TECHNICAL REPORT

Sibyl: A Qualitative Decision Management System

Jintae Lee

January, 1990

CCSTR#107 SSWP#3143

CENTER FOR
COORDINATION
SCIENCE

Massachusetts Institute of Technology
Sloan School of Management
Cambridge, Massachusetts

Sibyl: A Qualitative Decision Management System

Jintae Lee

January, 1990

CCSTR#107 SSWP#3143

To appear in Winston, P., & Shellard, S. (eds) *Artificial Intelligence at MIT: Expanding Frontiers*, vol.1., Ch 5, The MIT Press, Cambridge, MA, in press.

5

What should you buy? From whom? Each time questions like these are decided, the decision-making effort should become a piece of history that others can benefit from later on. Yet decisions of essentially the same kind are made over and over and the participants in the decision making go over the same tedious ground already plowed by thousands of others.

In this chapter, Lee shows that what we need is a language for describing how decisions are made in terms that make goals and arguments explicit. With such a language, all sorts of questions can be answered that traditional decision science can deal with only obliquely. To what extent, for example, can a past decision serve as a precedent for a current decision? How would a particular decision turn out if cost were not important? What would person X decide to do? What are the risks involved in choice Y? What would be the effect of product Z, if announced?

Thus Lee's language makes a new kind of what-if exercise possible for decision makers. Importantly, Lee's language also makes what-happened exercises vastly easier too. Just keeping track of what happened in big government system-integration contracts, so as to provide an audit trail later on, can consume half of the total cost. Were decisions made in the way envisioned by Lee, the audit trail would be a byproduct of decision making, not an add-on activity.

SIBYL: A Qualitative Decision Management System

Jintae Lee

In this chapter, we describe a Qualitative Decision Management System, called SIBYL.[1] The goal of a Qualitative Decision Management System is to help users represent and manage the qualitative aspects of the decision making process—such as the alternatives being considered, the goals to satisfy, and the arguments evaluating alternatives with respect to the goals. For this purpose, SIBYL provides a language for representing decision making processes and also provides a set of services that help to manage the dependencies represented in the language.

We proceed as follows. We first present a scenario that illustrates the kinds of services that a qualitative decision management system should provide. In the remainder of the chapter, we describe how SIBYL provides these services. First we describe the Decision Representation Language (DRL) that SIBYL uses to represent the qualitative aspects of decision making processes. In the section, "An Example," we show an example decision graph—that is, the representation in DRL of a particular decision making process. In the section, "Services," we discuss the major types of services that we have identified—the management of dependency, plausibility, viewpoints, and precedents. We illustrate these services with the example in the section, "An Example." In the section, "Related Work," we discuss how our approach is related to similar attempts to support qualitative decision making. Finally, we conclude with topics for future research.

[1] A sibyl was one of a number of prophetesses in Greek mythology who gave wise counsel by mediating with the Gods on behalf of human supplicants.

Scenario

Imagine you are trying to decide which knowledge representation language to use for your project called ZEUS. You ask your project members to use a qualitative decision management system to enter thoughts relevant to this decision. These include all the alternatives that should be considered, the constraints an ideal alternative should satisfy, the relevant facts and claims for evaluating the alternatives, and the dependencies that hold among these claims. The system provides a language and a user-interface for using the language, such as object editors, menus, and graphic displays. Also, the system is distributed so that users can examine what has already been entered and then can enter additional opinions incrementally. Suppose users have given their initial inputs; the following scenario illustrates a session with a hypothetical qualitative decision management system. Here, it assumes natural language input for the purpose of exposition, but the interface can be graphical and mouse-based, as it is implemented in SIBYL.

User: Show me the current status.

System: *The system displays the decision matrix shown in figure 1.*

Status of the Decision: Find an Optimal Knowledge Representation Language for Zeus			
Goals / Importance / Alternatives	Support Zeus Requirements H+	Minimize Cost H	Customizability L
STROBE	H	H	H- *
KEE	H-	L	M
LOOPS	H	unresolved	H

Figure 1. An overall evaluation of the alternatives for the decision problem of finding an optimal knowledge representation language for ZEUS.

In the figure, each row represents an alternative, each column represents a desirable property or a goal, and each cell represents the evaluation of the alternative with respect to the goal. Although the evaluation is shown in terms of H (high), M (medium), and L (low), the evaluation measure can be numerical or defined by the user. An evaluation measure marked with a '' means that there is some qualification on the measure.*

User: What is the qualification on the evaluation of STROBE with respect to customizability?

System: The evaluation of STROBE is 'H-' with respect to Customizability, provided that we can get the source code.

When this qualifying assumption turns out to be wrong, the system updates all the objects that depended on this assumption and reevaluates the alternatives.

User: Why is LOOPS' evaluation unresolved with respect to Minimize Cost?

System: The following questions need to be answered for evaluation.

Q1 Which hardware platform should be used for the project?
Q2 Would ENVOS give us a free copy of LOOPS?

User: What are the possible answers to Q1?

System: Sun, Symbolics, or Dandetiger. By the way, there is a decision record associated with this question. See Decision Problem, Hardware Platform, if you want to look at it.

User: What are the current overall evaluations for the hardware alternatives?

The system retrieves the decision record associated with this question and shows the overall evaluation of the alternatives.

System: Sun (very good)
Symbolics (very good)
Dandetiger (poor)

This overall evaluation is computed by merging the evaluation measures, like the ones shown in figure 1, but for this hardware decision problem, properly weighted by the importance of the different goals. The system can use a default merging procedure or one that the user specifies. The user can now examine how the evaluation was arrived at by asking the system to display the decision matrices for this hardware platform decision. But, instead, she goes back to the current decision problem, and wants to make an assumption to see its consequences.

User: Assume that we use Symbolics. How would it affect the evaluations?

System: STROBE (very good)
KEE (good)
LOOPS is not an alternative any more because it is incompatible with the hardware choice.

LOOPS *does not run on Symbolics. This dependency information has been entered earlier, so the system knows that* LOOPS *should* *no longer be an alternative. Other arguments that depend on this new knowledge get updated and the alternatives are reevaluated.*

User: Tell me how you arrived at the evaluation of STROBE with respect to Minimize Cost?

System: *The system displays the matrix shown in figure 2. The matrix shown is like the matrix in figure 1 except that the columns now show the subgoals of the goal, Minimize Cost.*

Status of the Subgoals of: the Goal: Minimize Cost		
Goals / Importance / Alternatives	Minimize Development Cost M	Get One Cheaper L
STROBE	H-	H
KEE	H-	L
LOOPS	M	H

Figure 2. Evaluation of the alternatives with respect to the subgoals of the goal, Minimize Cost.

User: Tell me how you arrived at the evaluation of STROBE with respect to Minimize Development Cost?

System: *The system displays the the argument graph in figure 3. The argument graph shows the arguments evaluating* STROBE *with respect to the goal of Minimizing Development Cost.*

User: Suppose we do not consider the goal, Minimize Cost. How would that change the evaluation?

System: KEE (very good)
STROBE (good)

The user can ask to see the matrix responsible for these evaluations but, instead, the user wants to freeze the current state so that she can come back to it later or compare it to other states that have also been frozen.

User: Create a viewpoint with the current assumptions and call it "Not Considering Development Cost."

System: The current assumptions that have been introduced since last time are:

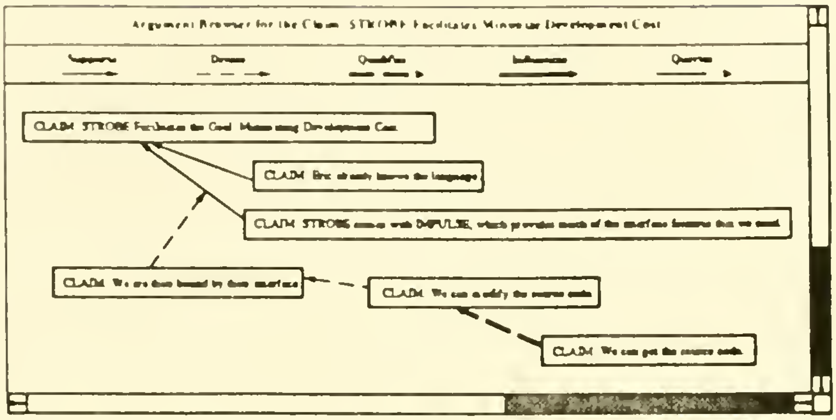

Figure 3. Arguments evaluating STROBE with respect to the goal, Minimize Development Cost.

1 We use Symbolics as the hardware.

2 "Get One Cheaper" is no longer a goal. Do you still want to create the viewpoint?

User: Yes

System: The viewpoint, "Not Considering Development Cost" has been saved.

