in the $Y_U^\delta(\{av, ev\})$ condition is not freely observable, SUDO-PLANNER retrieves from its KB the set of action types asserted to render *ev* observable: $\{a_i \mid CO(a_i, ev)\}$ (see Section 5.7). As above, let Π_1 be the plan class where *av*'s action type is under consideration. For each action a_i in the set of observable creators, SUDO-PLANNER generates the plan class $\Pi_{1,i}$ by posting an action constraint of type a_i on Π_1. The dominance prover adds conditions of the form $D(\Pi_{2,i}, \Pi_{1,i})$ to the plan graph, where $\Pi_{2,i}$ is obtained from $\Pi_{1,i}$ by posting the monotone policy constraint (δ, av, ev).

For example, at a later point in the model construction cycle, the dominance prover obtains the result $Y_U^-(\{AAA\ repair, cath\ result\})$. (The conclusion follows from *AAA repair*'s negative synergy with *CAD* and the positive relation between *CAD* and *cath result.*) As shown in Figure 7.2, SUDO-PLANNER must ensure that the monotone policy constraint is only in force when *cath result* is observable, that is, when cardiac catheterization has in fact been performed.

[2]The notation *role*$\langle a_i \rangle$ is used for KB event variables to distinguish them from their corresponding QPN variables. For instance, **presence⟨AAA-repair⟩** corresponds to the QPN variable *AAA repair.*

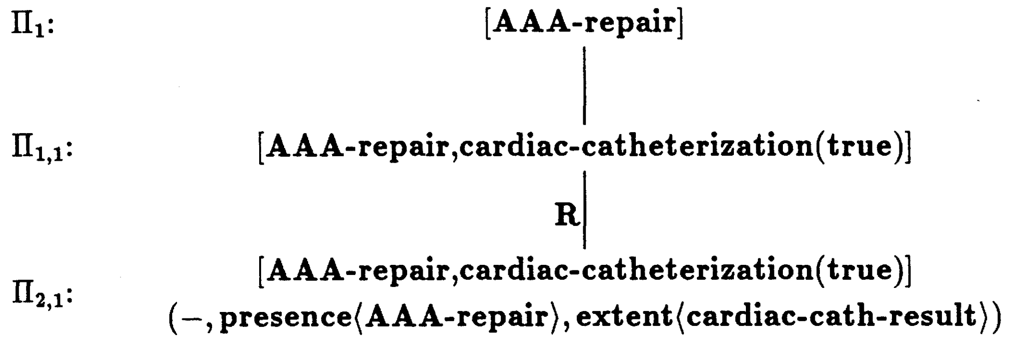

Π_1:
[AAA-repair]
$\Pi_{1,1}$:
[AAA-repair,cardiac-catheterization(true)]
R
$\Pi_{2,1}$:
[AAA-repair,cardiac-catheterization(true)]
(−, presence⟨AAA-repair⟩, extent⟨cardiac-cath-result⟩)

Figure 7.2: A plan graph fragment created to record a dominance result. The *AAA repair* policy is decreasing in *cath result* whenever the latter is made observable by catheterization.

The dominance prover performs one additional task during inspection of reduced QPNs. Any variables having known synergies with focus variables are marked as such. As described in Section 6.3.2, this is one of the criteria for directing the course of backward chaining in the model construction process.

7.3 The Tradeoff Barrier

The dominance-proving methods described above, in concert with the model construction procedure of the previous chapter, comprise SUDO-PLANNER's algorithm for planning up to tradeoffs. Although QPNs are powerful enough to justify an important class of commonsense decisions, the inherent weakness of the $\oplus$ operator prevents SUDO-PLANNER from reaching conclusions when the contributing factors conflict, the hallmark of a tradeoff situation.

Tradeoffs are dominance-proving dead ends for SUDO-PLANNER. Qualitative relations of sign "?" tend to proliferate, as a single unknown link renders ambiguous any composite pathway that spans it ($? \otimes \delta = ? \oplus \delta = ?$, for any $\delta \neq 0$). If a decision problem involves a tradeoff, so do any decisions for which it is a subproblem.

7.3.1 Example: A Subtle Tradeoff

The seriousness of the tradeoff barrier is illustrated by a subtle tradeoff encountered by SUDO-PLANNER in the running example. At one point in the model construction process, the dominance prover is presented with the QPN of Figure 7.3.[3] SUDO-PLANNER's task is to derive the synergy between *CAD* and *AAA repair* on *value*.

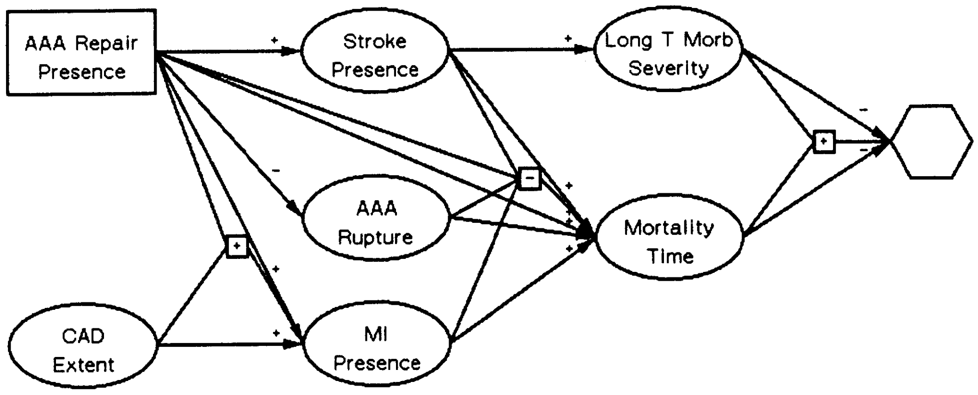


Figure 7.3: A subtle tradeoff. Although *AAA repair* and *CAD* interact in their influence on *MI*, SUDO-PLANNER's dominance prover cannot derive a synergy on *value.*

As shown in the figure, *CAD* and *AAA repair* have a direct positive synergy on *MI*. This means that increasing *CAD* increases the risk of *MI* due to the AAA surgery. While *CAD* increases the risk of *MI* in any event—a fact represented separately by the $S^+(CAD, MI)$ link—the expected increase is greater in the case of aneurysm surgery. Because *MI* is undesirable, the Y^+ relation to *MI* is an argument for a Y^- relation on *value*, and therefore for avoiding *AAA repair* in the presence of higher values of coronary disease. This is the justification given informally in Section 1.2 for the conclusion that *AAA repair* should be a threshold policy on *CAD.*

This factor is highlighted by the partly reduced QPN of Figure 7.4. If we could

[3]The QPN depicted here is simplified for expository purposes. Figure 7.3 displays the essential variables and qualitative relations, suppressing those that do not affect the analysis.

ignore the direct influence of *AAA repair* on *value*, then $Y_U^-(\{AAA\ repair, CAD\})$ would hold by simple propagation of synergies through influences (Theorem 4.20). Of course, we are not permitted to recklessly ignore this direct relation. Other effects of *AAA repair* may interact with *MI* and are therefore relevant to repair policy given *CAD*.

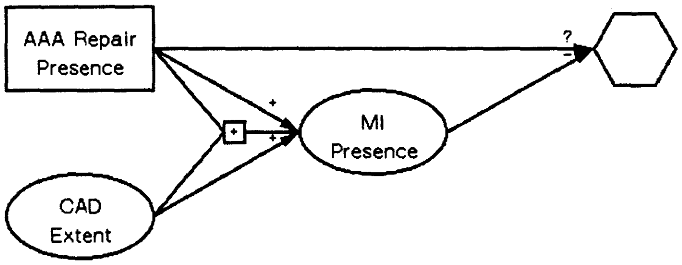


Figure 7.4: Partly reduced version of the QPN above. SUDO-PLANNER cannot derive the synergy between *AAA repair* and *CAD* on *value* because of *AAA repair*'s unresolved synergy with *MI*.

Three basic factors contribute to the synergy relation between *CAD* and *AAA repair*:

1. The positive interaction between the two variables on *MI*. As argued above, the increase in surgical risk presented by *CAD* supports a conclusion of Y_U^-.

2. *AAA rupture* is negatively synergistic with *MI* on *mortality*. Intuitively, this relation holds because a decrease in the rupture rate has a smaller impact on survival when the patient's life expectancy is already reduced by *MI*. In fact, all of the influences on *mortality* in this example are sub-synergistic, for the same reason that the noisy OR model implies Y^- (Section 4.7.2). In this case, the negative synergy combined with *CAD*'s positive influence on *MI* implies that the benefit of *AAA repair* in reducing ruptures is less important as *CAD* increases. Thus, this factor also supports Y_U^-.

3. On the flip side, the other influencers of *mortality* (such as *stroke*) also have less impact when *CAD* and therefore *MI* are increased. In this respect, a greater extent of *CAD* tends to decrease the mortality risk attributable to *AAA repair*, thereby supporting the conclusion Y_U^+.

The disagreement in sign between the second and third factors above is the reason that $Y_U^?(\{AAA\ repair, MI\})$ holds in the QPN of Figure 7.4 ($- \oplus + = ?$). Combining this indeterminate synergy with the other relations in the figure leads to the conclusion $Y_U^?(\{AAA\ repair, CAD\})$ in the completely reduced QPN.

We cannot resolve these factors qualitatively because they represent a genuine tradeoff. The argument that the *AAA repair* policy should be increasing in *CAD* because worse coronary disease decreases the importance of surgical risks is not spurious in the least: there are many examples in medicine and elsewhere where "having less to lose" is a legitimate basis for taking riskier action. Nor is this effect necessarily subordinate to the direct interaction. If we were considering a cause of mortality that interacted only slightly with *AAA repair*, this factor would hold sway.

7.3.2 Beyond the Tradeoff Barrier

This tradeoff indeed represents a dead end, preventing SUDO-PLANNER from drawing conclusions about any events or actions affecting *CAD*: observed *CAD history*, *cath result*, and *CABG* in this example. We know in this case that the first factor easily outweighs the other two, but this fact is not expressible in the QPN of Figure 7.3. This situation is particularly frustrating because given $Y_U^-(\{AAA\ repair, CAD\})$, SUDO-PLANNER could go on to derive useful dominance results involving these other variables.

In the next two sections, I consider the possibility of transcending the tradeoff barrier. Section 7.4 discusses methods for incorporating externally resolved tradeoffs into the tradeoff formulation process. A variety of approaches for generating these tradeoff resolutions are investigated in Section 7.5.

7.4 Externally Resolved Tradeoffs

Though the process of tradeoff resolution may depend on precise, absolute knowledge, the end result is essentially qualitative. The product of a tradeoff resolver is simply the sign δ (with $\delta = ?$ signifying failure) of the originally indeterminate relation. Given the resolution δ, SUDO-PLANNER can proceed from the dead end as if δ had been derived with its own qualitative methods.

7.4.1 A Black-Box Interface

Because a tradeoff formulator like SUDO-PLANNER can apply resolutions without knowing their pedigree, QPN-based inference complements other dominance-proving mechanisms. For example, if an external source determines that factor one (from Section 7.3.1) prevails over factor three, SUDO-PLANNER can conclude that *AAA repair* and *CAD* are negatively synergistic on *value*, and can go on to produce dominance results depending on that fact.

As Figure 7.5 illustrates, SUDO-PLANNER regards an external tradeoff resolver as a black box. When the attempt to add qualitative values results in ambiguity, SUDO-PLANNER presents the situation to the resolver. If resolution is successful, the resolver returns a non-"?" δ and SUDO-PLANNER proceeds from there.

7.4.2 Tradeoff Stubs

I have implemented a simple stub tradeoff resolver to demonstrate SUDO-PLANNER's ability to recover from dead ends. All qualitative relations in SUDO-PLANNER's KB

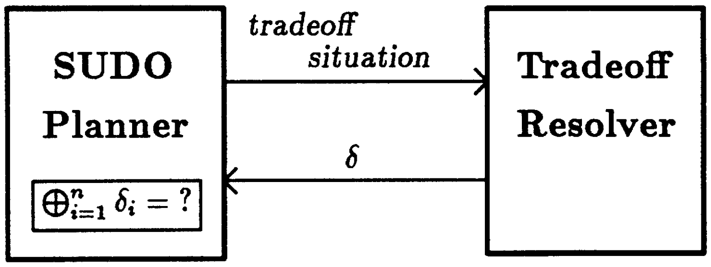


Figure 7.5: Black-box interface with an external tradeoff resolver.

are tagged with an identifier to facilitate reference to tradeoff situations. When sign addition is indeterminate, the stub resolver consults its list of answers to see if the present situation was anticipated. A total of five stub resolutions are required for the running example, including the conclusion that Y_U^- holds in the tradeoff presented in Section 7.3.1.

7.4.3 Resolver Candidates

Virtually any program that solves decision problems involving partially satisfiable goals can play the role of tradeoff resolver in the diagram of Figure 7.5. The ideal candidates are those that exploit knowledge difficult to express in SUDO-PLANNER's representation scheme yet can report their results in its qualitative terms.

Forms of knowledge suitable for resolving medical tradeoffs range from physiological models, like that in Long's program for heart failure [84], to representations of clinical trial studies, as proposed by Rennels [111]. The nature of tradeoff resolution knowledge in other domains is similarly unconstrained. Therapy predictions produced from the heart failure program [83] take the general form of qualitative influences, though the underlying model uses numeric relations.

The resolver's knowledge representation may itself be qualitative, though to complement SUDO-PLANNER it must make different qualitative distinctions, employ separate knowledge sources, or apply more powerful inference techniques. The feasibility of generating qualitative relations from other types of qualitative models is supported by the existence of programs performing similar tasks. For example, Downing's qualitative sensitivity analysis [22], and Weld's comparative analysis [155] both derive relations among variables from models describing *absolute* qualitative behavior.

7.5 Qualitative Tradeoff Resolution

The phrase "qualitative tradeoff resolution" is oxymoronic in the sense that qualitative unresolvability is my proposed definition for a tradeoff. But the meaning of "qualitative" is always relative to a set of qualities, hence varying the quality basis leads to different standards of what is qualitative. The implication for SUDO-PLANNER's task is that tradeoffs from one perspective may be non-tradeoffs from another. A tradeoff formulator should have the ability to select perspectives and reconcile the consequences of alternate viewpoints.

