MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Working Paper No. 325 August 1989

Decision Representation Language (DRL) and Its Support Environment

Jintae Lee

Abstract

In this report, I describe a language, called Decision Representation Language (DRL), for representing the qualitative aspects of decision making processes such as the alternatives being evaluated, goals to satisfy, and the arguments evaluating the alternatives. Once a decision process is represented in this language, the system can provide a set of services that support people making the decision. These services, together with the interface such as the object editor and the different presentation formats, form the support environment for using the language. I describe the services that have been so far identified to be useful – the managements of dependency, plausibility, viewpoints, and precedents. I also discuss how this work on DRL is related to other studies on decision making.

DRL (Decision Representation Language) is a language for representing decision processes. The goal of DRL is foremost to provide a vocabulary for representing the qualitative aspects of decision making -- such as the issues raised, pro and con arguments advanced, and dependency relations among alternatives and constraints, that typically appear in a decision making process. Once we have a language for representing these basic elements of decision making, it becomes possible to provide services that use this language to support decision making. A DRL environment consists of a set of such services. The present DRL environment include services such as plausibility management, dependency management, and viewpoint management.

This report describes DRL and its environment. In the first section, I discuss the motivations underlying the design of DRL. In Section 2, I describe the DRL constructs. In Section 3, I describe the DRL services that form its environment. In Section 4, I try to justify the DRL constructs by characterizing intuitively the things that we want to represent in decision making processes and illustrating how DRL and other relevant works succeed or fail to represent them. In the final section, I describe the topics for current and future research.

1. Motivations

The goal of DRL, as mentioned above, is to provide a language for representing the qualitative aspects of decision making processes. This goal, in turn, is motivated by the following goals: knowledge sharing, decision support, and a problem solving paradigm based on arguments. I describe each of them below. Also, in Appendix 1, I present a sequence of scenarios that illustrates these motivations in a concrete context.

Knowledge Sharing

Decision making usually involves gathering and relating pieces of knowledge relevant to evaluating alternatives along some criteria. Such knowledge, once explicitly represented, can be shared by others who have to make similar decisions. Past decisions can tell us not only the factual information that we need but also the ways that a decision can be structured, the ways that different goals can be achieved, or the attributes against which alternatives should be evaluated. Furthermore, past decisions provide the additional knowledge of whether the approach they took were successful or not.

Documents,, Basis for Learning and Justification

The records of how decisions were made serve as documents, which in turn serve as a basis for justification and learning. In particular, when the language is used by the system for representing the decisions it makes, the decision records allow a special kind of knowledge acquisition: when a system fails, we can examine the records of the decisions it made and find out why it failed. That way, we can find out what piece of knowledge is missing or not utilized. We can also test the completeness of a system this way before releasing it to the world.

Decision Support

Once we have a language for representing the qualitative structure of decision making processes, the system can provide many services that support decision making. The system can manage the dependencies among objects (e.g. This claim depends on another claim. MacOS is an alternative only if MacIntosh is chosen as the hardware platform, etc.) so that if an object changes its status, its consequences would be propagated. The system can keep track of multiple viewpoints.

A Problem Solving Paradigm based on Arguments

Another motivation underlying DRL is to study the dialectical process as a computational problem solving paradigm. So far, I have discussed DRL mostly in the context of providing support for human decision making. However, the importance of a 'deliberate' decision process -- in the sense of involving arguments pro and con, evaluating alternatives, compromising and negotiating -- is being more appreciated also in automated reasoning. Hewitt [Hewitt 86, Hewitt 88], for example, has been pointing out the features of open systems, such as incompleteness and inconsistency, which makes a purely deductive approach unrealistic. What really 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. DRL is an attempt to identify and articulate the objects and the processes involved in such deliberate decision making processes. DRL, in that sense, can be viewed as a protocol for cooperative and distributive problem solving.

2. DRL Constructs

Figure 1 lists the constructs, and Figure 2 displays them graphically. The fundamental objects of DRL are Goals, Alternatives, and Claims. Alternatives represent the options to choose from, Goals specify the properties of the ideal option, and Claims constitute arguments relevant for making the choice. Other objects are no less essential in a decision making, but either they are special cases of the above three (e.g. Decision Problem is a subclass of Goal) or they are useful in general (e.g. Group, Viewpoint) or they are auxilary (e.g. Question, Procedure). In the following, for each of the objects and relations, I discuss the rationale for having it and what its intended meaning is. An example decision graph, i.e. the representation of a decision problem in DRL, is shown in Appendix 2. In Section 4.2, I illustrate the use of the following constructs through another example.