Having created a viewpoint, the user can add further arguments under this viewpoint. Alternatively, she could switch to another viewpoint, and add arguments in that viewpoint.

The user would like to know if there have been any decisions in the past that would be worth examining.

User: Retrieve useful past decisions.

System: The following are the past decisions that might be relevant in the order of similarity to the current decision problem.

Choose an optimal knowledge representation language for the project XANADU. (Decision 140)
Choose a programming language for the project, ZEUS. (Decision 245)
Choose an operating system for the project, ZEUS. (Decision 247)

After the user examines these past decisions, she decides to transfer some of the relevant knowledge from these earlier decisions.

User: Show me the relevant information from DP 140.

System: *The system displays the folder shown in figure 4. The folder contains all the alternatives, the desirable properties, and the arguments that the system judged to be relevant. The user examines each of these objects in more detail and selects those that she judges to be relevant.*

User: Transfer all the objects that I selected to the current decision problem.

System: Done

User: Show me the current overall evaluation of the alternatives with the new knowledge.

System: STROBE (very good)
NEXPERT (very good)
KEE (good)
KNOWLEDGE CRAFT (unresolved)

The user explores these alternatives and examines relevant arguments in the ways described. She then goes on to add further issues and arguments, saves them, and calls it a day.

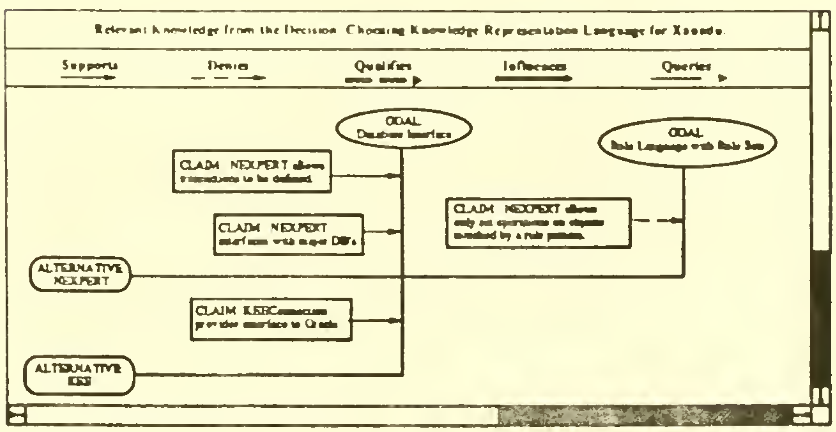

Figure 4. Knowledge judged relevant from a past decision.

The above scenario illustrates the features desirable for a qualitative decision management system. The rest of the chapter shows how they are realized in SIBYL, our implemented system.

Decision Representation Language (DRL)

The services of SIBYL, revolve around decision graphs—the records of the pros and cons evaluating alternatives with respect to the goals. In this section, we briefly describe the language used for constructing these decision graphs, which we call the Decision Representation Language.

Figure 5 shows the objects and relations that form the vocabulary of DRL. Figure 6 presents them graphically. The fundamental objects of DRL are **Alternatives**, **Goals**, and **Claims**. Other objects in DRL are no less essential in decision making, but they are either special cases of the above three, or they are objects useful beyond the context of decision making, as we discuss below.

```
Alternative
Goal
  Decision Problem
Claim
  DRL Relation
    Is-A-Sub-Decision-Of (Dec. Prob., Dec. Prob.)
    Is-A-Subgoal-Of (Goal, Goal)
    Facilitates (Alternative, Goal)
      Is-An-Alternative-For (Alternative, Dec. Prob.)
    Supports (Claim, Claim)
    Denies (Claim, Claim)
    Qualifies (Claim, Claim)
    Queries (Question, Claim)
    Influences (Question, Claim)
    Are-Arguments-For (Group of Claims, Claim)
    Is-An-Answering-Procedure-For (Procedure, Question)
    Is-A-Result-Of (Claim, Procedure)
    Answers (Claim, Question)
    Are-Possible-Answers-To (Group of Claims, Question)
    Is-A-Sub-Procedure-Of (Procedure, Procedure)
    Is-A-Kind-Of (Object, Object)
Question
Procedure
  Procedure Description
  Executable Procedure
Group
Viewpoint
```

Figure 5. The DRL Vocabulary.

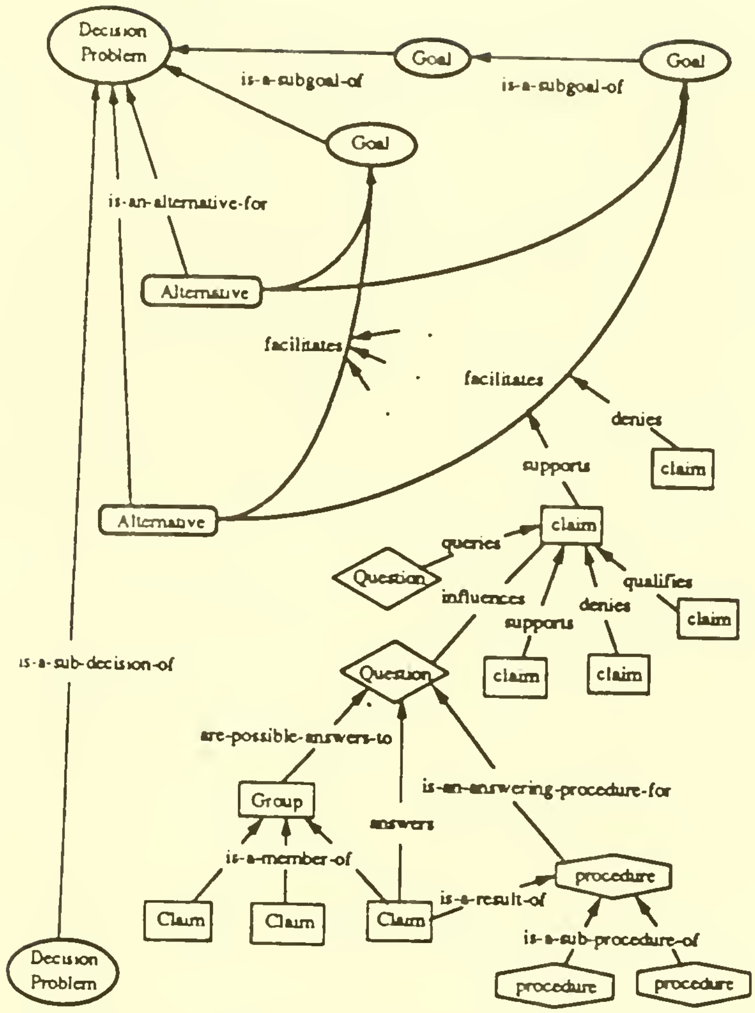

Figure 6. The DRL Model.

Alternatives represent the options to choose from. Goals specify the properties that an ideal option should have. A Goal G1 may be related to another Goal G2 through an Is-A-Subgoal-Of relation, meaning that satisfying G1 facilitates satisfying G2. A special subtype of Goal is Decision Problem, representing the topmost goal of the form "Choose X optimal for Y." Hence, all the goals are subgoals of the decision problem because they elaborate ways of satisfying this top level goal. A Decision Problem DP1 is related to another Decision Problem DP2 through an

Is-A-Sub-Decision-Of relation if DP2 requires solving DP1. For example, choosing the best computer environment for one's project has among its subdecisions choosing the hardware, choosing the operating system, and choosing the programming language. When this is the case, the alternatives of the parent decision consist of combinations of the alternatives from its subdecisions.

Claims are used to represent arguments relevant for choosing among the alternatives. DRL makes no distinction between facts and claims. Any statement is defensible, that is, it can be denied in DRL. A Claim can be related to another Claim through Supports, Denies, or Qualifies relations. A Claim C1 Supports another Claim C2 if the plausibility of C2 becomes higher (or lower) when that of C1 goes up (or down). A Claim C1 Denies another Claim C2 if the plausibility of C2 becomes lower (or higher) when that of C1 goes up (or down). Exactly how the plausibilities get updated is discussed in the section, "Plausibility Management." A Claim C1 Qualifies another Claim C2 if the plausibility of C2 becomes null when that of C1 becomes low enough. As long as the plausibility of C1 remains high, it has no effect on the plausibility of C2. All DRL relations are subtypes of Claim. For example, Supports (C1,C2) is itself a claim that the claim C1 supports the claim C2. Hence, the relation itself can be supported, denied, or qualified—for example, when a person agrees with C1 and C2 but does not agree that C1 *supports C2*. The Is-A-Kind-Of relation is a claim asserting a specialization relation between two objects. For example, Is-A-Kind-Of (A1, A2) holds when A1 is the alternative "Use Sun" and A2 is the alternative "Use Sun 4." All the claims that influence the plausibility of a claim can be grouped and related to it through an Are-Arguments-For relation.