The following sections are speculative discussions of extensions to SUDO-PLANNER that could provide a more powerful tradeoff resolution capacity. They work by manipulating the notion of "qualitative" in one of the following ways:

1. Changing perspectives. Find another perspective or level of description in which the tradeoff is qualitatively resolved.

2. Introducing new qualitative distinctions. Other qualitative concepts can provide additional dominance-proving opportunities.

3. Modulating the degree of precision.

7.5.1 Spanning Influences and Synergies

A *spanning relation* is a domain relation that summarizes a body of network structure by directly connecting its endpoints. In a multilevel representation, the spanning relation coexists with the structure it summarizes at the same abstraction level. To interpret spanning relations, a reasoner needs the ability to maintain multiple exclusive views of a single relationship, perhaps using techniques similar to those employed by ALTERID [9] or CADUCEUS [107], mentioned in Section 6.5.

Formally, a spanning influence from a to b of sign δ asserts $S^{\delta}(a, b, G)$, where G is the network where all variables on pathways between a and b have been reduced. Spanning synergies are defined analogously. A spanning link mechanism is one that can reason about such relationships in the same model that contains the intermediate variables. A generalized spanning link mechanism would permit the G argument to vary arbitrarily across simultaneously considered qualitative relations. The answer to a query is the strongest result obtainable through transformation operators applied to any G.

Spanning links provide a means to express in the KB information currently encoded in tradeoff stubs. For example, if we know that the positive influence paths from a to b outweigh the negative ones, we can add $S^{+}(a, b)$ as a spanning link to express this fact. Even if $dir(a, b) = ?$ in the most detailed G, the spanning link can provide the stronger answer directly.

The implementation of spanning links, particularly the necessary adaptation of SUDO-PLANNER's model construction algorithm, presents some interesting problems. Modifying the assertion language to encompass spanning links is itself a significant extension, since the current representation never explicitly refers to G. It appears that a successful treatment of spanning links will require a clarification of the semantical issues raised in Section 5.4.1.

7.5.2 Negligibility Reasoning

The "order of magnitude" techniques [17, 110, 154] recently investigated in qualitative physics present another possibility for tradeoff resolution. In the case when one parallel influence can be declared *negligible* with respect to another—for example, the mildly unpleasant taste of an orally-administered drug relative to its curative powers—indeterminacy can be avoided by simply ignoring the former when in conflict with the latter.

Asserting that a factor is negligible means quite literally that it can be neglected without loss of validity. Researchers in qualitative physics have formalized this notion in terms of nonstandard analysis [18], an axiomatic framework for infinitesimal quantities. In the planning task, the standard for negligibility is in a factor's effect on the ultimate decision. Further work is required to formalize this standard in terms of infinitesimal quantities, and to relate these quantities to the probabilities and utilities that define QPN relations.

7.5.3 Incorporating More Precise Information

Another broad tradeoff resolution strategy is to introduce more precise forms of information into the dominance-proving process. Extending a model to include partial or complete descriptions of the magnitudes of probabilities and utilities enlarges the set of decidable comparisons.

At the heart of any tradeoff is an unresolvable inequality on expected utilities. Though deciding this inequality based on an arbitrary set of constraints is intractable, a considerable body of research on AI and decision theory addresses restricted versions of this problem. For example, Sacks describes an algorithm for inequality proving based on algebraic constraints [119]. Other work has focused on special properties of classes of probabilities and utility functions (for a sampling, see [31, 50, 156, 162]).

Researchers in qualitative reasoning have also begun to consider mechanisms that employ precise knowledge to resolve ambiguities inherent in the original qualitative representations [74, 164]. Work along these lines should also be applicable to qualitative probabilistic reasoning.

Chapter 8

The Complete Example

Our running example, the AAA/CAD/CVD case, has been presented piecemeal throughout this report. In this chapter I assemble the fragments into a more cohesive, chronological description of SUDO-PLANNER's performance on this problem.

8.1 Input Specification

The case, introduced in Section 1.2, is simply stated: a patient with a known history of CAD and CVD presents with a large AAA. From a medical perspective, of course, this description is exceedingly sketchy. Nevertheless, the description is sufficient to specify the relevant tradeoff formulation task, and any further information could not be exploited by SUDO-PLANNER.

The encoding of the case for input to SUDO-PLANNER is also quite simple. The status quo optimality heuristic of Section 6.4 dictates that the planner focus on changes to the current state. In this case, the underlying assumption is that the current therapy appropriately takes into account the patient's CAD and CVD, therefore planning should focus on strategy modifications warranted by the new finding, AAA.[1] There-

[1]Relaxing this assumption merely entails consideration of a broader set of findings. In this example, we could reexamine the current strategy by treating CAD and CVD as if they had just been discovered, like the AAA. The result is a blunter focus.

fore, the input to SUDO-PLANNER specifies that **size⟨AAA-event⟩** be represented in the initial QPN, and that it be a focus variable.

The case description also specifies that the variable **size⟨AAA-event⟩**, as well as **history⟨CAD-event⟩** and **history⟨CVD-event⟩**, is freely observable. The initial findings of the case are eligible to appear in conditional plans without additional test actions to reveal their values.

That is the complete SUDO-PLANNER input for the running example. Realistic medical cases would specify many more changes and observations.

8.2 The Evolving QPN

Snapshots of the QPN in various stages of development appear throughout the previous two chapters. Figure 8.1 recapitulates the first few stages of QPN evolution, described in Section 6.3. In the initial QPN (Figure 8.1a), *AAA size* negatively influences *value* by virtue of being a *disease severity*. This relationship is elaborated in Figure 8.1b, revealing that the undesirability of the aneurysm is due to its potential for rupture. Further elaboration (not shown) specializes *aneurysm rupture* to *AAA rupture*. Backward chaining on this variable introduces *AAA repair* to the QPN (Figure 8.1c), presenting the first opportunity to reason about possible action.

At this point, however, SUDO-PLANNER cannot determine anything about the *AAA repair* policy because the synergy between *AAA repair* and *AAA rupture* on *value* is indeterminate. A sequence of further elaboration and backward chaining steps replaces the direct relation from *AAA repair* to *value* with a collection of paths through such intermediate variables as *stroke*, *MI*, and *mortality*, which together account for the negative effects of the vessel surgery. Most of the new structure arises from elaboration; *mortality* and *value* are the only variables extended back in the process. The resulting QPN is illustrated in Figure 7.1a of the previous chapter. As

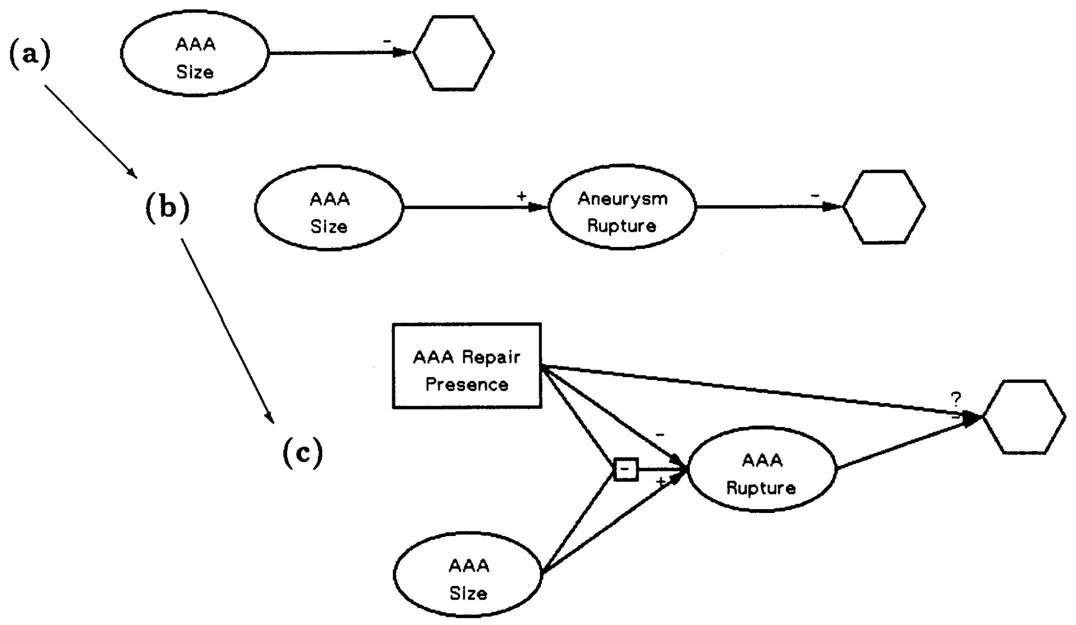


Figure 8.1: Early evolution of the QPN.

demonstrated by the reduction of Figure 7.1b, this model is sufficient to justify our first dominance result: the *AAA repair* policy should be monotonically increasing in *AAA size.*

Backward chaining on *MI* introduces *CAD* to the network, producing the QPN partially depicted in Figure 7.3. As described in Section 7.3.1, SUDO-PLANNER cannot resolve the synergy between *AAA repair* and *CAD* on *value*, due to a subtle tradeoff involving the effects of *CAD*. This conflict is resolved by one of SUDO-PLANNER's tradeoff stubs (Section 7.4.2), allowing the model construction process to continue. A similar tradeoff involving *CVD* (introduced by backward chaining on *stroke*) also requires a stub for resolution.

Resolving these synergies renders the two disease variables extensible. *CVD* is the next variable selected for backward chaining. Because *endarterectomy* is related to *CVD* by a Markov influence, the variable undergoes a mitosis process as described in Section 6.3.3. The resulting QPN is illustrated in Figure 8.2.

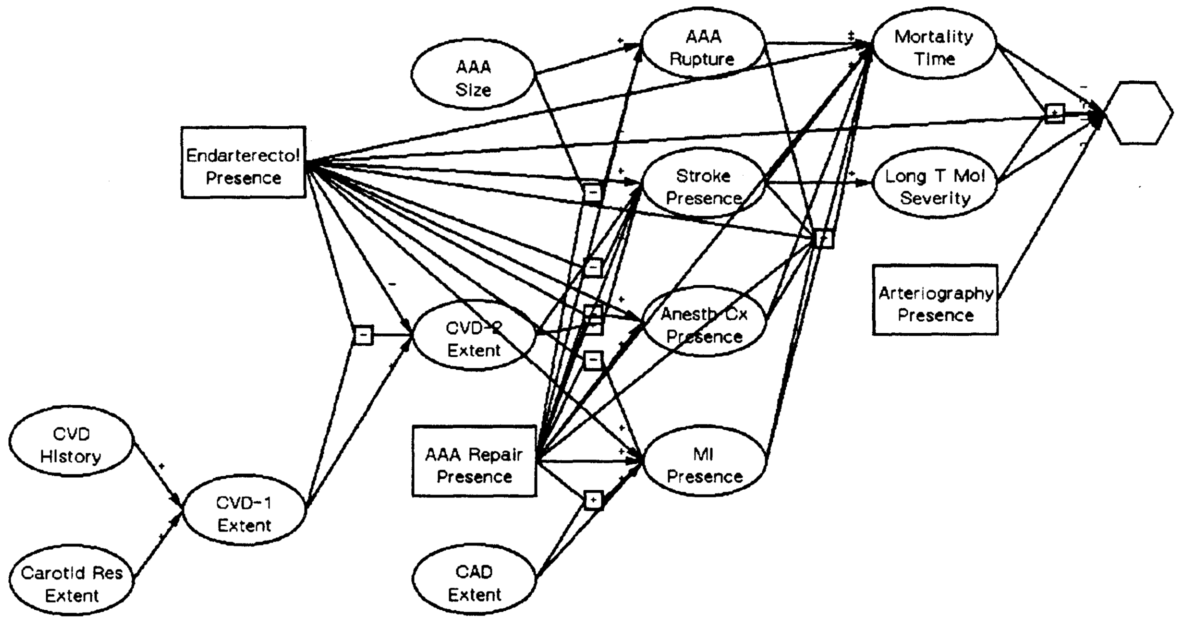


Figure 8.2: The QPN after backward chaining on *CVD*. The variable is divided in response to the Markov influence from *endarterectomy*.

By backward propagation of synergies (Corollary 4.22), SUDO-PLANNER's dominance prover can establish at this point that *CVD*'s predecessors, *CVD history* and *carotid arteriography result*, are negatively synergistic with *AAA repair* on *value*. The arteriography result is observable only if the test is performed; SUDO-PLANNER adds the action variable to the QPN as shown in Figure 8.2. Figure 8.4 (next section) illustrates the plan graph at this stage of the model-construction/dominance-proving process.

Backward chaining on *CAD* leads it to undergo mitosis and introduces structure parallel to that for *CVD* (see Figure 6.11). After elaboration removes a few spurious high-level links, the KB is exhausted and the model construction cycle terminates.[2] The final QPN is depicted in Figure 8.3.

8.3 Dominance Results

SUDO-PLANNER invokes the dominance prover on every QPN produced in the evolutionary sequence described above. It obtains the first dominance result from the QPN of Figure 7.1, and the next two after backward chaining on *CVD* to produce the QPN of Figure 8.2. The intermediate plan graph recording these three restrictions is shown in Figure 8.4. The restriction of Π_1 to Π_4 corresponds to the dominance result encoded by the plan graph fragment of Figure 7.2.

As the dominance prover derives further results, it creates and classifies the necessary plan classes and posts the dominance conditions on the plan graph. Figure 8.5 depicts the final plan graph, SUDO-PLANNER's terminating output for this example. The plan graph contains dominance results corresponding to each of the intuitive tradeoff formulation conclusions discussed in Section 1.2.

[2]If the KB were much larger, running the process exhaustively would be infeasible. Because plan graph validity is invariant, however, the planner can be halted at any time for inspection of its dominance results.

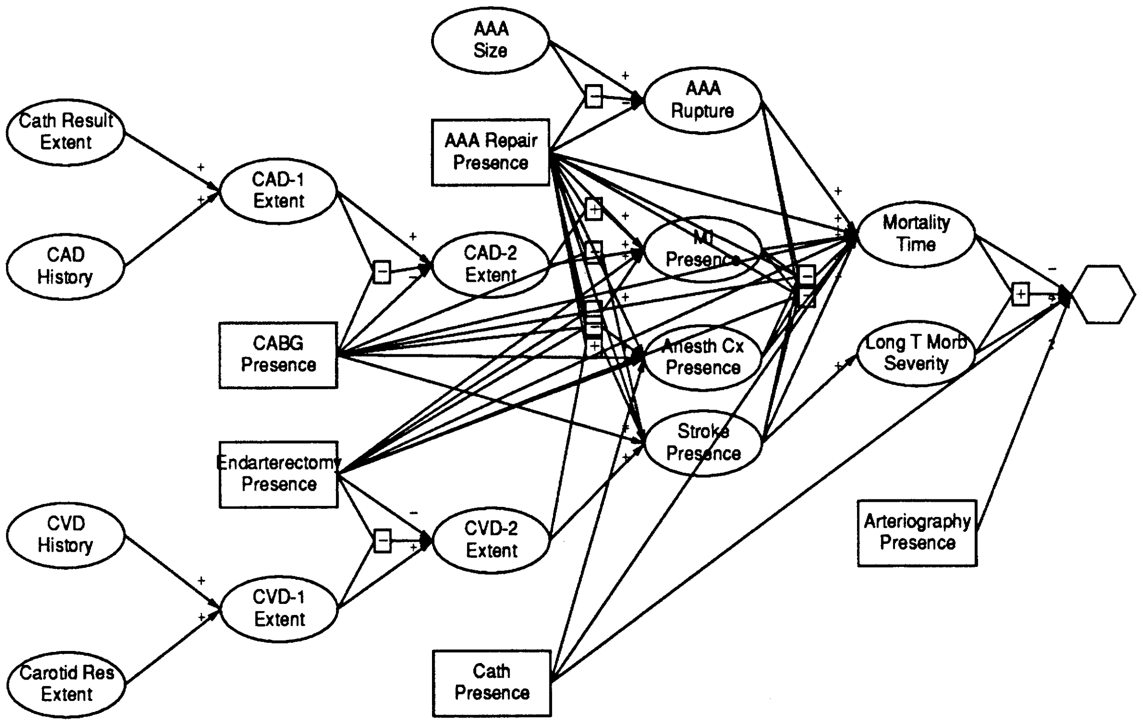
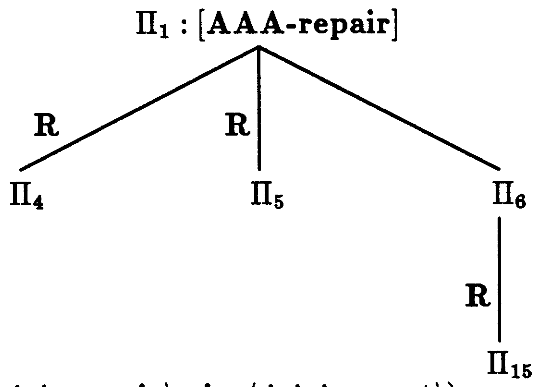


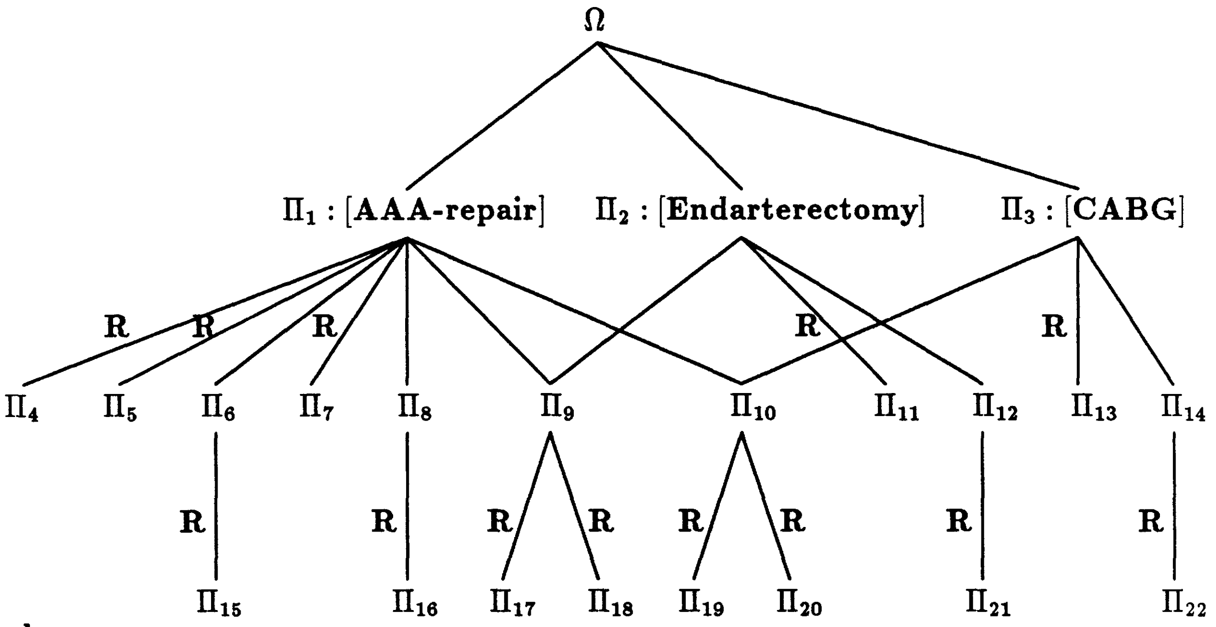
Figure 8.3: Final QPN for the AAA/CAD/CVD example (repeated from Figure 1.12).



key:

Π_4 : (+, **presence⟨AAA-repair⟩, size⟨AAA-event⟩**)
Π_5 : (−, **presence⟨AAA-repair⟩, history⟨CVD-event⟩**)
Π_6 : [**carotid-arteriography(true)**]
Π_{15}: (−, **presence⟨AAA-repair⟩, extent⟨carotid-arteriography-result⟩**)

Figure 8.4: The plan graph after reducing the QPN of Figure 8.2. Plan classes are described by the constraint distinguishing them from their parents.



key:

Π_7 : **(−, presence⟨AAA-repair⟩, history⟨CAD-event⟩)**
Π_8 : **[cardiac-catheterization(true)]**
Π_{11}: **(+, presence⟨endarterectomy⟩, history⟨CVD-event⟩)**
Π_{12}: **[carotid-arteriography(true)]**
Π_{13}: **(+, presence⟨CABG⟩, history⟨CAD-event⟩)**
Π_{14}: **[cardiac-catheterization(true)]**
Π_{16}: **(−, presence⟨AAA-repair⟩, extent⟨cardiac-cath-result⟩)**
Π_{17}: **(+, presence⟨AAA-repair⟩, presence⟨endarterectomy⟩)**
Π_{18}: **(+, presence⟨endarterectomy⟩, presence⟨AAA-repair⟩)**
Π_{19}: **(+, presence⟨AAA-repair⟩, presence⟨CABG⟩)**
Π_{20}: **(+, presence⟨CABG⟩, presence⟨AAA-repair⟩)**
Π_{21}: **(+, presence⟨endarterectomy⟩, extent⟨carotid-arteriography-result⟩)**
Π_{22}: **(+, presence⟨CABG⟩, extent⟨cardiac-cath-result⟩)**

Figure 8.5: The final plan graph. Plan classes Π_9 and Π_{10} are defined by the union of their parents' constraints. All leaf plan classes dominate (and therefore restrict) their parents.

8.4 Performance

The entire process takes roughly two minutes of real time on a Symbolics 3650.[3] The code is not optimized, and no extensive metering has been performed to determine the allocation of computation time. SUDO-PLANNER appears to spend most of its time in redundant or superfluous dominance proving that could be avoided without degrading performance.

8.5 Discussion

This example illustrates SUDO-PLANNER's tradeoff formulation performance on a realistic, albeit small, medical decision problem. The program successfully plans up to tradeoffs, failing to resolve only genuine qualitative conflicts. Some of these are quite subtle; the competing factors became apparent to me only upon analysis of the QPN reduction. Given resolutions for these, SUDO-PLANNER proceeds to derive all of the desired results.

Two aspects of the example are disappointing. First is its fragility: small modifications to the KB can produce seemingly chaotic change in the course of QPN evolution and sometimes lead to weakened dominance results or even invalid ones if incompatible KB relations are brought into the model. I ascribe this defect to the cognitive complexity of the model construction procedure and the difficulty of predicting the implications of closed-world assumptions. Second, SUDO-PLANNER fails to take maximal advantage of abstraction, producing dominance results later in the process than necessary. As mentioned in Section 6.3.4, this is a by-product of the conservative knowledge engineering policy of including indeterminate high-level relations to prevent improper assumptions of independence.

[3]This figure assumes that the KB is pre-loaded and that the graphical display of QPNs is disabled.

The AAA/CAD/CVD case is the only complete, working SUDO-PLANNER example. In the process of implementing the program I have constructed several other QPN examples, including the digitalis model of Chapter 4. The difficulty of knowledge engineering mentioned above precluded development of a more comprehensive KB supporting a variety of model construction examples. This is not simply the traditional "knowledge acquisition bottleneck"; rather, it reflects the unpredictability of SUDO-PLANNER's QPN generation procedure. Improving its robustness is a prerequisite for widening the scope of SUDO-PLANNER.

Above all, the example serves as a demonstration of plausibility for the basic components of the SUDO-PLANNER approach: dominance proving, QPNs, and decision model construction from a multilevel KB. Weaknesses in SUDO-PLANNER exposed by the example provide a starting point for efforts to improve tradeoff formulation technology.

Chapter 9

Conclusion

I conclude the dissertation with an assessment of what this research has achieved and some remarks about what remains to be accomplished.

9.1 Summary of Contributions

The two central elements of this work are the dominance-proving architecture of Chapter 2 and the QPN formalism of Chapter 4. I have implemented these ideas in SUDO-PLANNER, a program that formulates tradeoffs by constructing decision models from a multilevel KB of qualitative relations. The following sections summarize these contributions, and enumerate some other products of this project scattered throughout the thesis.

9.1.1 A Dominance-Proving Architecture for Planning with Partially Satisfiable Goals

In the classical framework, planners search for a course of action guaranteed to achieve a specified goal predicate. As I argue in Section 1.4, this formulation of the task is fundamentally inadequate to account for partial goal satisfiability, including the special case where the effects of actions are uncertain. Although numerous researchers have developed planners to handle some of its aspects, no comprehensive computational framework for the more general problem has yet emerged.

The dominance-proving architecture is an attempt to fill that void. As presented in Chapter 2, the architecture's criterion for choice among plans is highly general; in particular it admits a Bayesian approach to preferences and belief without imposing *ad hoc* restrictions on the form of utility functions and probabilities. It goes beyond pure decision theory, however, in addressing the assembly of plans from more primitive descriptions of action. The architecture prescribes a division of computational labor among three components: the plan space manager, dominance prover, and domain problem solver. The plan graph representation of the search space supports techniques from traditional planning, in particular hierarchical (constraint-posting) planning and dependency maintenance.

The generality of the dominance-proving architecture precludes a final evaluation based solely on the results of this project. SUDO-PLANNER is one instance of a dominance-proving planner, developed specifically for the tradeoff formulation task. Other instances, defined by alternate choices in the design of plan class representation, domain modeling language, and dominance prover, should be expected to exhibit a variety of planning behaviors.

9.1.2 QPNs: A Formalism for Qualitative Probabilistic Influences and Synergies

Qualitative probabilistic networks are representations for constraints on probabilistic relations among a set of variables. The formalism provides constructs for two types of constraint:

- Qualitative influences describe the direction of the relation between a pair of variables.

- Qualitative synergies describe the direction of interaction among influences.

Both qualitative relations have rigorous probabilistic definitions that justify sound inference procedures based on efficient network transformations. Because the semantics for qualitative influences and synergies are based on relative, rather than absolute, statements about conditional probabilities, QPNs can be used to derive properties of the relative values of alternative plans. In particular, the relation of variables to the special *value* variable determine qualitative properties of the optimal decision, which can be exploited directly by a dominance-proving planner. I summarize the formal properties of QPNs in Section 4.9.1.

Qualitative relations directly support tradeoff formulation. In fact, resolvability in QPNs is perhaps the best available criterion for a formal tradeoff definition. As argued in Section 1.1, knowledge for tradeoff formulation is more abstract, modular, and robust than that required for general decision making. QPNs provide a mechanism to exploit these properties as far as possible before resorting to the less convenient forms of knowledge needed to solve the residual planning problem.

9.1.3 Other Contributions

In developing the main ideas above and building the SUDO-PLANNER implementation, I have been led to address a variety of peripheral topics. Those figuring prominently in the thesis include:

- Identification of the tradeoff formulation task and its role in planning (Section 1.1).

- Design and analysis of knowledge representations for plan classes, actions, and the effects of actions (Chapters 3 and 5). Development of a subsumption algorithm for the plan constraint language (Section 3.5).
- Model construction at multiple levels of abstraction (Chapter 6).
  - Desiderata for multilevel reasoning (Section 6.1).
  - Exploration of issues in the automatic construction of decision models (Sections 6.4 through 6.6).
- Interpretation of the STRIPS assumption for actions with uncertain effects (Section 5.2.2).
- A theoretical framework for decision making in Savage's "grand world" (Section 6.6.3).

9.2 Limitations of SUDO-Planner

The design of SUDO-PLANNER reflects an emphasis on decision problems substantially different from those typically considered in AI planning research. As such, the domain relations expressible in its KB and the conclusions derivable by its inference procedure are novel to the field. Conversely, SUDO-PLANNER's mechanisms fail to cope with many issues that traditional systems handle routinely. Most of these omissions are not critical; straightforward extensions to SUDO-PLANNER would attain the desired capabilities. Others, however, do not have clear remedies.

In the sections below, I discuss two glaring limitations of SUDO-PLANNER's competence. Several additional shortcomings of this work have been reported in foregoing chapters:

- The limited expressive power of the plan constraint language (Section 3.5.4).

- An unsatisfactory semantics for effect assertions in the KB (Section 5.4.1).
- The fragility of the model construction algorithm (Section 6.3.4).

9.2.1 Tradeoffs

The tradeoff barrier is an obvious boundary of SUDO-PLANNER's decision-making abilities. Sections 7.3 through 7.5 discuss the nature of this obstacle and propose some approaches toward overcoming it.