Facilitates (A, G) is the Claim that the alternative A facilitates satisfying the goal G. The plausibility of this Facilitates claim is the measure of how satisfactory the alternative is with respect to the goal in question. Is-An-Alternative-For (A, DP) is the claim that the alternative A represents an option for solving the decision problem, DP, that is, that A facilitates satisfying the top level goal represented by D. Hence, the Is-An-Alternative-For relation is a subtype of the Facilitates relation.

A Question represents an uncertain state which requires more information to determine its outcome uniquely. There are two kinds of Questions. A Question might be a simple request for an explanation—for example, Why do we need an email interface? Alternatively, it may represent a major uncertainty whose different potential outcomes might lead the decision in different ways. The first kind of Question can be linked to any object through a Queries relation. The second kind of Question is linked to a claim through an Influences relation. A Question Q Influences a Claim C if the plausibility of C depends on the answer to Q. A Claim C Answers a Question Q if C represents a possible answer to Q. All the possible answers to a Question can be grouped and related to the Question through an Are-Possible-Answers-To relation.

A Procedure represents either an actual executable procedure (Executable Procedure) or a textual description of a procedure (Procedure Description). A Procedure object can be related to a Question through an Is-An-Answering-Procedure-For if it is believed that the procedure can be used to answer the Question. A Claim C Is-A-Result-Of a Procedure P if C is the information obtained as a result of executing P. A Procedure may be related to other Procedures through an Is-A-Sub-Procedure-Of or an Is-A-kind-of relation. The Is-A-Sub-Procedure-Of relation describes the part/whole relationship among procedures, and it is used when one wants to describe a procedure in terms of the component procedures that implement it. For example, "Get Simulation Software" is a sub-procedure of "Run Simulation." The Is-A-Kind-Of relation mentioned above can be used to specify the specialization relationship among procedures. For example, "Run System Dynamics Simulation" Is-A-Kind-Of "Run Simulation."

A Group represents a set of objects of the same type, among which we want to indicate some relationship. A Group object has the following attributes: Members and Member-Relationship. The Members attribute points to all the objects that belong to the group. The Member Relationship attribute takes as values such relationships as Conjunctive, Disjunctive, Mutually Exclusive, and Exhaustive. A Viewpoint is an object that represents a collection of objects sharing a given constraint. For example, all the alternatives, goals, and the claims that were created under the assumption that the hardware platform is Symbolics form a viewpoint. As such, a Viewpoint is a generalized Group, where the members are not just a set of objects of one type, but objects of different types related to one another through various relations. Viewpoints are discussed in more detail in the section, "Viewpoint Management."

An Example

Figure 7 shows an example decision graph constructed using DRL. This graph is a small portion of the decision graph constructed in the course of an actual decision process using SIBYL, slightly modified for the purpose of presentation. SIBYL has been implemented in Object Lens [Lai *et al.* 1989], which is a general tool for computer supported cooperative work and information management that provides many of the features needed for SIBYL. These include a friendly user interface, a knowledge base interface, hypertext capability, and an email interface. SIBYL has been actually used in several decision making processes—such as choosing an optimal computer environment for different projects, and cooperatively designing a

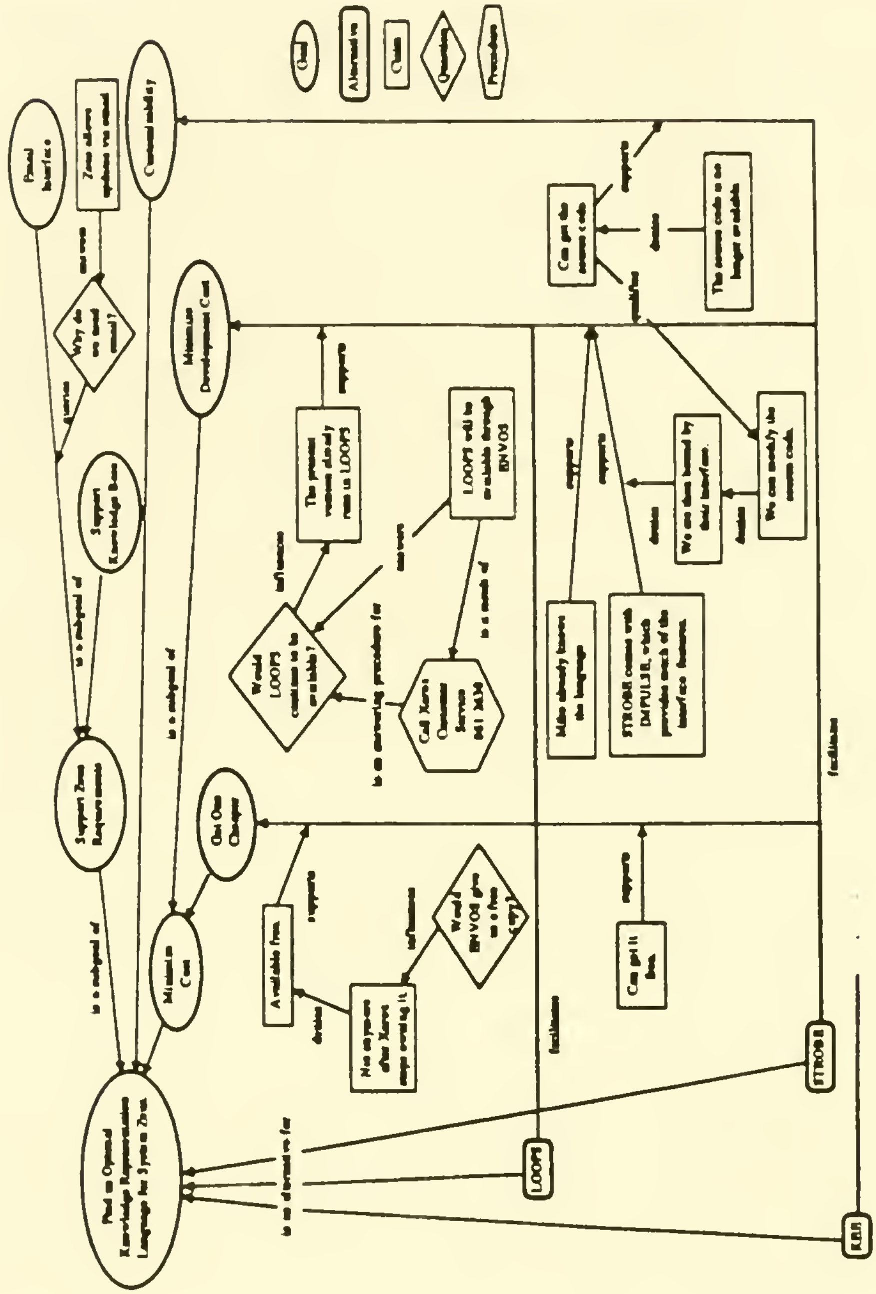

Figure 7. An example decision graph.

floor space. The example in this section will be used later to illustrate the services that SIBYL provides.

The decision problem is that of choosing an optimal knowledge representation language for implementing the system called ZEUS. Three alternatives (LOOPS, STROBE, and KEE) are considered, although all the arguments concerning KEE have been omitted. The desirable properties of the knowledge representation language are represented as goals. The arguments evaluating the alternatives are represented as claims about the **Facilitates** relations between the alternative and the relevant goals. For example, the claim that the present version already runs in LOOPS is an argument in favor of the claim that the alternative LOOPS facilitates satisfying the goal of minimizing development time, represented by the **Facilitates** relation between them. Hence, the plausibility of the **Facilitates** relation represents the evaluation measure of LOOPS with respect to the goal in question.

As noted above, all the relations in DRL are claims. Hence, they can be supported, denied, or qualified. For example, if one accepts the claim, "STROBE comes with IMPULSE, which provides much of the interface functionalities" but one does not agree that this claim supports the claim that "STROBE facilitates minimizing development time," then one denies the **Supports** relation between the claim and the **Facilitates** relation in question (see figure 7).

It is important to emphasize that the decision graph is only a graphical rendering of the knowledge base, and that the user does not see it in the form presented in figure 7. The system selectively displays portions of the decision graph or relevant attributes of selective objects in different formats (for example, a table, a graph, or a matrix) appropriate to different contexts. It also provides a template-based editor for easily creating and linking objects. Hence, the implemented user interface is much more friendly, but it is not the topic of this chapter.

To reduce the complexity, figure 7 omits the group objects that convey the information about the relations among a set of claims or goals; an example of a group is shown in figure 8.

Services

Dependency Management

Dependency Management is responsible for maintaining a consistent state of the knowledge base when a change is introduced. In decision making, objects or their attribute values depend on others. For example, in figure 7, the fact that LOOPS is an alternative depends on which hardware is chosen for implementing ZEUS. If Symbolics is the chosen hardware, LOOPS is no longer an alternative because it does not run on Symbolics. We should be able to represent such dependency as well as representing what should be done when we acquire the information.

Figure 8a shows how such a dependency is represented in DRL. Any uncertainty involved in a dependency is represented as a **Question**. Hence, we create a **Question** object, "Which hardware is chosen for implementing ZEUS," and link it to the **Is-An-Alternative-For** relation through an **Influences** relation. The possible outcomes of the **Question** are claims of the form, We use X, which are linked to the **Question** through the **Answers** relation. Alternatively, if we wanted to specify a relationship among these outcomes (for example, disjunctive or mutual exclusive relations), we would group these claims via a Group object as shown in figure 8b.

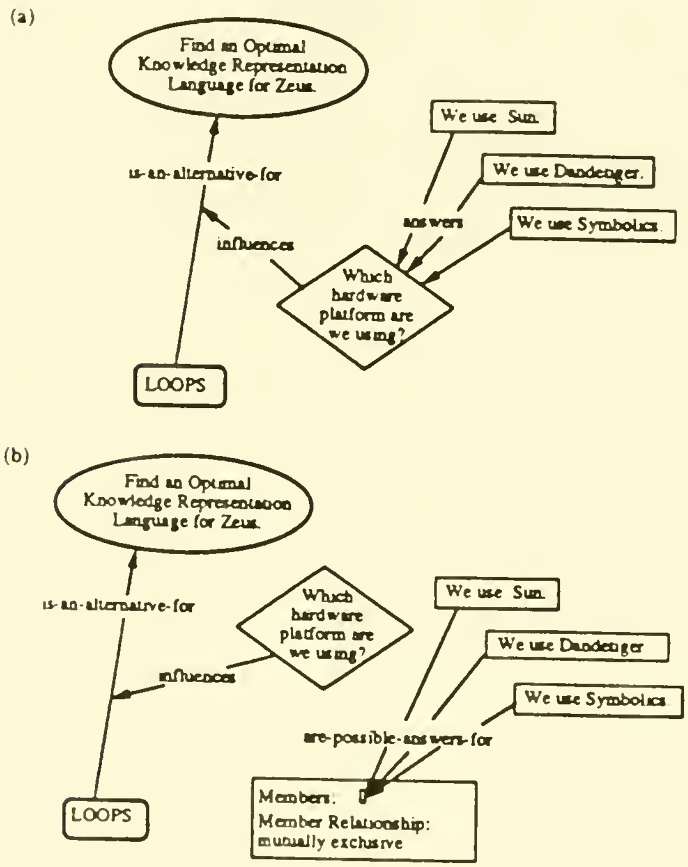

Figure 8. A representation of dependency in DRL.

Each of these claims has an attribute called UpdateProcedure, which provides the following information: a predicate on the plausibility of the claim, and the action to perform when the predicate is true. The predicate can be, for example, (> 30) if the plausibility measure used is numerical, or (> Very-High) if the measure is categorical. The system provides a set of standard plausibility measures and the appropriate predicates, but the user is free to invent her own. A typical action that one can specify is that of updating the plausibility of the influenced claim. Thus, we can specify the following pair, ((> Very-High), Set-Plausibility(0)) as the value of the UpdateProcedure attribute of the claim, "We use Symbolics" so that when Symbolics becomes the most likely hardware of choice, LOOPS would no longer be considered as an alternative. Figure 9 shows the updated decision graph. The alternative, LOOPS, and the relevant claims, do not get deleted, however; they only become invisible and have no effect on the decision. They become visible again if their importance or plausibility becomes non-zero. Other kinds of actions that can be performed include creating new objects and linking them to existing objects via specified relations. We plan to provide a high level language for specifying such actions. With such a language, the action can be an arbitrary procedure written in the language.

Other relations among claims, such as Qualifies, Supports, Denies, are special cases of Influences, and represent more specialized kinds of dependency. The action of a Qualifies relation is to set the plausibility of the qualified claim to zero when the plausibility of the qualifying claim becomes low. That is, the UpdateProcedure of the qualified claim is of the form, ((< Threshold), Set-Plausibility(0)), where the value of Threshold is set either globally or locally. Above the threshold, the qualifying claim has no effect on the plausibility of the claim qualified. For example, in figure 10, the claim, "Can get the source code" had no effect on the claim it qualifies. However, when it is denied by the claim, "The source code is no longer available" and if its plausibility becomes low enough, the plausibility of the claim it qualifies, "We can modify the source code," is set to zero. A consequence is that this change propagates so as to make the evaluation of STROBE with respect to the goal of Minimize Development Cost less strong. The action of a Supports relation is to raise (or lower) the plausibility of the supported claim when the plausibility of the supporting claim goes up (or down). The action of a Denies relation is to lower (or raise) the plausibility of the denied claim when the plausibility of the denying claim goes up (or down).

Plausibility Management

Plausibility Management is a special case of Dependency Management. It is responsible for maintaining consistency among the plausibilities of related

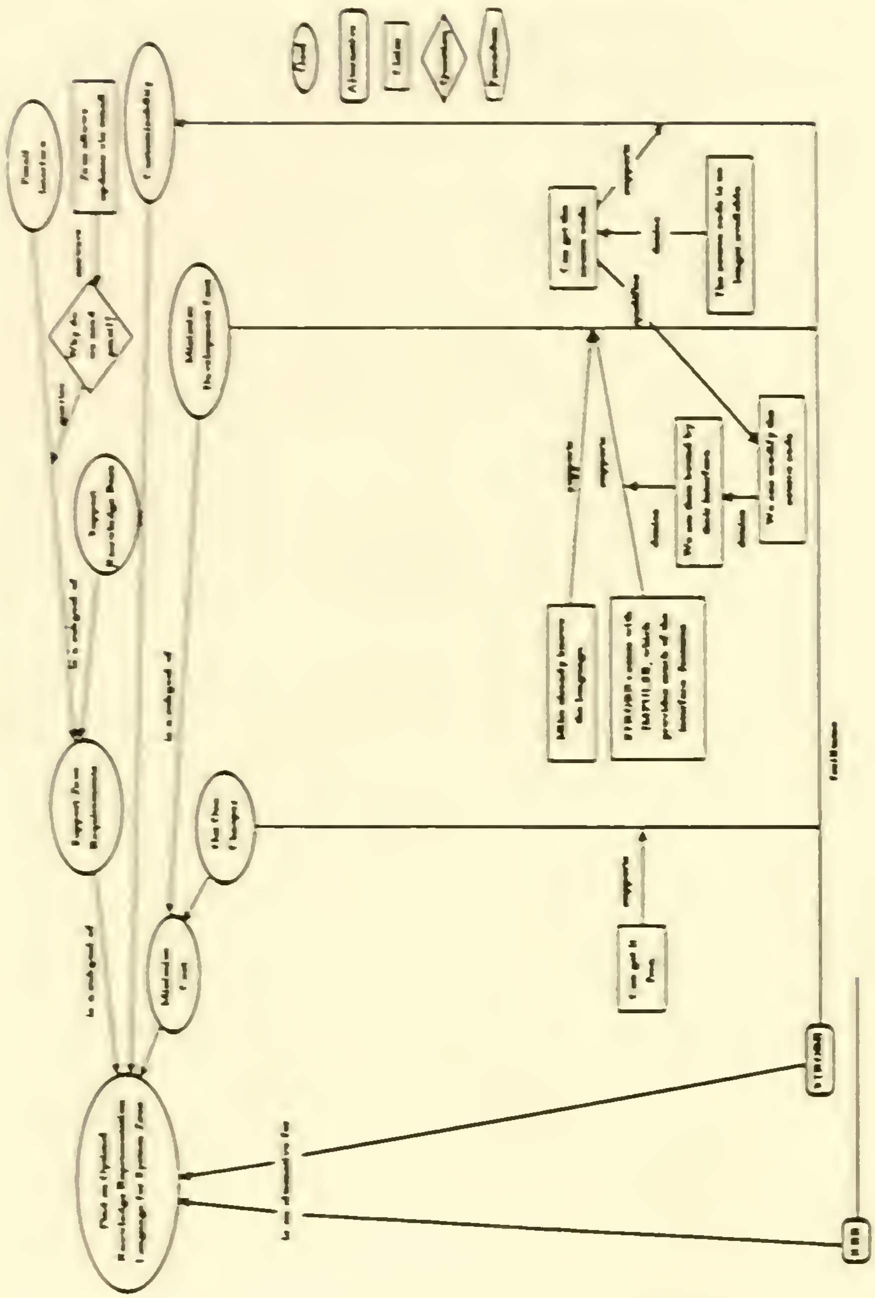

Figure 9. The example decision graph with Symbolics chosen as the hardware platform.

claims. Suppose, in figure 7, we want to know how good STROBE is overall, given the arguments produced so far. In DRL, the overall evaluation of an alternative is the plausibility of the `Is-An-Alternative-For` relation between the alternative and the top level goal (that is, the tension problem). This plausibility is in turn a function of the plausibility of the `Facilitates` relations between the alternative and the associated subgoals as well as a function of the importance of these goals. The plausibility of a claim, in particular a `Facilitates` relation, is a function of the plausibility of the claims that support, deny, or qualify it, as well as the claims answering a question influencing the claim. Hence, to see how good STROBE is as an alternative, we need to propagate plausibilities across the links and merge them into one. This merging should not only depend on the component plausibilities but also on the type of link that they have been propagated across. Plausibility Management is responsible for such propagation and merging of the plausibilities.