The subtle tradeoff of Section 7.3.1 suggests that—though close inspection reveals that this situation contains genuinely conflicting factors—the QPN formalism does not correspond exactly to our intuitive notion of tradeoff. Further theoretical and empirical analysis is required to allocate blame for this mismatch to inferential weakness, omission of critical qualitative distinctions, and various cognitive elements. The first step is to develop a precise characterization of the completeness of QPN inference mechanisms, that is, the potential for spurious ambiguity.[1]

9.2.2 Time

No theory of planning is complete without an adequate treatment of temporal relations among actions and events. SUDO-PLANNER's representations for plan classes, actions, and events, however, are entirely atemporal. Failing to deal with time is one of the major shortcomings of SUDO-PLANNER.

Most planners handle simple sequencing among actions. For typical medical decision problems, such a mechanism is not nearly adequate and barely provides an advantage over no temporal reasoning at all. For example, in patients presenting with symptoms of appendicitis, there is a tradeoff between performing an appen-

[1]Kuipers has carried out an analogous analysis of qualitative simulation [73], concluding that spurious qualitative behaviors are in fact produced by local inference schemes.

dectomy immediately or waiting and observing the patient to better establish the diagnosis. Waiting avoids some unnecessary surgeries while increasing the risk of untreated disease. In this planning problem, the ordering of actions is already known; the task is to determine the optimal waiting time as a function of the observations.

To attack this problem with SUDO-PLANNER, one would have to treat its temporal aspect as just another domain-dependent characteristic of the situation. Because they enjoy no special status in the planner, references to time must be expressed within the existing representations for action and event variables. In this case, we can include a **time-of-surgery** role in the specification of the action **appendectomy**. The filler for that role is a time variable whose effects can be represented via the mechanisms described in Chapter 5.

SUDO-PLANNER does supply one construct specifically to account for time relations: the Markov influence assertion of Section 5.5. Markov influences implicitly encode the temporal ordering among an action and the value of an event variable before and after its application.

But even a large set of representation constructs like Markov influences combined with vigorous use of special time-related roles is no substitute for an explicit uniform treatment of time within the planner and in the representation for effects of actions. Failing to recognize that the **time-of-surgery** role of **appendectomy** is related to other temporal measures in the KB and plan graph inevitably leads to significant missed inference opportunities.

Temporal reasoning is likely to pose special problems in the dominance-proving architecture because of its emphasis on deriving properties of plan classes. In categorical planning, plans with partial temporal descriptions require the planner to establish that facts hold in all consistent completions, a problem known to be difficult [21]. It remains to be seen whether techniques from traditional planning can be adapted to this framework, or, for that matter, whether the dominance-proving task will admit some special solutions not available in the traditional case.

9.3 Further Work

The preceding chapters contain numerous digressions outlining plans and speculations about future work. This section elaborates on a few of these possibilities.

9.3.1 Extending the Dominance Prover

There are a few straightforward extensions to the SUDO-PLANNER implementation that should enlarge the dominance prover's scope without presenting conceptual difficulties. One would be to incorporate explicit reasoning about landmark values, as described in Section 4.7.5. For example, CAD_0 might be an important landmark value for CAD, representing no coronary disease or perhaps the normal extent of CAD for a given population. A medical KB would also be likely to include the value of z^*, the normal threshold value on *AAA size* above which *AAA repair* is indicated. The threshold z^* is a rule of thumb, valid as a decision criterion only for patients with no extenuating circumstances. Thus, use of z^* implicitly assumes $CAD = CAD_0$, among other things. Suppose a patient presents with CAD worse than CAD_0 and a AAA smaller than z^*. The patient's coronary disease is an "extenuating circumstance" for the *AAA repair* decision, but we know from SUDO-PLANNER's plan graph that extensive CAD only decreases our willingness to perform aneurysm surgery. Since the patient's AAA is below the nominal threshold, we can conclude that it is also below the revised threshold and therefore the optimal plan does not prescribe aneurysm repair.

The reasoning scenario presented above makes use of dominance results produced by the current version of SUDO-PLANNER. Hence, extending SUDO-PLANNER to handle landmark values would not require changes in either the QPN formalism or the model construction process.

A second extension of the dominance prover is the incorporation of explicit conditioning of qualitative relation assertions and dominance results. Maintenance of context-dependent conclusions (such as the conditional dominance relations of Definition 2.3) can be implemented directly in a standard TMS. SUDO-PLANNER's plan class representation already provides for conditional constraints (Section 3.4.3), although its subsumption algorithm for plan classes with conditionals is incomplete (Section 3.5.3).

Other extensions—involving new qualitative relations, plan class constraints, or tradeoff resolution capabilities—are more fundamental, requiring significant modifications of SUDO-PLANNER's existing mechanisms. I expect that further work on SUDO-PLANNER will explore each of these directions.

9.3.2 Temporal Representations

Designers of temporally competent dominance-proving planners need to address three basic representation requirements. A temporal knowledge representation must include facilities to express:

1. the temporal relation among actions and observations in plans,
2. effects of action over time, and
3. time preference for the occurrence of events, including the temporal resolution of uncertainty [72].

The first item involves extensions of the plan constraint language, while the second and third are the responsibility of the domain modeling language.

There is no reason to develop languages for expressing temporal relations among actions in a plan unless the planner has some basis for distinguishing the effects of the different temporal patterns of action. The first step in adding temporal concepts to SUDO-PLANNER, therefore, should be to extend the QPN formalism to include conditions on relations among variables over time. One approach is suggested by Cox's investigations of a temporal version of first-order stochastic dominance [16]. Because atemporal FSD is the basis for the existing QPN relations, such a condition is a promising candidate for a smooth extension into the temporal dimension.

A contrasting approach is to add temporal structure to the event variables rather than to the relations. This is the path taken in a representation scheme I designed for a proposed health outcome knowledge base [157]. The KB design is organized around a taxonomy of temporal patterns, orthogonal to the health-related characteristics of the concepts. Given a temporal semantics for the pattern-definition constructs (perhaps in the spirit of Allen [1]), relations on these variables have clear temporal interpretations. The same constructs could then be applied to the specification of temporal characteristics of actions as well.

9.3.3 Critiquing Based on the Dominance-Proving Architecture

Perhaps the most immediate application of the dominance-proving architecture should be to *critiquing systems* [91, 159], programs that analyze a user's proposed plan rather than generate solutions from scratch. A dominance-proving planner critiques by exploring the plan-space neighborhood of the given plan or plan class. Dominance results involving plan classes that encompass the user's proposal suggest incremental improvements to be recommended by the critiquer.

The user's strategy focuses the dominance-proving planner on a narrow slice of the search space that might have received only fractional attention in unrestricted search. Though the planner may not have gotten there in the first place, it is possible that once there it can derive an abundance of useful results.

A proposed plan often implicitly contains resolutions of tradeoffs that would have caused SUDO-PLANNER to hang. This might be attributable to more domain knowledge, more information about the case context, or greater willingness to make assumptions or guesses. Assumptions inferred from the proposal can be asserted in the planner's KB, in much the same way as results are incorporated from black-box tradeoff resolvers in the framework of Section 7.4. Further refinements or modifications to the plan justified by its implicit assumptions are valid dominance results that should be reported in the critique.

We have incorporated some of these ideas in BUNYAN, a program for critiquing medical decision trees [160]. One of BUNYAN's "critiquing principles" is to alert the analyst to decision models that contain qualitatively dominated strategies. Although the program does not perform explicit dominance proving, the criteria for detecting dominance are based on qualitative probabilistic relations holding between abstract concepts of medical decision problems.

9.3.4 Tradeoff Resolution

Section 7.5 outlines three approaches toward resolving tradeoffs in ways compatible with SUDO-PLANNER's existing dominance-proving mechanisms:

1. Spanning relations.

2. Negligibility reasoning.

3. Higher-precision knowledge.

Pursuit of each of these directions should be part of a full-scale assault on SUDO-PLANNER's tradeoff barrier.

9.4 Outlook on SUDO Planning

The central task that this research attacks—planning for partially satisfiable goals—is so pervasive, and the need for comprehensive AI approaches so acute, that partial solutions are well worth striving for. The same holds for other major issues addressed by SUDO-PLANNER: qualitative probability, multilevel knowledge representation, and decision model construction. It is a virtual certainty, then, that others will take up these problems, whether or not they adopt techniques from SUDO-PLANNER.

However, the outlook for SUDO-PLANNER: THE PROGRAM is quite dim. In this chapter and throughout the thesis I have chronicled numerous defects of SUDO-PLANNER that preclude it from direct application to realistic planning problems. SUDO-PLANNER is merely a demonstration vehicle, providing an instance of the dominance-proving architecture for planning with partially satisfiable goals, and a testbed for the application of qualitative probabilistic networks to problems of decision making under uncertainty. Future tradeoff formulators will surely operate much differently, as researchers discover and exploit regularities in problems and domains. And dominance-proving planners of the future will emphasize tradeoff resolution to the point that resemblance to SUDO-PLANNER will be slight at best.

In contrast, I find it easy to be optimistic about the prospects for SUDO-PLANNER: THE APPROACH. The full range of dominance-proving planners has not yet been explored, nor has the variety of possible uses for QPNs. The SUDO-PLANNER experience to date has been encouraging; however, much work remains before we can evaluate with confidence the utility of these ideas in establishing a principled basis for automated decision making.

Appendix A

Notation

Listed below are brief descriptions of notational symbols employed in the body of this thesis, with references to the pages where they are introduced or defined. For symbols with more than one interpretation, the appropriate choice should always be clear in context.

a^*	A landmark value of variable a, either specially designated as such or determined to correspond to landmark value of some other variable (p. 119).
a_i	1. An action in the action alphabet A (p. 43).
	2. A generic action type (p. 71).
	3. A specific value taken on by variable a: $a_i \in X(a)$. Used in conditional probability expressions as an abbreviation for the proposition $a = a_i$ (p. 96).
$a_i(vr)$	The action type formed by value-restricting some role of a_i to vr (p. 71).
$a_g(b)$	A function that returns the value of a maximizing $g(a, b)$ (p. 113).
A	1. An alphabet of actions for a simple plan language (p. 43).
	2. The proposition interpreted as $a = \mathtt{true}$, where a is a boolean variable in a QPN (p. 93).
A^*	The set of strings in the alphabet A, or a wildcard in a regular expression (p. 43).
$\bar{A}$	The proposition $a = \textbf{false}$ (p. 93).

$\mathcal{A}$	The universal action class. $\mathcal{A} = \mathcal{X}(\mathbf{action})$ (p. 71).
$\mathcal{A}_i$	A generic set of actions. $\mathcal{A}_i = \mathcal{X}(a_i)$ (p. 71).
action	The root of the action taxonomy (p. 71).
$\arg\max_a$	The value of a that maximizes the associated expression (p. 113).
av	An action variable type (p. 73).
AV	An action variable instance of type av (p. 74).
$AV(\pi)$	The value of action variable instance AV in the unconditional plan π (p. 74).
c	The consequence descriptor in decision analysis used as argument to the utility function (p. 179).
$\underline{c}$	The minimal value of variable c (p. 228).
$CO(a_i, ev)$	A creates-observable assertion. Performing an action of type a_i renders event variable ev observable (p. 141).
D	1. Dominance relation over plan classes (p. 46).
	2. The set of decision variables in a QPN (p. 89).
D'	Strict plan class dominance (p. 46).
D_S	Conditional dominance relation for condition S (p. 47).
$dF_b(b_0)$	The derivative of the cumulative distribution, appearing in integrals. Equivalent to $f_b(b_0)db$ (p. 95).
$dep(G)$	The dependency graph of a QPN G (p. 97).
$dir(a, b, G)$	The direction of influence of a on b in G (p. 97).
e_i	A generic event type (p. 79).
$e_i : \Pi$	A conditional plan class constraint that the plan must be in Π under condition e_i (p. 79).
e_ϕ	Expectation under the monotonic transform ϕ (p. 113).
$E_x[g(x)]$	The expectation of $g(x)$ with respect to x. Equivalent to $\int g(x) f_x(x_0) dx_0$ (p. 179).

$\mathcal{E}$	The universal event class. $\mathcal{E} = \mathcal{X}(\textbf{event})$ (p. 75).
$\mathcal{E}_i$	A generic set of events. $\mathcal{E}_i = \mathcal{X}(e_i)$ (p. 79).
ev	An event variable type (p. 75).
EV	An event variable instance of type ev (p. 75).
$EV(obs)$	The value taken by the observable event variable instance EV in the observation obs (p. 75).
event	The root of the event taxonomy (p. 75).
f_b	Probability density function for b (p. 96).
$f_b(b_0\|x)$	Conditional probability density function for condition x evaluated at $b = b_0$ (p. 96).
F_b	Cumulative density function (CDF) for b (p. 95).
$F_b(\cdot\|x)$	Conditional CDF for b given x (p. 96).
FSD	First-order stochastic dominance relation (p. 95).
G	1. A goal predicate (p. 18).
	2. A qualitative probabilistic network (p. 89).
id_i	Unique identifier associated with an action type in a plan class *in-list* (p. 76).
K^{δ_1,δ_2}	A Markov influence assertion with influence direction δ_1 and synergy direction δ_2 (p. 139).
$\mathcal{O}$	The set of possible patterns of observation over time (p. 68).
obs	An observation in $\mathcal{O}$ (p. 68).
$\mathcal{P}$	The set of plan classes expressible in the constraint language, $\mathcal{P} \subseteq 2^{\Omega}$ (p. 52).
$\mathcal{P}_i$	The plan classes expressible in planner-i's constraint language in the cascaded planning architecture (p. 70).
$\mathcal{P}_E$	The plan classes accepted by the executor (p. 70).
$pred$	Predecessor function (p. 93).

Q	The set of qualitative relations in a QPN (p. 89).
r_j^δ	The extreme value in role r_j's range, maximum if $\delta = +$ and minimum if $\delta = -$ (p. 190).
R	A generic relation on CDFs (p. 104).
R_δ	Ordinal comparison relation holding for undefined terms by default (p. 78).
$\mathbf{R}$	Annotation on a plan graph link indicating a restriction relation (p. 26).
$\Re$	The real numbers (p. 52).
$red(b, G)$	The network obtained by reducing b from G (p. 99).
$rev(a, b, G)$	The network obtained by reversing a and b in G (p. 99).
result	Function returning the state resulting from performing a plan in a given state in the situation calculus (p. 45).
$role\langle a_i \rangle$	The action variable formed from concept a_i and role *role* (p. 73).
s_i	The initial situation in situation calculus planning (p. 45).
S_a	Set of event variables directly affected by action a (p. 134).
S'_a	Set of event variables directly affected by action a, plus any other variables that affect those in S_a (p. 135).
S^δ	A qualitative influence assertion (pp. 92, 93, 96).
u	The utility function (pp. 96, 179).
U^δ	Qualitative influences on utility (p. 96).
v	The specially designated value node in a QPN (p. 89).
V	The set of variables in a QPN (p. 89).
x	Context variable in qualitative relation definition (p. 93).
$X(a)$	The domain of variable a (p. 89).
$\mathcal{X}(c)$	The extension of a (NIKL) concept c (p. 71).
Y^δ	A qualitative synergy assertion (pp. 109, 110).