Going back to our example, evaluating STROBE overall requires, among other things, knowing how good STROBE is with respect to the goal of Minimize Development Cost—that is, the plausibility of the `Facilitates` relation between the alternative, STROBE, and the goal, Minimize Development Cost. This knowledge in turn requires knowing the plausibility of the claims supporting, refuting, and qualifying the `Facilitates` claim. Each claim has *a priori* plausibility initially given by its author, reflecting the measure of how confident she is about the claim. The plausibility of a claim is a function of this *a priori* plausibility and the plausibilities of the claims that are related to it via one of the relations such as `Supports`, `Denies`, and `Qualifies`. When we view a claim as a hypothesis and the supporting or denying claims as pieces of evidence, the problem of computing the plausibility of a claim is the classical problem in belief management. And there are many existing theories of confirmation, such as Bayesian or Dempster-Shafer's which have their own requirements, such as independence or exhaustiveness assumptions.

Rather than being committed to a particular theory or proposing a new confirmation theory, SIBYL provides an interface for any confirmation theory and lets the user choose one that suits her best. Such an interface requires the following information: the merging procedure, the arguments the procedure takes, and the information about the way that the pieces of evidence are related to one another. The system provides some of the well-known procedures such as Bayes or Dempster-Shafer so that the user can choose one of them. However, she can also write her own procedure. Furthermore, because SIBYL allows the user to specify a merging procedure for a claim, she can specify one procedure as the default globally, yet locally override this procedure by specifying a different one for a given claim or a type of claim.

There are two kinds of knowledge that a confirmation procedure needs at each step. One is the plausibility values of the component claims that are being merged. In the case of Dempster-Shafer, each piece of evidence (a claim C) needs to supply an information pair, (Belief(C), Plausibility(C)). In one version of the Bayesian procedure [Duda *et al.* 1979], each claim needs to supply $Pr(C)$, and each relational claim that links two claims, C1 and C2, needs to supply a pair $(Pr(C2|C1), Pr(C2|\neg C1))$. The parameters that the chosen procedure needs as input is to be found in the Plausibility attribute of the claims involved. Thus, if the user is supplying her own procedure, then she needs to ensure that the Plausibility attribute of claims supply appropriate arguments for the procedure. The other kind of information that a procedure needs is how the component claims are related—for example, whether they are independent, mutually exclusive, and/or conjunctive. As discussed above, this knowledge is contained in the Member Relationship attribute of the group object which groups the claims related to a given claim.

Once we compute the plausibility of all the `Facilitates` relations between STROBE and all of the goals, we need to propagate them across the `Is-A-Subgoal-Of` relations and merge them properly to produce the overall evaluation of STROBE. Given that G2 and G3 are subgoals of G1, that we have the evaluation measures of STROBE with respect to G2 and G3, and that we have the importance measures for G2 and G3, how should they be reflected in the evaluation of STROBE with respect to G1? For example, what is the evaluation of STROBE with respect to "Minimize Cost" when we have computed its evaluations with respect to the subgoals, "Get One Cheaper" and "Minimize Development Cost"?

The answer to the above question requires a careful analysis of the relations that can hold among goals. The `Is-A-Subgoal-Of` relation is not sufficient to capture these relations. For example, subgoals can be independent or have tradeoffs; they may exhaust their parent goal; or they can be mutually exclusive or overlapping. Also, subgoals may interact in various ways, as planning research has discovered [Sussman 1975; Sacerdoti 1977; Chapman 1985]. How the evaluation measures propagate and merge across the goals will depend on the kinds of relations that hold among them. We are still working out the taxonomy of these relations, based on Quinlan's work [1983]. Depending on what kind of relation holds among the subgoals, we may be able to use a simple algorithm such as the weighted average of the plausibilities by the importance. Again, rather than being committed to a particular algorithm we provide an interface for a merging procedure so that the user has control over how the merging should be done.

Another issue that we have not yet resolved satisfactorily is how to represent and reflect the degree of consensus. If more users agree with a claim, the plausibility of that claim should go up. On the other hand, the mere number of users should not be an indication of the plausibility because, for instance, users might not endorse a claim even when they agree with it, judging it to be obvious. Hence, the number of users who happen to endorse a claim explicitly is somewhat accidental. In the present version of SIBYL, when a user makes a claim, she is associated with the claim as its creator. Then, if users want to express approval of the claim, they do so by associating themselves with the claim as its co-creators. This scheme allows the decision maker to assign plausibility to the claim by considering not only the number but also other factors such as the credibility and the expertise of the users associated with the claim. We are not completely satisfied with this scheme, and we would like to find one that places less burden on the decision maker. We plan to find a better measure of consensus by studying other work, for example, Lowe [1986].

Viewpoint Management

Viewpoint Management is responsible for creating, storing, retrieving, mapping, and merging Viewpoints. A Viewpoint is an object that represents a collection of objects that share certain assumptions. Multiple Viewpoints on a decision record represent multiple perspectives on the given decision problem. Figure 10 shows a viewpoint which evaluates alternatives without considering a particular goal. The importance of the goal of getting a cheaper language (and its subgoals if there were any) has been set to null so that it no longer shows; nor do the the claims that were relevant to these goals.

There are at least five types of cases where we want to create multiple Viewpoints in SIBYL. We need viewpoints with:

- Different assignments of importance to goals.
- Different assignments of plausibilities to claims.
- Different subset of objects such as goals and alternatives.
- Different, hypothetical, answers to a question.
- Objects at different time points.

The first two cases are obvious. They are needed when we want to see what happens if we place different weights on goals or claims. A typical example of the third case is found when one wants to consider only goals or claims whose importance or plausibility measure is beyond certain threshold. Another example, mentioned earlier, is the one where all the goals of minimizing cost have been deactivated. In the fourth case, a viewpoint corresponds to a state in a truth-maintenance system. When faced with uncertainty, one makes different assumptions and pursues their consequences from different viewpoints. In the fifth case, Viewpoints provide a version mechanism. One should be able to freeze the states into viewpoints at different times so that one can see the way that the decision making unfolded at a later time, or even revert back to a previous viewpoint if necessary.