Y_U^δ	Qualitative synergy on utility (p. 114).
δ	A qualitative direction, one of $+$, $-$, ?, or 0 (pp. 78, 92).
(δ, av, ev)	A monotonic policy constraint in direction δ between av and ev (p. 78).
θ	A state of nature (pp. 118, 179).
Θ	The set of possible states of nature (p. 179).
π	A plan (p. 45, defined in Section 3.1).
$\pi[e_i]$	A partial plan function defined only on the extension $\mathcal{E}_i$ of e_i (p. 79).
Π	A plan class or partial plan (p. 43).
$\Pi^\delta(av, ev)$	The plan class where av is a monotonic function (with direction δ) of ev (p. 54).
$\Pi[e_i]$	A class of partial plan functions defined only on the extension $\mathcal{E}_i$ of e_i. (p. 80).
ϕ	A monotonic transform (p. 95).
ω	A course of action in Ω_{exe} (p. 68).
Ω	The universal plan class, or set of all syntactically valid plans (p. 43).
Ω_0	The set of (semantically) legal plans (p. 69).
Ω_{exe}	The set of executable courses of action (p. 68).
$\succ$	Strict preference relation over plans (p. 45).
$\succeq$	Non-strict preference relation over plans (p. 45).
$\sim$	Indifference relation over plans (p. 45).
$\otimes$	Sign multiplication operator (Table 4.1, p. 100).
$\oplus$	Sign addition operator (Table 4.1, p. 100).

Appendix B

Proofs of QPN Results

Lemma 4.9 *If $b \notin pred^*_G(a)$ then a and b are d-separated in $dep(G)$ by any S such that $pred_G(b) \subseteq S \subseteq pred^*_G(b)$.*

Proof: Two variables are d-separated iff every undirected path between them is blocked according to one of the conditions of Definition 4.8. Every path between a and b must pass through one of b's predecessors or one of its successors. Because $pred_G(b) \subseteq S$, the paths through the predecessors are blocked by the first condition. Consider a path through a successor of b. Let t be the first variable on the path, starting from b, that has both incident edges leading in. Such a variable must exist because $b \notin pred^*_G(a)$. Because it is the first, there is a directed path to it from b. Note that every variable in the separating set S has a directed path to b because $S \subseteq pred^*_G(b)$. Therefore if t or any of its successors were in S, there would be a cycle from b through t. Because $dep(G)$ is acyclic, t blocks the path via the second condition of Definition 4.8. □

Theorem 4.10

$$S^{\delta_1}(a,b,G) \wedge S^{\delta_2}(b,c,G) \wedge S^0(a,c,G) \Rightarrow S^{\delta_1 \otimes \delta_2}(a,c,red(b,G)),$$

where $\delta_i \in \{+,-,0,?\}$ and $\otimes$ denotes sign multiplication, described by Table 4.1.

Proof: I will prove the case $\delta_1 = \delta_2 = +$; the others are analogous. Choose a_1 and a_2 such that $a_1 \geq a_2$, and an x_0 in $X(pred(b) \cup pred(c) - \{a, b\})$ that is consistent with a_1 and a_2. Let F_c denote the conditional CDF for c and $\underline{c}$ the minimal value of the variable. By the definition of cumulative probability we have

$$F_c(c_0|a_ix_0) = \int_{\underline{c}}^{c_0} \int f_{bc}(b_0c_1|a_ix_0)db_0dc_1.$$

Changing the order of integration and decomposing the joint probability yields[1]

$$F_c(c_0|a_ix_0) = \int \int_{\underline{c}}^{c_0} f_c(c_1|a_ib_0x_0)f_b(b_0|a_ix_0)dc_1db_0. \tag{B.1}$$

Because a and c are conditionally independent given b and x, by the S^0 premise and Lemma 4.9, we can remove a_i from the f_c expression. Rewriting the density function as the derivative of a cumulative, we get

$$F_c(c_0|a_ix_0) = \int \int_{\underline{c}}^{c_0} f_c(c_1|b_0x_0)dc_1dF_b(b_0|a_ix_0). \tag{B.2}$$

The inner integral is simply the CDF for c given b_0.

$$F_c(c_0|a_ix_0) = \int F_c(c_0|b_0x_0)dF_b(b_0|a_ix_0). \tag{B.3}$$

Because b positively influences c, the pointwise FSD condition (4.3) implies that for any c_0, $F_c(c_0|b_0x_0)$ is a *decreasing* function of b_0. And $S^+(a,b)$ entails FSD of $F_b(b_0|a_1x_0)$ over $F_b(b_0|a_2x_0)$. Therefore, (4.4) applies with the inequality reversed (negating $F_c(c_0|bx_0)$ yields an increasing function), leading to the conclusion

$$\forall c_0 \quad F_c(c_0|a_1x_0) \leq F_c(c_0|a_2x_0),$$

implying FSD. Because a_1, a_2, and x_0 were chosen arbitrarily, we have finally $S^+(a,c)$.
□

[1]If some values of b_0 are inconsistent with x_0, then distributions of c conditioned on b_0 and x_0 (and therefore the right-hand sides of equations (B.1), (B.2), and (B.3)) are not well-defined. This has no consequence, however, because the value of $f_b(b_0|a_ix_0)$ in such cases will always be zero.

Theorem 4.11

$$S^{\delta_1}(a,b,G) \wedge S^{\delta_2}(b,c,G) \wedge S^{\delta_3}(a,c,G) \Rightarrow S^{(\delta_1 \otimes \delta_2) \oplus \delta_3}(a,c,red(b,G)),$$

where $\oplus$ denotes sign addition, also described in Table 4.1.

Proof: Proceed as for the proof of Theorem 4.10 to equation (B.1). Because δ_3 is not generally zero, we cannot remove a_i in the next two steps.

$$F_c(c_0|a_i x_0) = \int F_c(c_0|a_i b_0 x_0) dF_b(b_0|a_i x_0).$$

Define $\hat{F}_c$ as a variant where a_i is fixed to a_1 in the first term

$$\hat{F}_c(c_0|a_i x_0) = \int F_c(c_0|a_1 b_0 x_0) dF_b(b_0|a_i x_0).$$

Note that $\hat{F}_c(c_0|a_1 x_0) = F_c(c_0|a_1 x_0)$ and that

$$\delta_3 = +(-) \Rightarrow \forall c_0 \;\; F_c(c_0|a_1 b_0 x_0) \le (\ge) \; F_c(c_0|a_2 b_0 x_0), \text{ therefore}$$

$$\forall c_0 \;\; \hat{F}_c(c_0|a_2 x_0) \le (\ge) \; F_c(c_0|a_2 x_0). \tag{B.4}$$

When $\delta_3 = ?$ it is possible that the relation varies with c_0. Regardless of δ_3, $F_c(c_0|a_1 b_0 x_0)$ is a decreasing/increasing/non-monotonic function of b_0 as δ_2 is $+/-/?$. For concreteness, suppose $\delta_1 = \delta_2 = +$ (again, the other cases are analogous). Following the reasoning in the proof of Theorem 4.10 above, we get

$$\hat{F}_c(c_0|a_1 x_0) = F_c(c_0|a_1 x_0) \text{ FSD } \hat{F}_c(c_0|a_2 x_0).$$

If $\delta_3 = +$ (more generally if δ_3 agrees with the polarity of the FSD relation), this result combines with (B.4) to imply FSD of the corresponding unhatted F_cs, thereby establishing the result. Without such agreement FSD may be violated, permitting us to conclude only $S^?(a,c,red(b,G))$. $\Box$

Theorem 4.12 *Let $G' = rev(a,b,G)$. G' inherits all the qualitative influences of G except:*

1. *$dir(a,b,G')$ is undefined.*

2. *$dir(b,a,G') = dir(a,b,G)$.*

3. *$\forall w \in pred_{G'}(b)$,*

$$dir(w,b,G') = [dir(w,a,G) \otimes dir(a,b,G)] \oplus dir(w,b,G).$$

4. *$\forall w \in pred_{G'}(a) - \{b\}$,*

$$dir(w,a,G') = \begin{cases} dir(w,a,G) & \textit{if } dir(w,b,G) = 0 \\ ? & \textit{otherwise.} \end{cases}$$

Proof: First, note that all variables outside $pred_G(a) \cup pred_G(b)$ retain the same set of d-separations. Second, let us verify each relation above:

1. There is no longer an influence from a to b.

2. To show that the influence on the reversed link remains unchanged it is convenient to work with the likelihood form of S^δ, equation (4.6). Applying Bayes's formula:

$$f_b(b_j|a_ix) = \frac{f_a(a_i|b_jx)f_b(b_j|x)}{f_a(a_i|x)}.$$

Choose four values $a_1 \geq a_2$ and $b_1 \geq b_2$.

$$\frac{f_b(b_1|a_ix)}{f_b(b_2|a_ix)} = \frac{f_a(a_i|b_1x)f_b(b_1|x)}{f_a(a_i|b_2x)f_b(b_2|x)} = g(b_1,b_2,x)\frac{f_a(a_i|b_1x)}{f_a(a_i|b_2x)}.$$

Using the monotone likelihood property, $dir(a,b,G) = +(-)$ implies

$$\frac{f_b(b_1|a_1x)}{f_b(b_2|a_1x)} \geq (\leq) \frac{f_b(b_1|a_2x)}{f_b(b_2|a_2x)}.$$

Rearranging we get

$$\frac{f_b(b_1|a_1x)}{f_b(b_1|a_2x)} \geq (\leq) \frac{f_b(b_2|a_1x)}{f_b(b_2|a_2x)},$$

the MLRP for b given a. As noted above (and proven by Milgrom [90]), this is necessary and sufficient for our posterior FSD condition to hold for any prior $F_a(a_0|x)$.

3. In G, the influence of w on b is relative to a predecessor set that includes a. In G' the influence is not so conditioned and is therefore equivalent to the influence on b obtained by splicing a out of the network. Applying Theorem 4.11 with the original influences yields the expression above.

4. Here we are transforming an unconditional relation to a conditional one. If $dir(w,b,G) = 0$, w and b are d-separated by $pred(b)$ in $dep(G)$ (by Lemma 4.9), therefore $f_w(w_0|abx) = f_w(w_0|ax)$ by conditional independence. In that case the MLRP obviously holds for the conditional density iff it holds for the marginal one. If w has nonzero influence on b in G, this independence does not hold. Because a and w may interact significantly in their influence on b we cannot say anything about their relation given b. For example, let the three variables be binary with a and w marginally independent (that is, $dir(w,a,G) = 0$), $\Pr(A) = \Pr(W) = .5$, $\Pr(B|\bar{A}\bar{W}) = .1$, $\Pr(B|\bar{A}W) = .2$, and $\Pr(B|AW) = .9$. Then $dir(w,a,G')$ can be $+$ or $-$ depending on whether $\Pr(B|A\bar{W})$ is less than or greater than .45. Either possibility is consistent with an initial G with $dir(a,b) = dir(w,b) = +$.

□

Theorem 4.13 *Let $S^+(a,b)$ be defined by (4.10). Given the following conditions:*

1. *Theorem 4.10*

2. For binary b, $a_1 \geq a_2$, and x,

$$F_b(\cdot|a_1x)\ R\ F_b(\cdot|a_2x) \Leftrightarrow \Pr(B|a_1x) \geq \Pr(B|a_2x) \tag{B.5}$$

the weakest R is FSD.

Proof: First, note that FSD satisfies these conditions. Next, assume that R satisfies them but R does not entail FSD. We will start with an instantiation of Theorem 4.10 and derive a contradiction. Let a, b, and c be the only variables (so we can safely ignore x) with $S^+(a,b)$, $S^+(b,c)$, and no other direct links. For concreteness, let b range over the unit interval $[0,1]$ and c be binary with $\Pr(C|ab) = \phi(b)$, for some $\phi : [0,1] \to [0,1]$ monotonic. The monotonicity of ϕ guarantees $S^+(b,c)$ and its independence from a validates $S^0(a,c)$ in the original network. By assumption, Theorem 4.10 applies, yielding the conclusion $S^+(a,c)$ and therefore $F_c(c_0|a_1)\ R\ F_c(c_0|a_2)$. Because c is binary, (B.5) must hold. Using $\Pr(C|a_i) = \int \Pr(C|a_ib_0)dF_b(b_0|a_i)$, the RHS of (B.5) becomes

$$\int_0^1 \phi(b_0)dF_b(b_0|a_1) \geq \int_0^1 \phi(b_0)dF_b(b_0|a_2). \tag{B.6}$$

Because ϕ may be any monotonic function, FSD is necessary for (B.6) and is therefore entailed by R. □

Theorem 4.14 *Suppose $U^{\delta_2}(b,G)$ and $U^0(a,G)$. A necessary and sufficient condition for $U^{\delta_1 \otimes \delta_2}(a, red(b,G))$ is $S^{\delta_1}(a,b,G)$ as in Definition 4.4.*

Proof: The expected utility of a_i with any x is given by

$$u(a_i,x) = \int u(b_0,x)dF_b(b_0|a_ix). \tag{B.7}$$

Let us prove the case $\delta_1 = \delta_2 = +$. $U^+(a)$ is satisfied in the reduced network iff $u(a_i,x)$ is increasing in a_i. From (4.8) we know that $u(b_0,x)$ is monotonically increasing in b_0. In fact, it can be *any* monotonic function. Therefore, (B.7) is increasing in a_i under the same conditions as (4.4), which is exactly the S^+ condition (4.5) of Definition 4.4. □

Lemma 4.18 *$Y^+(\{a,b\},c)$ (respectively Y^- and Y^0) holds iff the function*

$$e_\phi(a,b|x) = \int \phi(c_0) f_c(c_0|abx) dc_0 \tag{B.8}$$

is supermodular (submodular, modular) in a and b for all increasing functions ϕ and contexts x.

Proof: Choose arbitrary $a_1 \geq a_2$, $b_1 \geq b_2$, and x. By Definition 4.17, e_ϕ is supermodular iff

$$e_\phi(a_1,b_1|x) + e_\phi(a_2,b_2|x) \geq e_\phi(a_1,b_2|x) + e_\phi(a_2,b_1|x).$$

Rearranging,

$$e_\phi(a_1,b_1|x) - e_\phi(a_2,b_1|x) \geq e_\phi(a_1,b_2|x) - e_\phi(a_2,b_2|x).$$

Substituting the definition of e_ϕ (B.8) and combining the integrals,

$$\int \phi(c_0)\,[f_c(c_0|a_1b_1x) - f_c(c_0|a_2b_1x)]\,dc_0 \geq \int \phi(c_0)\,[f_c(c_0|a_1b_2x) - f_c(c_0|a_2b_2x)]\,dc_0. \tag{B.9}$$

A necessary and sufficient condition for (B.9) to hold for any increasing function ϕ is that the bracketed distribution differences be related by FSD. (Recall the equivalence between (4.3) and (4.4) in Section 4.3.2.) That is,

$$\forall c_0 \quad F_c(c_0|a_1b_1x) - F_c(c_0|a_2b_1x) \leq F_c(c_0|a_1b_2x) - F_c(c_0|a_2b_2x).$$

This is exactly the Y^+ condition of Definition 4.16. □

Theorem 4.20 *Synergies can be extended along qualitative influences by reduction according to the following.*

$$Y^{\delta_1}(\{a,b\},c,G) \wedge S^{\delta_2}(c,d,G) \wedge S^0(a,d,G) \wedge S^0(b,d,G) \Rightarrow$$
$$Y^{\delta_1 \otimes \delta_2}(\{a,b\},d,red(c,G)).$$

Proof: Let us assume that $\delta_1 = \delta_2 = +$; the other cases are analogous. We can describe the cumulative for d conditional on a and b by integrating over its counterpart for c.

$$\begin{aligned} F_d(d_0|abx) &= \int_{\underline{d}}^{d_0} \int f_d(d_1|abc_0x) f_c(c_0|abx) dc_0 dd_1 & \text{(B.10)} \\ &= \int \int_{\underline{d}}^{d_0} f_d(d_1|c_0x) dd_1 \; f_c(c_0|abx) dc_0 & \text{(B.11)} \\ &= \int F_d(d_0|c_0x) f_c(c_0|abx) dc_0. & \text{(B.12)} \end{aligned}$$

In going from (B.10) to (B.11) I took advantage of the conditional independence between d and each of a and b given c implied by the S^0 conditions and Lemma 4.9. Because $S^+(c,d)$, $F_d(d_0|c_0)$ is a decreasing function of c_0 for any d_0. Therefore, equation (B.12) and Lemma 4.18 imply that $F_d(d_0|abx)$ is a *sub*modular function of a and b for all d_0 (a function g is submodular iff $-g$ is supermodular). By the definition of submodularity,

$$\forall d_0 \quad F_d(d_0|a_1b_1x) - F_d(d_0|a_2b_1x) \leq F_d(d_0|a_1b_2x) - F_d(d_0|a_2b_2x), \qquad \text{(B.13)}$$

which is the condition for $Y^+(\{a,b\},d)$ of Definition 4.16. □

Theorem 4.21

$$\begin{aligned} &Y^{\delta_1}(\{a,b\},c,G) \wedge S^{\delta_2}(c,d,G) \wedge Y^{\delta_3}(\{a,c\},d,G) \wedge Y^{\delta_4}(\{b,c\},d,G) \\ &\quad \wedge S^{\delta_5}(a,c,G) \wedge S^{\delta_6}(b,c,G) \wedge Y^{\delta_7}(\{a,b\},d,G) \Rightarrow \\ &\qquad\qquad Y^{(\delta_1 \otimes \delta_2) \oplus (\delta_3 \otimes \delta_6) \oplus (\delta_4 \otimes \delta_5) \oplus \delta_7}(\{a,b\},d,red(c,G)). \end{aligned}$$

Proof: Start as in the proof of Theorem 4.20, but do not use conditional independence.

$$\begin{aligned} F_d(d_0|abx) &= \int_{\underline{d}}^{d_0} \int f_d(d_1|abc_0x) f_c(c_0|abx) dc_0 dd_1 \\ &= \int F_d(d_0|abc_0x) dF_c(c_0|abx). \end{aligned}$$

As in the proof of Theorem 4.11, define $\hat{F}_d$ to be the CDF with the conditioning variables fixed in the first term, to a_1 and b_1 in this case.

$$\hat{F}_d(d_0|abx) = \int F_d(d_0|a_1b_1c_0x)dF_c(c_0|abx).$$

Regardless of a_1 and b_1, $F_d(d_0|a_1b_1c_0x)$ has monotonicity properties determined solely by δ_2. Following the reasoning of the Theorem 4.20 proof, we have the following fact about $\hat{F}_d$ (a hatted version of (B.13)):

$$\forall d_0 \quad \hat{F}_d(d_0|a_1b_1x) - \hat{F}_d(d_0|a_2b_1x) \;\; R \;\; \hat{F}_d(d_0|a_1b_2x) - \hat{F}_d(d_0|a_2b_2x), \tag{B.14}$$

with R the relation $\leq$, $\geq$, $=$, or ? as $\delta_1 \otimes \delta_2$ is $+$, $-$, 0, or ?. Henceforth I will refer to functions satisfying conditions of the form (B.14) as *R-modular*. Let $\hat{F}'_d$ be intermediate between F_d and $\hat{F}_d$ where only b is fixed

$$\hat{F}'_d(d_0|abx) = \int F_d(d_0|ab_1c_0x)dF_c(c_0|abx).$$

Note that $\hat{F}'_d(d_0|a_1b_ix) = \hat{F}_d(d_0|a_1b_ix)$ for either b_i. Therefore $\hat{F}'_d$ is R-modular iff

$$\forall d_0 \quad \hat{F}_d(d_0|a_1b_1x) - \hat{F}'_d(d_0|a_2b_1x) \;\; R \;\; \hat{F}_d(d_0|a_1b_2x) - \hat{F}'_d(d_0|a_2b_2x). \tag{B.15}$$

Using (B.14) and a little rearrangement, a sufficient condition for (B.15) is

$$\forall d_0 \quad \Delta_1(d_0) \;\; R \;\; \Delta_2(d_0), \text{ where} \tag{B.16}$$

$$\Delta_i(d_0) \equiv \hat{F}_d(d_0|a_2b_ix) - \hat{F}'_d(d_0|a_2b_ix) \tag{B.17}$$

Expanding the definitions for $\hat{F}_d$ and $\hat{F}'_d$,

$$\Delta_i(d_0) = \int \left[F_d(d_0|a_1b_1c_0x) - F_d(d_0|a_2b_1c_0x)\right] dF_c(c_0|a_2b_ix). \tag{B.18}$$

The difference inside the integral of equation (B.18) is an increasing, decreasing, or constant function of c_0 as δ_3, the synergy of a and c, is $-$, $+$, or 0. The influence of b on

c, δ_6, determines an FSD relation among the $F_c(c_0|ab_i)$. Therefore, condition (B.16) holds if $\delta_3 \otimes \delta_6$ agrees with R, which was determined by $\delta_1 \otimes \delta_2$.

Another application of this line of reasoning with the roles of a and b reversed leads to the conclusion that $\hat{F}''_d$, where

$$\hat{F}''_d(d_0|abx) = \int F_d(d_0|a_1bc_0x)dF_c(c_0|abx).$$

is R-modular if $\delta_4 \otimes \delta_5$ agrees with R. Thus, agreement among these pairwise products yields R-modularity of $\hat{F}_d$, $\hat{F}'_d$, and $\hat{F}''_d$.

Suppose that δ_7 also agrees with R. Then, from the Y^δ definition we have

$$\forall d_0 \quad F_d(d_0|a_1b_1c_0x) - F_d(d_0|a_2b_1c_0x) \;\; R \;\; F_d(d_0|a_1b_2c_0x) - F_d(d_0|a_2b_2c_0x),$$

which entails the following inequality when integrating over a positive function:

$$\begin{aligned}\forall d_0 \quad & \int \left[F_d(d_0|a_1b_1c_0x) - F_d(d_0|a_2b_1c_0x)\right] dF_c(c_0|a_2b_2x) \\ R \quad & \int \left[F_d(d_0|a_1b_2c_0x) - F_d(d_0|a_2b_2c_0x)\right] dF_c(c_0|a_2b_2x).\end{aligned}$$

Equivalently,

$$\forall d_0 \quad \hat{F}_d(d_0|a_2b_2x) - \hat{F}'_d(d_0|a_2b_2x) \;\; R \;\; \hat{F}''_d(d_0|a_2b_2x) - F_d(d_0|a_2b_2x). \tag{B.19}$$

We can transform (B.19) to a relation on F_d alone by applying some R-modularity conditions already known and taking advantage of the equivalences among the hatted and primed Fs for particular values of a and b. Combining (B.19) with R-modularity of $\hat{F}''_d$,

$$\begin{aligned}\forall d_0 \quad & F_d(d_0|a_1b_1x) - \hat{F}''_d(d_0|a_2b_1x) + \hat{F}_d(d_0|a_2b_2x) - \hat{F}'_d(d_0|a_2b_2x) \\ & R \;\; F_d(d_0|a_1b_2x) - F_d(d_0|a_2b_2x).\end{aligned}$$

Applying R-modularity of $\hat{F}'_d$ yields

$$\forall d_0 \quad 2F_d(d_0|a_1b_1x) - F_d(d_0|a_2b_1x) - \hat{F}''_d(d_0|a_2b_1x) + \hat{F}_d(d_0|a_2b_2x)$$

$$R\ \ F_d(d_0|a_1b_2x) - F_d(d_0|a_2b_2x) + \hat{F}'_d(d_0|a_1b_2x),$$

and finally, R-modularity of $\hat{F}$ leads to the result

$$\forall d_0\quad F_d(d_0|a_1b_1x) - F_d(d_0|a_2b_1x)\ \ R\ \ F_d(d_0|a_1b_2x) - F_d(d_0|a_2b_2x).$$

Therefore, unanimity among the terms in the new synergy expression given by the theorem statement implies R-modularity of F_d, the condition of interest. Dissent by any term results in a synergy of $Y^?$, vacuously true. $\Box$

Appendix C

SUDO-Planner Knowledge Base

This appendix provides a complete description of SUDO-PLANNER's KB for the running example.

C.1 Event Taxonomy

Figure C.1 is a graphical view of the event taxonomy, which includes the action taxonomy fragment depicted in Figure 3.4. The taxonomy consists of NIKL concepts representing event types. All of SUDO-PLANNER's event variables are defined by combining these concepts with the NIKL roles depicted in Figure C.2. The specialization relations in these two taxonomies define the taxonomic dimension of the event variable KB described in Section 6.2.

C.2 Qualitative Relation Assertions

Table C.1 lists the qualitative influence (S^{δ}) assertions in SUDO-PLANNER's KB. A subset of these assertions appear in graphical format in the event variable KB fragment of Figure 6.3. Qualitative synergy (Y^{δ}) assertions are listed in Table C.2 and Markov influences (K^{δ_1,δ_2}) in Table C.3.

C.3 Observable Creators

The SUDO-PLANNER KB requires only two tests for this example (see Table 1.1):

$CO(\textbf{cardiac-catheterization(true)}, \textbf{extent}\langle\textbf{cardiac-cath-result}\rangle$

$CO(\textbf{carotid-arteriography(true)}, \textbf{extent}\langle\textbf{carotid-arteriography-result}\rangle$

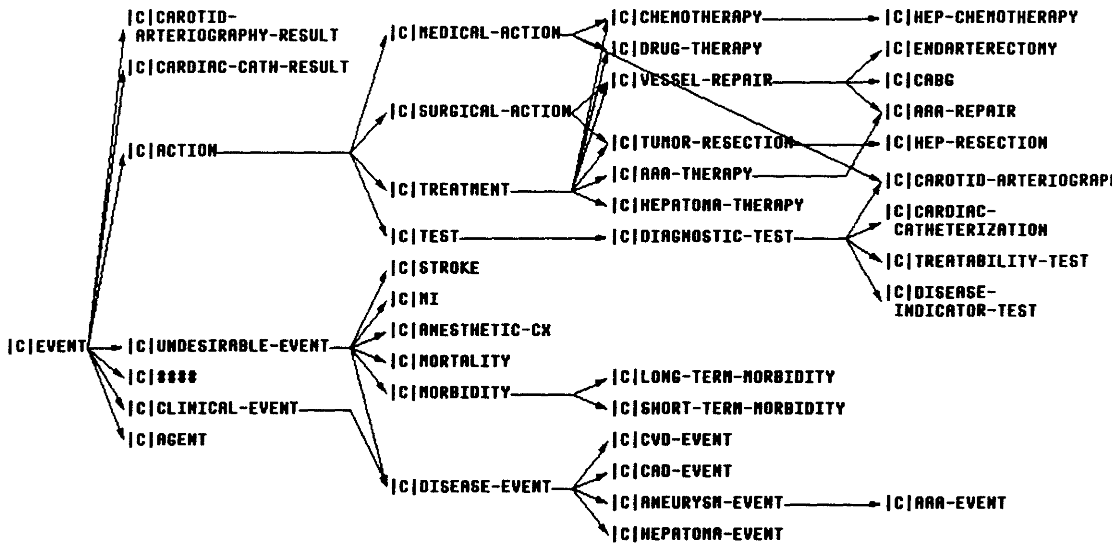


Figure C.1: SUDO-PLANNER's event taxonomy. The taxonomy also includes several event types not used in the running example.