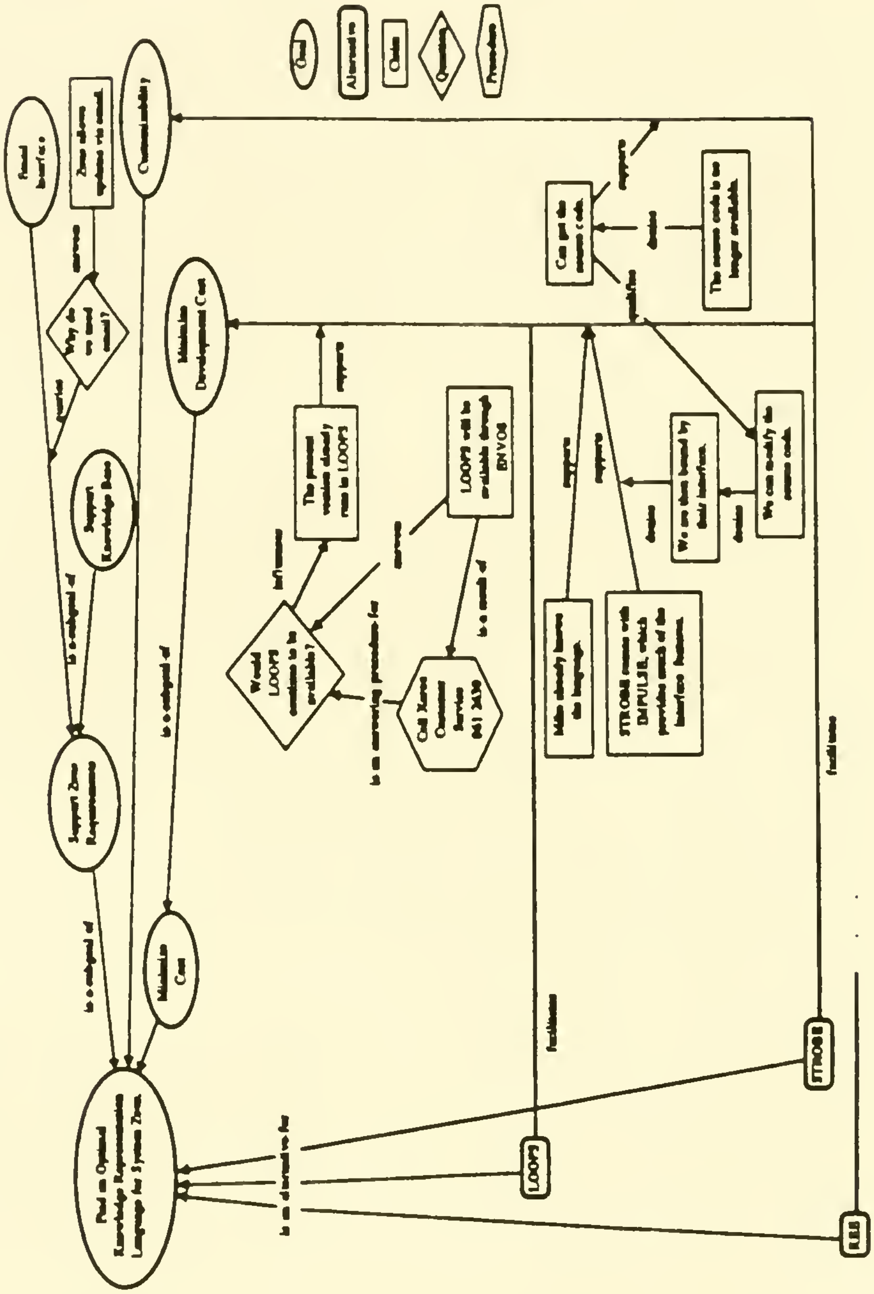

Figure 10. A viewpoint not considering the goal, "Get One Cheaper."

Viewpoints are first-class objects in DRL. As such, they can appear as alternatives in a meta-decision problem such as whether we should stop exploring additional alternatives. Also, Viewpoints can be related to one another in more ways than chronologically. The following relations have not yet made their way into DRL, but they will in the future: `Is-A-Next-Version-Of`, `Elaborates`, `Restricts`, `Has-Different-Importance`, `Has-Different-Plausibilities`.

It is also important to be able to establish mappings between objects in different viewpoints. Such mappings would show how the problem representation has evolved in the course of a decision making process. For example, a claim may differentiate into more specific claims; or several alternatives may be merged into one if their differences prove unimportant for a given purpose. We would represent this evolution as mapping between viewpoints representing different time slices. It is also useful to be able to merge viewpoints. For example, if the marketing group and the design group were both evaluating alternatives for a computer environment, it is natural that they make their evaluations within their own viewpoints and then later merge these viewpoints for an overall evaluation. We are still working out this aspect of Viewpoint Management. Relevant literature includes: studies on view and schema integrations in database research [Battini *et al.* 1986; Lee & Malone 1988], work on versions [Katz *et al.* 1984; Goldstein & Bobrow 1981; Bobrow *et al.* 1987], and work on viewpoints [Attardi 1981; Barber 1982; Kornfeld 1982]. We would like to incorporate some of these ideas in our next version of Viewpoint Management.

Precedent Management

Precedent Management is responsible for indexing past decisions and retrieving ones that are useful for the current decision problem. Once they are retrieved, the precedent manager extracts from them the pieces of the knowledge that are relevant for the present problem and places them in the present decision graph.

SIBYL uses goals to index past decisions. Two decision problems are judged to be similar and potentially useful to the extent that they share goals. Using goals as the index also allows the system to determine which parts of the retrieved decision graph are actually relevant. It is those objects—the claims and the alternatives—that are linked to the shared goals and their subgoals. We can, so to speak, lift out the shared goals and the subgoals and take with them all the objects that are linked to them. We place this structure in the current decision graph by overlaying the shared goals. Figure 11 shows a new decision problem, to which the shared goals have been overlaid. Using precedents this way is also useful because

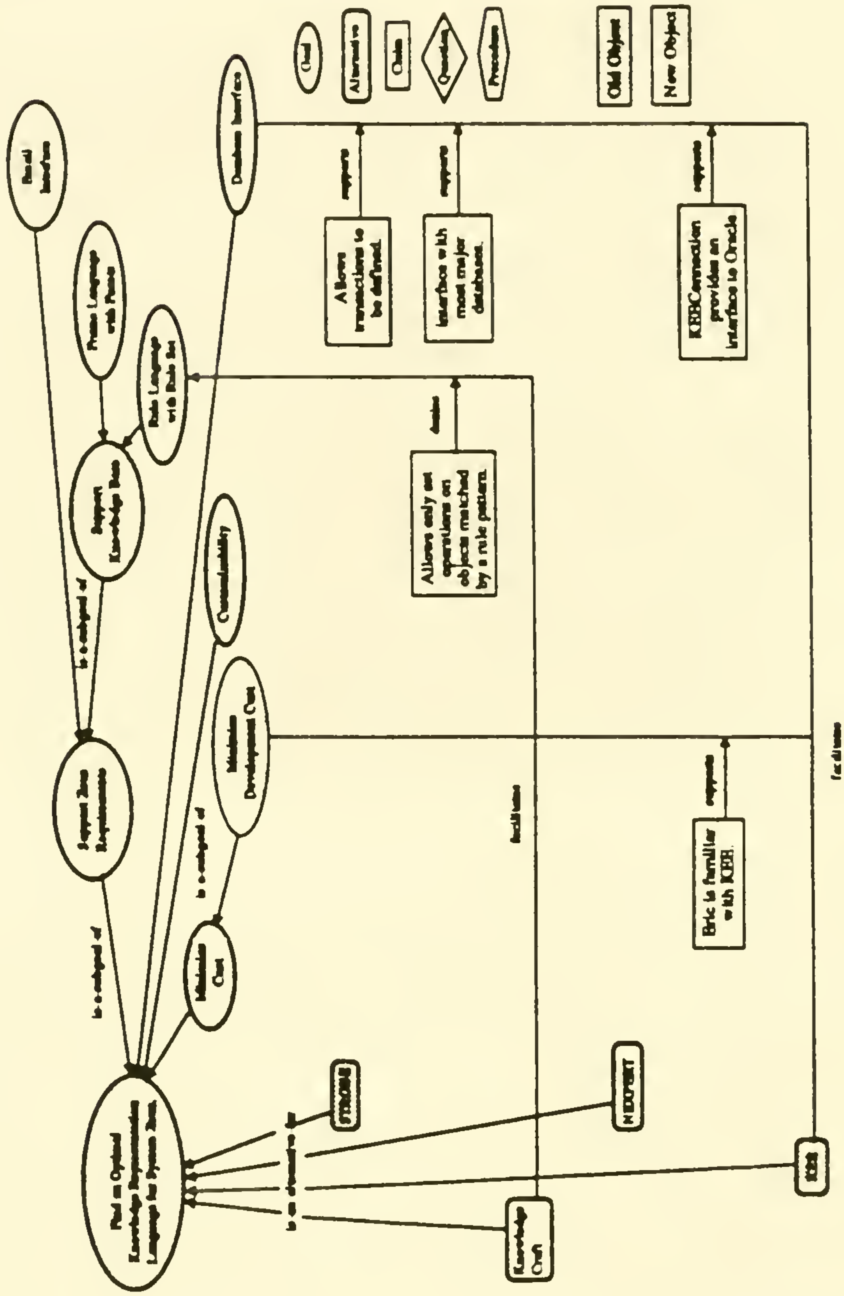

Figure 11. The example decision graph augmented by knowledge from a past decision.

it allows the new decision maker to become aware of other alternatives or of different ways of achieving a given goal.

The following problems are still to be solved. First, some of the goals and claims that have been transferred in this way would not be relevant after all, due to context dependency. What is represented in a decision graph usually leaves out many things that were assumed in that context. For example, the claim, "Eric already knows the language" was a supporting claim for the claim that KEE facilitates minimize development time, because in that context Eric was the person who was going to develop the system. This fact is probably not true any more, especially if it is a different group that is retrieving this decision graph. The present solution is to let users filter out irrelevant facts from the ones that the system suggests. We will, undoubtedly, try to improve the algorithm for determining the relevance so that the user has less to filter out. However, we do not think that such an algorithm can entirely eliminate the need for the user to check the result without, at the same time, requiring her to supply details that would otherwise be unnecessary.

Another problem is to determine which goals are shared across decision problems. Certainly, matching names would not be appropriate as names can be arbitrary strings. A solution is to provide a taxonomy of goals, from which users can create instances. Then, we would have a basis for judging the similarity among goals. We have partially developed such a taxonomy, but more work needs to be done. Yet another problem is that of credit assignment. One would not want to make a decision based on the knowledge used for past decisions that turned out to be disasters. Nevertheless, some pieces of the knowledge in such decisions may still be useful. We would like to associate the failure or the success of a decision with the pieces of knowledge responsible for the result. The present solution in SIBYL is to represent this information in the following attributes of decision problem object: Outcome and Responsible-Objects. This way, SIBYL at least allows users to represent and examine the needed information. However, at the moment, these attributes have no computational significance. We want to explore ways to make these attributes computationally useful. Obviously, there is room here for applying the ideas from learning and case-based reasoning research [Michalski *et al.* 1986; Kolodner 1988; Winston 1986]. It would be a challenge to incorporate some of the ideas from this research into Precedent Management.

Related Work

Studies on decision making abound. There are quantitative decision theories, psychological studies on human decision making, organizational theories, political decision making, decision support systems, and studies on qualitative decision making. Many of these studies are relevant to the work described in this chapter. For example, a qualitative decision management system like SIBYL can be viewed as complementing classical maximum expected utility theory. In the classical decision theory, only the alternatives and possible consequences are represented explicitly; goals are merged into the utility function and the arguments disappear into the probability distribution over the consequences. In this sense, a qualitative decision management system provides a means for retrieving and, when necessary, changing the intuitions behind the utility and probability assignments. However, it should be clear that a qualitative decision management system and the classical decision theory are quite different in both their goals and structures.

In the rest of this section, we discuss the relation of SIBYL to various computational studies on qualitative decision making because they share the goal of a qualitative decision management system; namely the representation and management of the qualitative aspects of decision making processes. This category includes work such as Toulmin's [1969] theory of arguments, gIBIS [Conklin & Begeman 1988], and Doyle's [1980] model of deliberation. We include Toulmin's work, though it is not computational, because his work has been adopted widely as a model of argument in many computational studies [Birnbaum *et al.* 1980; Lowe 1986] as well as in other fields. Below, we give a brief comparison of these studies to SIBYL.

A British philosopher, Stephen Toulmin, proposed a model of argument in 1969. Figure 12 shows Toulmin's model and an example. A *Claim* is the main assertion being made; a *Datum* supports the claim; a *Warrant* is the basis on which the Datum is said to support the claim, a *Backing*, in turn, supports the Warrant; a *Qualifier* qualifies the extent to which the Datum supports the claim, and a *Rebuttal* gives a reason for the Qualifier.

We decided not to use Toulmin's model for representing arguments because Toulmin's object types do not support the uniformity and extendibility of representation. For example, Toulmin's model allows a Claim to be backed up, qualified, or rebutted but not a Datum or a Warrant. It is not clear what to do if one wants to argue about the validity of a Datum or disagrees with a Warrant. Also, one can deny an existing claim only in a round-about way by supporting its negation. One can qualify a claim, but it is not clear whether that qualification is due to some uncertainty over situations, less than perfect plausibility of data, or the weak relation between data and claim.

gIBIS (graphical Issue Based Information System) is a "hypertext tool for exploratory policy discussion" that is being developed by Conklin and Begeman [1988]. The goal of gIBIS is to capture the design rationale: the design problems, alternative resolutions, tradeoff analysis among these alternatives, and the tentative and firm commitments that were made during

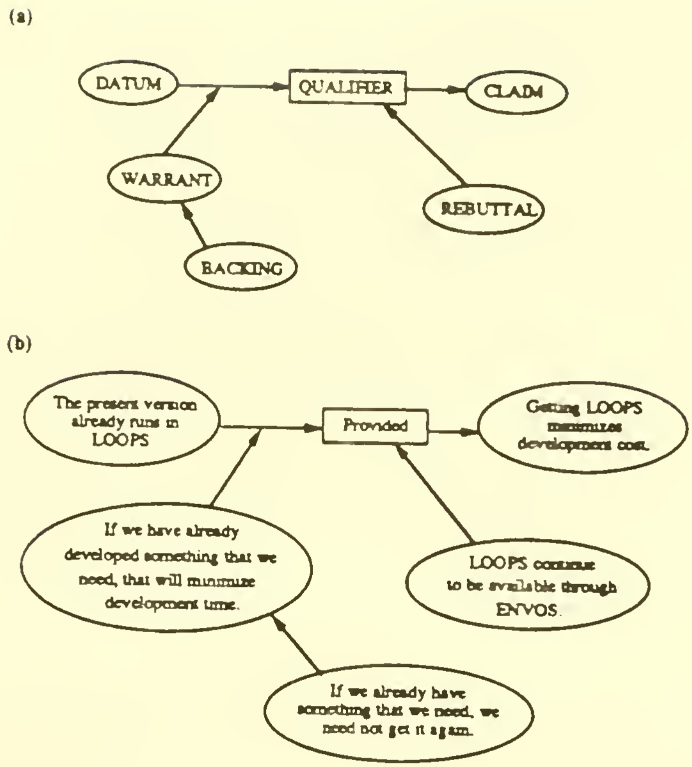

Figure 12. (a) Toulmin's model of argument and (b) an example.

the decision making process. gIBIS is similar to SIBYL in its goal and also in that it is a system designed to *support* human decision making.

The difference between SIBYL and gIBIS comes from the structures provided in achieving this goal and the types of services they provide using these structures . Figure 13 shows the ontology of gIBIS. *Issue* corresponds to Decision Problem in DRL, *Position* corresponds to Alternative, and *Argument* corresponds to Claim. Notably lacking in gIBIS is the explicit representation of Goal. In Lee [1989a], we point out the importance of making goals explicit: they allow modular representation of arguments; they force users to articulate evaluation criteria; they let users argue about them, and they provide a basis for precedent management as well as multiple viewpoints. Another general difference between gIBIS and SIBYL is that gIBIS is mainly a hypertext system whose services focus on the *presentation* of objects in such a way that it is easy to enter arguments and easy to see the structure of what has been represented. On the other hand, SIBYL is

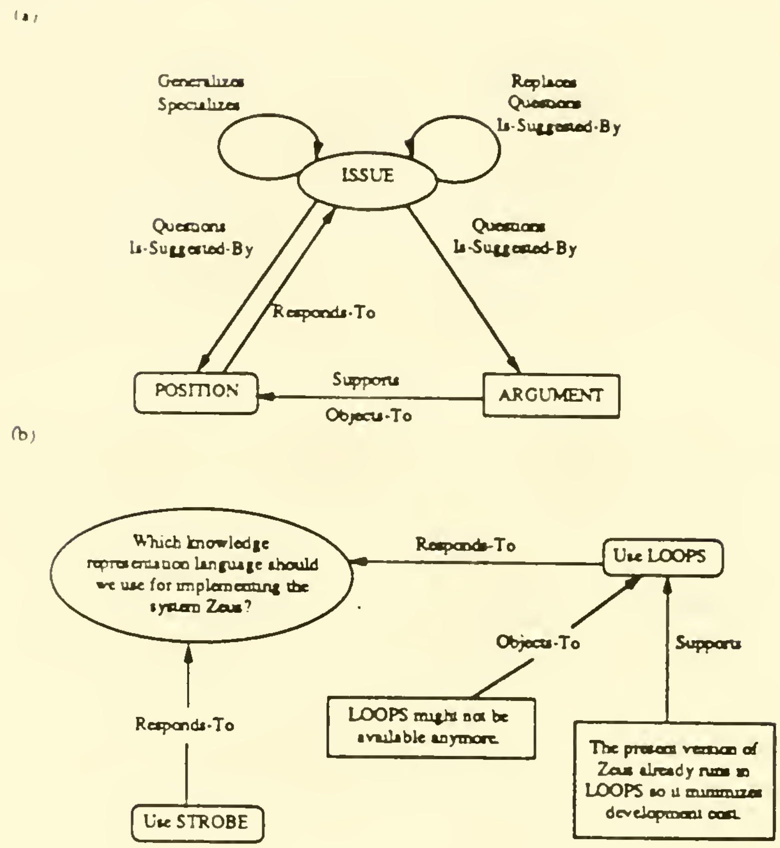

Figure 13. (a) The gIBIS model of argument and (b) an example.

a knowledge-based system whose services focus on the *management of the dependency* among the objects.