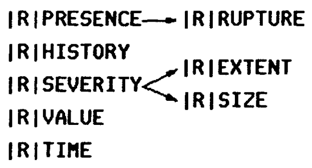


Figure C.2: Roles used to define event variables in the running example.

presence⟨undesirable-event⟩	$\xrightarrow{-}$	**value⟨agent⟩**
time⟨undesirable-event⟩	$\xrightarrow{-}$	**value⟨agent⟩**
severity⟨morbidity⟩	$\xrightarrow{-}$	**value⟨agent⟩**
history⟨disease-event⟩	$\xrightarrow{+}$	**severity⟨disease-event⟩**
presence⟨treatment⟩	$\xrightarrow{?}$	**value⟨agent⟩**
presence⟨surgery⟩	$\xrightarrow{+}$	**severity⟨short-term-morbidity⟩**
presence⟨surgery⟩	$\xrightarrow{+}$	**time⟨mortality⟩**
presence⟨surgery⟩	$\xrightarrow{?}$	**value⟨agent⟩**
presence⟨vessel-repair⟩	$\xrightarrow{?}$	**value⟨agent⟩**
presence⟨medical-action⟩	$\xrightarrow{?}$	**value⟨agent⟩**
presence⟨anesthetic-cx⟩	$\xrightarrow{+}$	**time⟨mortality⟩**
presence⟨surgery⟩	$\xrightarrow{+}$	**presence⟨anesthetic-cx⟩**
rupture⟨aneurysm-event⟩	$\xrightarrow{+}$	**time⟨mortality⟩**
size⟨aneurysm-event⟩	$\xrightarrow{+}$	**rupture⟨aneurysm-event⟩**
size⟨AAA-event⟩	$\xrightarrow{+}$	**rupture⟨AAA-event⟩**
presence⟨AAA-repair⟩	$\xrightarrow{-}$	**rupture⟨AAA-event⟩**
history⟨CAD-event⟩	$\xrightarrow{+}$	**extent⟨CAD-event⟩**
presence⟨MI⟩	$\xrightarrow{+}$	**time⟨mortality⟩**
presence⟨stroke⟩	$\xrightarrow{+}$	**time⟨mortality⟩**
presence⟨stroke⟩	$\xrightarrow{+}$	**severity⟨long-term-morbidity⟩**
presence⟨vessel-repair⟩	$\xrightarrow{+}$	**presence⟨MI⟩**
presence⟨vessel-repair⟩	$\xrightarrow{+}$	**presence⟨stroke⟩**
presence⟨vessel-repair⟩	$\xrightarrow{+}$	**time⟨mortality⟩**
presence⟨AAA-repair⟩	$\xrightarrow{+}$	**time⟨mortality⟩**
(extent⟨CAD-event⟩	$\xrightarrow{+}$	**presence⟨MI⟩**
(extent⟨cardiac-cath-result⟩	$\xrightarrow{+}$	**extent⟨CAD-event⟩**
history⟨CVD-event⟩	$\xrightarrow{+}$	**extent⟨CVD-event⟩**
extent⟨CVD-event⟩	$\xrightarrow{+}$	**presence⟨stroke⟩**
extent⟨carotid-arteriography-result⟩	$\xrightarrow{+}$	**extent⟨CVD-event⟩**

Table C.1: Qualitative influences in the SUDO-PLANNER KB.

severity⟨long-term-morbidity⟩ **time⟨mortality⟩**	$\boxed{+}\!\longrightarrow$	**value⟨agent⟩**
presence⟨AAA-repair⟩ **size⟨AAA-event⟩**	$\boxed{-}\!\longrightarrow$	**rupture⟨AAA-event⟩**
presence⟨CABG⟩ **presence⟨AAA-repair⟩** **presence⟨endarterectomy⟩**	$\boxed{-}\!\longrightarrow$	**presence⟨anesthetic-cx⟩**
presence⟨vessel-repair⟩ **rupture⟨aneurysm-event⟩** **presence⟨MI⟩** **presence⟨stroke⟩** **presence⟨anesthetic-cx⟩**	$\boxed{-}\!\longrightarrow$	**time⟨mortality⟩**
extent⟨CAD-event⟩ **presence⟨AAA-repair⟩**	$\boxed{+}\!\longrightarrow$	**presence⟨MI⟩**
presence⟨CABG⟩ **presence⟨AAA-repair⟩**	$\boxed{-}\!\longrightarrow$	**time⟨mortality⟩**
presence⟨CABG⟩ **presence⟨AAA-repair⟩**	$\boxed{-}\!\longrightarrow$	**presence⟨stroke⟩**
presence⟨CABG⟩ **presence⟨AAA-repair⟩**	$\boxed{-}\!\longrightarrow$	**presence⟨MI⟩**
extent⟨CVD-event⟩ **presence⟨AAA-repair⟩**	$\boxed{+}\!\longrightarrow$	**presence⟨stroke⟩**
presence⟨endarterectomy⟩ **presence⟨AAA-repair⟩**	$\boxed{-}\!\longrightarrow$	**time⟨mortality⟩**
presence⟨endarterectomy⟩ **presence⟨AAA-repair⟩**	$\boxed{-}\!\longrightarrow$	**presence⟨stroke⟩**
presence⟨endarterectomy⟩ **presence⟨AAA-repair⟩**	$\boxed{-}\!\longrightarrow$	**presence⟨MI⟩**

Table C.2: Qualitative synergies in the SUDO-PLANNER KB.

presence⟨CABG⟩	$\boxed{-}\!\xrightarrow{-}$	**extent⟨CAD-event⟩**
presence⟨endarterectomy⟩	$\boxed{-}\!\xrightarrow{-}$	**extent⟨CVD-event⟩**

Table C.3: Markov influences in the SUDO-PLANNER KB.

Bibliography

[1] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[2] Steen Andreassen, Marianne Woldbye, Bjørn Falck, et al. MUNIN: A causal probabilistic network for interpretation of electromyographic findings. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 366–372, 1987.

[3] James O. Berger. *Statistical Decision Theory and Bayesian Analysis.* Springer-Verlag, second edition, 1985.

[4] Daniel G. Bobrow, editor. *Qualitative Reasoning about Physical Systems.* MIT Press, 1985.

[5] Ronald J. Brachman. 'I lied about the trees,' or, Defaults and definitions in knowledge representation. *AI Magazine*, 6(3):80–93, 1985.

[6] Ronald J. Brachman, Richard E. Fikes, and Hector J. Levesque. Krypton: A functional approach to knowledge representation. *Computer*, 16(10):67–73, 1983.

[7] Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of the National Conference on Artificial Intelligence*, pages 34–37. AAAI, 1984.

[8] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9:171–216, 1985.

[9] John S. Breese. Knowledge representation and inference in intelligent decision systems. Research Report 2, Rockwell International Science Center, Palo Alto, April 1987.

[10] Jack Breese and Edison Tse. Integrating logical and probabilistic reasoning for decision making. In *Proceedings of the Workshop on Uncertainty in Artificial Intelligence*, pages 355–362, July 1987.

[11] B. Chandrasekaran and Sanjay Mittal. Deep versus compiled knowledge approaches to diagnostic problem-solving. In *Proceedings of the National Conference on Artificial Intelligence*, pages 349–354. AAAI, August 1982.

[12] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[13] Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence.* Addison-Wesley, Reading, MA, 1985.

[14] Gregory Floyd Cooper. NESTOR: *A Computer-Based Medical Diagnostic Aid that Integrates Causal and Probabilistic Knowledge.* PhD thesis, Stanford University, November 1984.

[15] Gregory F. Cooper. The computational complexity of probabilistic inference using belief networks. Memo KSL-87-27, Knowledge Systems Laboratory, Stanford University, May 1988.

[16] Louis Anthony Cox, Jr. *Mathematical Foundations of Risk Measurement.* PhD thesis, Massachussetts Institute of Technology, May 1986.

[17] Ernest Davis. Order of magnitude reasoning in qualitative differential equations. Technical Report 312, New York University Computer Science Department, August 1987.

[18] Martin Davis and Reuben Hersh. Nonstandard analysis. *Scientific American*, 226(6):78–86, June 1972.

[19] Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[20] Johan de Kleer. Problem solving with the ATMS. *Artificial Intelligence*, 28:197–224, 1986.

[21] Thomas Dean and Mark Boddy. Incremental causal reasoning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 196–201. AAAI, 1987.

[22] Keith L. Downing. Diagnostic improvement through qualitative sensitivity analysis and aggregation. In *Proceedings of the National Conference on Artificial Intelligence*, pages 789–793. AAAI, 1987.

[23] Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12(2):231–272, 1979.

[24] Jon Doyle. A model for deliberation, action, and introspection. TR 581, MIT Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA, 02139, 1980.

[25] Jon Doyle. Artificial intelligence and rational self-government. Technical Report CS-88-124, Carnegie-Mellon University Computer Science Department, 1988.

[26] Van H. Dunn. Grand rounds, Beth Israel hospital. Unpublished decision analysis consult report, Division of Clinical Decision Making, Tufts-New England Medical Center, 1984.

[27] Richard Engelbrecht. A note on multivariate risk and separable utility functions. *Management Science*, 23:1143–1144, 1977. Note on Richard [112].

[28] Ronald Fagin, Joseph Y. Halpern, and Nimrod Megiddo. A logic for reasoning about probabilities. Research Report RJ 6190, IBM, April 1988.

[29] Jerome A. Feldman and Robert F. Sproull. Decision Theory and Artificial Intelligence II: The hungry monkey. *Cognitive Science*, 1:158–192, 1977.

[30] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[31] Peter C. Fishburn. Analysis of decisions with incomplete knowledge of probabilities. *Operations Research*, 13:217–237, 1965.

[32] Peter C. Fishburn. Von Neumann-Morganstern utility functions on two attributes. *Operations Research*, 22:35–45, 1974.

[33] Peter C. Fishburn and Raymond G. Vickson. Theoretical foundations of stochastic dominance. In Whitmore and Findlay [162].

[34] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[35] Kenneth D. Forbus. Interpreting measurements of physical systems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 113–117. AAAI, 1986.

[36] Kenneth D. Forbus. QPE: Using assumption-based truth maintenance for qualitative simulation. *International Journal of AI in Engineering*, 1988.

[37] Mark S. Fox. Constraint-directed search: A case study of job-shop scheduling. Technical Report CMU-RI-TR-83-22, CMU-CS-83-161, The Robotics Institute, Carnegie-Mellon University, December 1983.

[38] Mark S. Fox, Brad Allen, and Gary Strohm. Job-shop scheduling: An investigation in constraint-directed reasoning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 155–158. AAAI, 1982.

[39] Peter Gärdenfors. Qualitative probability as an intensional logic. *Journal of Philosophical Logic*, 4:171–185, 1975.

[40] Michael P. Georgeff and Amy L. Lansky, editors. *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop.* Morgan Kaufmann, 1986.

[41] I. J. Good. The Bayesian influence, or How to sweep subjectivism under the carpet. In *Good Thinking: The Foundations of Probability and Its Applications* [42]. Originally appeared in 1973.

[42] I. J. Good. *Good Thinking: The Foundations of Probability and Its Applications.* University of Minnesota Press, 1983.

[43] I. J. Good. On the principle of total evidence. In *Good Thinking: The Foundations of Probability and Its Applications* [42]. Originally appeared in *British Journal of Philosophy of Science*, 17:319-321, 1967.

[44] G. Anthony Gorry, Howard Silverman, and Stephen G. Pauker. Capturing clinical expertise: A computer program that considers clinical responses to digitalis. *American Journal of Medicine*, 64:452–460, March 1978.

[45] Benjamin N. Grosof. Non-monotonicity in probabilistic reasoning. In Lemmer and Kanal [79], pages 237–249.

[46] Ira J. Haimowitz, Ramesh S. Patil, and Peter Szolovits. Representing medical knowledge in a terminological language is difficult. In *Symposium on Computer Applications in Medical Care*, 1988.

[47] Joseph Y. Halpern and David A. McAllester. Likelihood, probability, and knowledge. In *Proceedings of the National Conference on Artificial Intelligence*, pages 137–141. AAAI, 1984.

[48] Joseph Y. Halpern and Michael O. Rabin. A logic to reason about likelihood. *Artificial Intelligence*, 32:379–405, 1987.

[49] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[50] Gordon B. Hazen. Partial information, dominance, and potential optimality in multiattribute utility theory. *Operations Research*, 34:296–310, 1986.

[51] David E. Heckerman and Eric J. Horvitz. The myth of modularity in rule-based systems for reasoning with uncertainty. In Lemmer and Kanal [79], pages 23–34.

[52] Max Henrion. Practical issues in constructing a Bayes' belief network. In *Proceedings of the Workshop on Uncertainty in Artificial Intelligence*, pages 132–139, July 1987.

[53] Max Henrion and Daniel R. Cooley. An experimental comparison of knowledge engineering for expert systems and for decision analysis. In *Proceedings of the National Conference on Artificial Intelligence*, pages 471–476. AAAI, 1987.

[54] Carl Hewitt. Offices are open systems. *ACM Transactions on Office Information Systems*, 4:271–287, 1986.

[55] Jerry R. Hobbs. Granularity. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 432–435, 1985.

[56] J. P. Hollenberg. The decision tree builder: An expert system to simulate medical prognosis and management. *Medical Decision Making*, 4(4), 1984. Abstract from the Sixth Annual Meeting of the Society for Medical Decision Making.

[57] Samuel Holtzman. *Intelligent Decision Systems*. PhD thesis, Stanford University, March 1985.

[58] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2:225–231, 1973.

[59] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[60] Eric J. Horvitz, John S. Breese, and Max Henrion. Decision theory in expert systems and artificial intelligence. *Journal of Approximate Reasoning*, 1988.

[61] Ronald A. Howard and James E. Matheson. Influence diagrams. In *The Principles and Applications of Decision Analysis* [62], pages 719–762.

[62] Ronald A. Howard and James E. Matheson, editors. *The Principles and Applications of Decision Analysis.* Strategic Decisions Group, Menlo Park, CA, 1984.

[63] Patrick Humphreys and Wendy McFadden. Experiences with MAUD: Aiding decision structuring versus bootstrapping the decision maker. *Acta Psychologica*, 45:51–69, 1980.

[64] Paul Humphreys. Cutting the causal chain. *Pacific Philosophical Quarterly*, 61:305–314, 1980.

[65] Harry B. Hunt, III, Daniel J. Rosenkrantz, and Thomas G. Szymanski. On the equivalence, containment, and covering problems for the regular and context-free languages. *Journal of Computer and System Sciences*, 12:222–268, 1976.

[66] Thomas S. Kaczmarek, Raymond Bates, and Gabriel Robins. Recent developments in NIKL. In *Proceedings of the National Conference on Artificial Intelligence*, pages 978–985. AAAI, 1986.

[67] Samuel Karlin and Herman Rubin. The theory of decision procedures for distributions with monotone likelihood ratio. *Annals of Mathematical Statistics*, 27:272–299, 1956.

[68] Ralph L. Keeney. Identifying and structuring values. Decision analysis series report, University of Southern California, Los Angeles, CA, December 1986.

[69] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* John Wiley and Sons, New York, 1976.

[70] Jin H. Kim and Judea Pearl. A computational model for causal and diagnostic reasoning in inference systems. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 190–193, 1983.

[71] David H. Krantz, R. Duncan Luce, Patrick Suppes, et al. *Foundations of Measurement.* Academic Press, New York, 1971.

[72] David M. Kreps and Evan L. Porteus. Temporal von Neumann-Morganstern and induced preferences. *Journal of Economic Theory*, 20:81–109, 1979.

[73] Benjamin Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.

[74] Benjamin Kuipers and Daniel Berleant. Using incomplete quantitative knowledge in qualitative reasoning. In *Proceedings of the National Conference on Artificial Intelligence.* AAAI, 1988.

[75] Benjamin Kuipers and Jerome P. Kassirer. Causal reasoning in medicine: Analysis of a protocol. *Cognitive Science*, 8:363–385, 1984.

[76] Curtis P. Langlotz, Lawrence M. Fagan, Samson W. Tu, et al. A therapy planning architecture that combines decision theory and artificial intelligence techniques. *Computers and Biomedical Research*, 20:279–303, 1987.

[77] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, B50, 1988.

[78] E. L. Lehmann. Some concepts of dependence. *Annals of Mathematical Statistics*, 37:1137–1153, 1966.

[79] John F. Lemmer and Laveen N. Kanal, editors. *Uncertainty in Artificial Intelligence 2*. North-Holland, 1988.

[80] Doug Lenat, Mayank Prakash, and Mary Shepherd. CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI Magazine*, 6(4):65–85, 1986.

[81] Hector J. Levesque. Making believers out of computers. *Aritificial Intelligence*, 30(1):81–108, October 1986.

[82] Vladimir Lifschitz. On the semantics of STRIPS. In Georgeff and Lansky [40], pages 1–9.

[83] W. J. Long, S. Naimi, M. G. Criscitiello, et al. Using a physiological model for prediction of therapy effects in heart disease. In *Proceedings of the Computers in Cardiology Conference*. IEEE, October 1986.

[84] W. J. Long, S. Naimi, M. G. Criscitiello, et al. An aid to physiological reasoning in the management of cardiovascular disease. In *Proceedings of the Computers in Cardiology Conference*, pages 3–6. IEEE, September 1984.

[85] William J. Long and Thomas A. Russ. A control structure for time dependent reasoning. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 230–232, 1983.

[86] David A. McAllester. An outlook on truth maintenance. AIM 551, MIT Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA, 02139, 1980.

[87] David A. McAllester. ONTIC: A knowledge representation system for mathematics. TR 979, MIT Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA, 02139, 1987.

[88] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[89] Drew McDermott. Planning and acting. *Cognitive Science*, 2:71–109, 1978.

[90] Paul R. Milgrom. Good news and bad news: Representation theorems and applications. *Bell Journal of Economics*, 12:380–391, 1981.

[91] Perry L. Miller. *Expert Critiquing Systems: Practice-Based Medical Consultation by Computer*. Springer-Verlag, 1986.

[92] Marvin Minsky. *The Society of Mind.* Simon and Schuster, 1986.

[93] Robert C. Moore. A formal theory of knowledge and action. In Jerry R. Hobbs and Robert C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex, Norwood, NJ, 1985.

[94] J. Mostow and K. Voigt. Explicit integration of goals in heuristic algorithm design. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1090–1096, 1987.

[95] D. S. Nau. Hierarchical abstraction for process planning. In D. Sriram and R. A. Adley, editors, *Knowledge Based Expert Systems in Engineering: Planning and Design*, pages 129–141. Computational Mechanics Publications, 1987.

[96] Bernhard Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34:371–383, 1988.

[97] Allen Newell. The knowledge level. *Artificial Intelligence*, 18(1):87–127, 1982.

[98] Ramesh S. Patil. Causal representation of patient illness for electrolyte and acid-base diagnosis. TR 267, MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA, 02139, October 1981.

[99] Ramesh S. Patil and Oksana Senyk. Efficient structuring of composite causal hypotheses in medical diagnosis. In *Symposium on Computer Applications in Medical Care*, pages 23–29. IEEE, November 1987.

[100] Ramesh S. Patil, Peter Szolovits, and William B. Schwartz. Causal understanding of patient illness in medical diagnosis. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 893–899, 1981.

[101] Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.

[102] Judea Pearl. Embracing causality in default reasoning. *Artificial Intelligence*, 35:259–271, 1988.

[103] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks and Belief.* Morgan Kaufmann, 1988.

[104] Judea Pearl and Thomas Verma. The logic of representing dependencies by directed graphs. In *Proceedings of the National Conference on Artificial Intelligence*, pages 374–379. AAAI, 1987.

[105] Edwin P. D. Pednault. Preliminary report on a theory of plan synthesis. Technical Note 358, SRI Artificial Intelligence Center, August 1985.

[106] Edwin P. D. Pednault. Extending conventional planning techniques to handle actions with context-dependent effects. In *Proceedings of the National Conference on Artificial Intelligence.* AAAI, 1988.

[107] Harry E. Pople, Jr. Heuristic methods for imposing structure on ill-structured problems: The structuring of medical diagnostics. In Peter Szolovits, editor, *Artificial Intelligence in Medicine*, volume 51 of *AAAS Selected Symposium Series*, pages 119–190. Westview Press, Boulder, Colorado, 1982.

[108] John W. Pratt, Howard Raiffa, and Robert Schlaifer. The foundations of decision under uncertainty: An elementary exposition. *Journal of the American Statistical Association*, 59:353–375, 1964.

[109] Howard Raiffa. *Decision Analysis: Introductory Lectures on Choices Under Uncertainty.* Addison-Wesley, Reading, MA, 1968.

[110] Olivier Raiman. Order of magnitude reasoning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 100–104. AAAI, 1986.

[111] Glenn D. Rennels. *A Computational Model of Reasoning from the Clinical Literature*, volume 32 of *Lecture Notes in Medical Informatics.* Springer-Verlag, 1987.

[112] Scott F. Richard. Multivariate risk aversion, utility independence and separable utility functions. *Management Science*, 22:12–21, 1975.

[113] Chuck Rieger and Milt Grinberg. The declarative representation and procedural simulation of causality in physical mechanisms. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 250–256, 1977.

[114] Stanley J. Rosenschein and Leslie Pack Kaelbling. The synthesis of digital machines with provable epistemic properties. In Joseph Y. Halpern, editor, *Theoretical Aspects of Reasoning About Knowledge: Proceedings of the 1986 Conference*, pages 83–98. Morgan Kaufmann, 1986.

[115] Sheldon M. Ross. *Introduction to Stochastic Dynamic Programming.* Academic Press, 1983.

[116] Cynthia J. Rutherford, Byron Davies, Arnold I. Barnett, et al. A computer system for decision analysis in Hodgkins Disease. TR 271, MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA, 02139, 1981.

[117] Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[118] Earl D. Sacerdoti. *A Structure for Plans and Behavior.* American Elsevier, 1977.

[119] Elisha P. Sacks. Hierarchical reasoning about inequalities. In *Proceedings of the National Conference on Artificial Intelligence*, pages 649–654. AAAI, 1987.

[120] Erik Sandewall. Nonmonotonic inference rules for multiple inheritance with exceptions. *Proceedings of the IEEE*, 74(10):1345–1353, 1986.

[121] Leonard J. Savage. *The Foundations of Statistics.* Dover Publications, New York, second edition, 1972.

[122] James G. Schmolze and Thomas A. Lipkis. Classification in the KL-ONE knowledge representation system. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 330–332, 1983.

[123] Ross D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.

[124] Ross D. Shachter. Probabilistic inference and influence diagrams. *Operations Research*, in press.

[125] Ross D. Shachter and David E. Heckerman. Thinking backward for knowledge acquisition. *AI Magazine*, 8(3):55–61, 1987.

[126] Glenn Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, 1976.

[127] Glenn Shafer. Constructive probability. *Synthese*, 48:1–60, 1981.

[128] Glenn Shafer. Savage revisited. *Statistical Science*, 1:463–501, 1986.

[129] Glenn Shafer and Amos Tversky. Languages and designs for probability judgment. *Cognitive Science*, 9:309–339, 1985.

[130] Yoav Shoham. What is the frame problem? In Georgeff and Lansky [40], pages 83–98.

[131] Herbert A. Simon. Spurious correlation: A causal interpretation. *Journal of the American Statistical Association*, 49:467–479, 1954.

[132] Herbert A. Simon. *The Sciences of the Artificial.* MIT Press, second edition, 1981.

[133] Brian Cantwell Smith. Limits of correctness in computers. Technical Report CSLI-85-36, Center for the Study of Language and Information, October 1985.

[134] David J. Spiegelhalter. Probabilistic reasoning in predictive expert systems. In Laveen N. Kanal and John F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 47–67. North-Holland, 1986.

[135] David J. Spiegelhalter and Robin P. Knill-Jones. Statistical and knowledge-based approaches to clinical decision-support systems, with an application in gastroenterology. *Journal of the Royal Statistical Society*, 147:35–77, 1984.

[136] Richard M. Stallman and Gerald J. Sussman. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9(2):135–196, 1977.

[137] Mark Stefik. Planning with constraints (MOLGEN: Part 1). *Artificial Intelligence*, 16:111–140, 1981.

[138] Mark Stefik. Planning and meta-planning (MOLGEN: Part 2). *Artificial Intelligence*, 16:141–170, 1981.

[139] Devika Subramanian and Michael R. Genesereth. The relevance of irrelevance. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 416–422, 1987.

[140] Patrick Suppes. *A Probabilistic Theory of Causality*. North-Holland Publishing Co., Amsterdam, 1970.

[141] William R. Swartout. XPLAIN: A system for creating and explaining expert consulting programs. *Artificial Intelligence*, 21:285–325, 1983.

[142] William Swartout and Robert Neches. The shifting terminological space: An impediment to evolvability. In *Proceedings of the National Conference on Artificial Intelligence*, pages 936–941. AAAI, 1986.

[143] Peter Szolovits, Ramesh S. Patil, and William B. Schwartz. Artificial intelligence in medical diagnosis. *Annals of Internal Medicine*, 108:80–87, 1988.

[144] Peter Szolovits and Stephen G. Pauker. Categorical and probabilistic reasoning in medical diagnosis. *Artificial Intelligence*, 11:115–144, 1978.

[145] Josh Tenenberg. Planning with abstraction. In *Proceedings of the National Conference on Artificial Intelligence*, pages 76–80. AAAI, 1986.

[146] Donald M. Topkis. Minimizing a submodular function on a lattice. *Operations Research*, 26:305–321, 1978.

[147] David S. Touretzky. *The Mathematics of Inheritance Systems.* Morgan Kaufman, Los Altos, CA, 1986.

[148] David S. Touretzky, John F. Horty, and Richmond H. Thomason. A clash of intuitions: The current state of nonmonotonic multiple inheritance systems. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 476–482, 1987.

[149] A. Tversky. On the elicitation of preferences: Descriptive and prescriptive considerations. In David E. Bell, Ralph L. Keeney, and Howard Raiffa, editors, *Conflicting Objectives in Decisions.* John Wiley and Sons, 1977.

[150] Marc B. Vilain. The restricted language architecture of a hybrid representation system. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 547–551, 1985.

[151] Detlof von Winterfeldt. Structuring decision problems for decision analysis. *Acta Psychologica*, 45:71–93, 1980.

[152] Richard Waldinger. Achieving several goals simultaneously. In E. Elcock and D. Michie, editors, *Machine Intelligence 8*, pages 94–136. Edinburgh University Press, 1977.

[153] Sholom M. Weiss, Casimir A. Kulikowski, Saul Amarel, et al. A model-based method for computer-aided medical decision making. *Artificial Intelligence*, 11:145–172, 1978.

[154] Daniel S. Weld. Exaggeration. In *Proceedings of the National Conference on Artificial Intelligence.* AAAI, 1988.

[155] Daniel S. Weld. Theories of comparative analysis. TR 1035, MIT Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA, 02139, May 1988.

[156] Michael Paul Wellman. Reasoning about preference models. TR 340, MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA, 02139, May 1985.

[157] Michael P. Wellman. Representing health outcomes for automated decision formulation. In R. Salamon, B. Blum, and M. Jørgensen, editors, *MEDINFO 86: Proceedings of the Fifth Conference on Medical Informatics*, pages 789–793, Washington, October 1986. North-Holland.

[158] Michael P. Wellman. Dominance and subsumption in constraint-posting planning. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 884–890, 1987.

[159] Michael P. Wellman. Review of Perry L. Miller, *Expert Critiquing Systems. Artificial Intelligence*, 35:273–276, 1988.

[160] Michael P. Wellman, Mark H. Eckman, Craig Fleming, et al. Automated critiquing of medical decision trees. Submitted for publication, 1988.

[161] Michael P. Wellman and David E. Heckerman. The role of calculi in uncertain reasoning. In *Proceedings of the Workshop on Uncertainty in Artificial Intelligence*, pages 321–331, July 1987.

[162] G. A. Whitmore and M. C. Findlay, editors. *Stochastic Dominance: An Approach to Decision Making Under Risk.* D. C. Heath and Company, Lexington, MA, 1978.

[163] Robert Wilensky. Meta-planning: Representing and using knowledge about planning in problem solving and natural language understanding. *Cognitive Science*, 5(3):197–233, 1981.

[164] Brian C. Williams. MINIMA: A symbolic approach to qualitative algebraic reasoning. In *Proceedings of the National Conference on Artificial Intelligence.* AAAI, 1988.

[165] William A. Woods. What's in a link: Foundations for semantic networks. In Bobrow and Collins, editors, *Representation and Understanding*, pages 35–82. Academic Press, New York, 1975.

[166] Ramin Zabih. Dependency-directed backtracking in non-deterministic Scheme. Master's thesis, Massachussetts Institute of Technology, Cambridge, MA, January 1987.

[167] Ramin Zabih, David McAllester, and David Chapman. Non-deterministic Lisp with dependency-directed backtracking. In *Proceedings of the National Conference on Artificial Intelligence*, pages 59–64. AAAI, 1987.

Index