Doyle [1980] proposed "a model for deliberation, action, and introspection." It is worth relating the model underlying SIBYL to his model because his is the most comprehensive and detailed model of defeasible reasoning that we have seen so far, that is, where the assertions are non-monotonic. As Doyle points out, this provides an important framework for dialectical reasoning, that is, reasoning based on arguments, Doyle's theory is very comprehensive and we could not do it justice here. However, Doyle's work is described in Lee [1989a]. Here, we note that the goal of Doyle's model is to control or direct the reasoning and actions of a *computer program* by explicitly representing and justifying the reasoning process itself. SIBYL has a less ambitious goal than Doyle's. SIBYL provides a model for decision making sufficient to support *users* making decisions. Furthermore, the scope of SIBYL is restricted to the kind of decision making which involves evaluating well-defined alternatives against a set of goals. That restriction in scope lets SIBYL provide task-specific constructs such as the Facilitates relations, which allows more modular representation of the arguments evaluating alternatives, as well as allowing one to argue about such relations.

We would like to view SIBYL as a system that is compatible with Doyle's model but is non-committal about some of its aspects. Thus, the model underlying DRL is not as explicit as Doyle's, however, it does not have to do things like making assertions, the way Doyle prescribed. What SIBYL requires is that claims are produced and linked to other claims through the relations that DRL provides. The claims could have been produced in a manner consistent with Doyle's model or otherwise. Also, one may decide, for computational tractability, not to keep justifications for every step in the reasoning process. In that sense, SIBYL can be viewed as a higher level interface to something like Doyle's model, but not bound to it.

Future Work

We mentioned above many of the specific problems that we are still tackling. Here, we discuss more general topics of future research.

Earlier we discussed four major types of services—dependency, plausibility, viewpoint, and precedent management. As we apply SIBYL to more complex decision making processes, we expect to identify more types of services that prove useful. For example, the importance of risk management has been pointed out by Boehm [1988]. Risk management would be responsible for identifying the significant sources of risk, evaluating the risk associated with each alternative, and helping users to allocate the resources based on the evaluation. Given that risk is the uncertainty of achieving important goals and that uncertainty is represented in DRL by an Influences relation, SIBYL can help identify risks by finding, for example, the important goals whose associated Facilitates relation is influenced by a Question, directly or indirectly. The technique used to resolve the risk can also be represented as a Procedure and linked to the Question through Is-An-Answering-Procedure-For so that the knowledge can be used by others later. We plan to explore this and further potential types of services that are useful in complex decision making.

We also plan to study the process of qualitative decision management as a computational problem solving paradigm based on arguments. So far, we have discussed SIBYL in the context of providing support for human decision making. However, the importance of human-like decision analysis—

in the sense of involving arguments, evaluating alternatives, compromising and negotiating—is also important in automated reasoning. Hewitt [1986], for example, pointed out the features of open systems, such as incompleteness and inconsistency, which makes a purely deductive approach unrealistic. What happens in real systems and what needs to be included in artificial systems is the complexity of the dialectical processes involving the interactions among agents with different goals, shared resources, and different expertise. A Qualitative Decision Management System is an attempt to identify and articulate the objects and the processes involved in such decision making processes. As such, we feel that there is no reason why the same language and the same services could not be used in making decisions by computational agents, provided that the attribute values of the objects are themselves computational objects. Doing this requires much more work, including a careful analysis of the relation between SIBYL and those in other systems such as Doyle's.

References

Attardi, G., and M. Simi [1981], "Semantics of Inheritance and Attributions in the Description System Omega," *Proceedings of IJCAI 81*, Vancouver, BC, Canada.

Barber, G. [1982], "Office Semantics," Ph.D. Thesis, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Battini, C., M. Lenzerini, and S. B. Navathe [1986], "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM Computing Surveys* vol. 18, no. 4, pp. 323-364.

Birnbaum, L., M. Flowers, R. McGuire [1980], "Towards an AI Model of Argumentation," in *Proceedings of AAAI*, pp. 313-315, Stanford, CA.

Bobrow, D., D. Fogelsong, and M. Miller [1987], "Definition Groups: Making Sources into First-Class Objects," in [Wegner & Shriver 1987] pp. [illegible]-146.

---

†An extended version of this chapter can be found in Lee [1989a].

‡I would like to thank Patrick Winston, Rick Lathrop, Jonathan Amsterdam, Gary Borchardt, and others in the learning group at the Artificial Intelligence Laboratory for keeping my enthusiasm alive as well as many useful comments. The author is also affiliated with the Center for Coordination Science at MIT Sloan School of Management. I appreciate the comments and the encouragements from those at the Center—especially Thomas Malone, Kevin Crowston, and most of all Kum-Yew Lai. Many others have contributed in various ways to the work reported here: Franklyn Turbak, Paul Resnick, John Mallery, Carl Hewitt, Marvin Minsky, David McAllester, and Jon Doyle from the Artificial Intelligence Laboratory; Frank Manola, Mike Brodie, and the members of the Intelligent Database Management Systems group at GTE Labs. I thank them all.

Boehm, B. W. [1988], "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, pp. 61-72.

Brodie, M., and J. Mylopoulos [1986], *On Knowledge Base Management Systems*, Springer-Verlag, New York.

Chapman, D. [1985], "Planning for Conjunctive Goals," Report AI-TR-802, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Conklin, J., and M. L. Begeman [1988], "gIBIS: A Hypertext Tool for Exploratory Policy Discussion," *Proceedings of Computer Supported Cooperative Work*, Portland, Oregon.

Doyle, J. [1980], *A Model for Deliberation, Action, and Introspection*, Report AI-TR-581, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Duda, R. O., P. E. Hart, and N. Nilsson [1976], "Subjective Bayesian Methods for Rule-based Inference System," in [Webber & Nilsson 1981], pp. 192-199.

Goldstein, I., and D. Bobrow [1981], "Layered Networks as a Tool for Software Development," *Proceedings of 7th IJCAI*, University of British Columbia, Vancouver, BC, Canada, pp. 913-919.

Hewitt, C. [1986], "Offices Are Open Systems," *Proceedings of ACM Conference on Office Information Systems*, Providence, RI.

Katz, R. H., M. Anwarrudin, and E. Chang [1986], "Organizaing a Design Data/-base Across Time," in [Brodie & Mylopoulos 1986].

Kolodner, J. [1988], (Editor) *Proceedings of a Workship on Case-Based Reasoning*, Clearwater Beach, FL.

Kornfeld, B. [1982], "Concepts in Parallel Problem Solving," Ph.D. Thesis Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Lai, K., T. Malone, and K. C. Yu [1989], "Object Lens: A Spreadsheet for Cooperative Work," *ACM Transaction on Office Information Systems*, vol. 6, pp. 332-353.

Lee, J. [1989a], "A Qualitative Decision Management System: An Overview," Report AIM-1191, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA (to appear).

Lee, J. [1989b], "Task-Embedded Knowledge Acquisition through a Task-Specific Language," *Proceedings of IJCAI Knowledge Acquisition Workshop*, Detroit, MI.

Lee, J., and T. Malone [1988], "How Can Groups Communicate When They Use Different Languages?" *Proceedings of ACM Conference on Office Information Systems*, Palo Alto, CA.

Lowe, D. [1986], "SYNVIEW: The Design of a System for Cooperative Structuring of Information," *Proceedings of Computer Supported Cooperative Work*, Austin, TX.

Malone, T., K. C. Yu, and J. Lee [1989], "What Good Are Semi-Structured Objects?" Sloan Working Paper, 3064-89-MS, Massachusetts Institute of Technology, Cambridge, MA.

Michalski, R. S., J. G. Carbonell, and T. Mitchell (editors) [1986], *Machine Learning* vol 2, Morgan Kauffman, Los Altos, CA.

Quinlan, J. R. [1983], "Inferno: A Cautious Appropach to Uncertain Inference," *The Computer Journal*, vol. 26, no. 3, pp. 255-267.

Sacerdoti, E. [1975], *A Structure for Plans and Behavior*, American Elsevier, New York, NY.

Shafer, G. [1976], *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ.

Shafer, G., and Logan, R. [1987], "Implementing Dempster's rule for hierarchical Evidence," *Artificial Intelligence* vol. 33, no. 3, pp. 271-98

Shriver, B., and P. Wegner [1987], *Research Directions in Object-Oriented Programming*, MIT Press, Cambridge, MA.

Sussman, G. [1975], *A Computational Model of Skill Acquisition*, American Elsevier, New York, NY.

Toulmin, S. [1969], *The Uses of Argument*, Cambridge University Press, Cambridge, England.

Webber, B., and N. Nilsson [1981], *Readings in Artificial Intelligence*, Tioga Publishing Company, Palo Alto, CA.

Winston, P. [1986], "Learning by Augmenting Rules and Accumulating Censors," in Michalski *et al.* [1986].

Date Due