2.1 DRL Objects

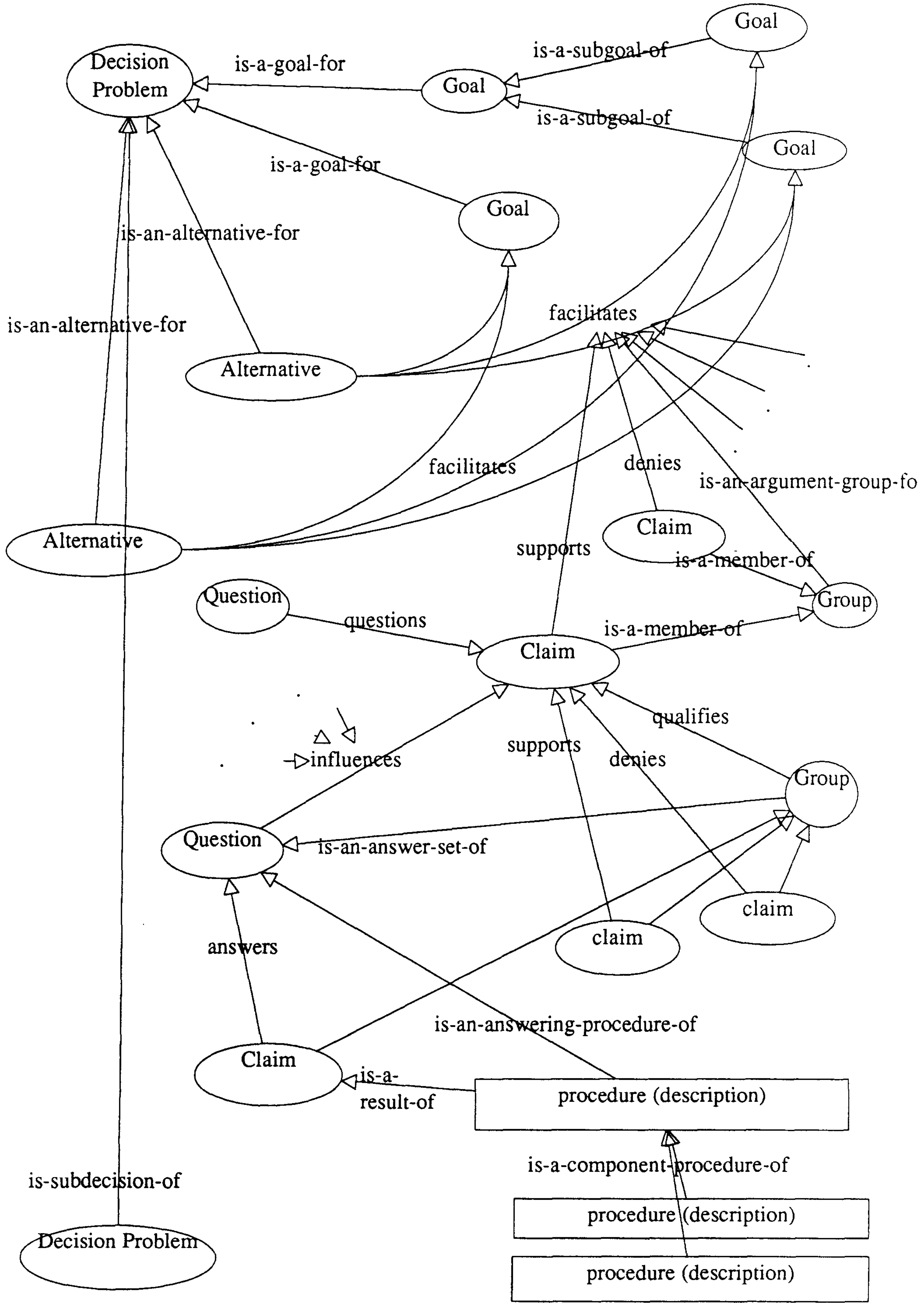
ALTERNATIVE

An ALTERNATIVE represents an option in consideration for the decision problem that it's associated with. An ALTERNATIVE may have other ALTERNATIVES as

Fig. 1 DRL Vocabulary

- Alternative
- Goal
  - Decision Problem
- Claim
  - DRL Relation
    - Is-A-Sub-Decision-Of (Decision Problem, Decision Problem)
    - Is-A-Goal-For (Goal, Decision Problem)
    - Is-A-Subgoal-Of (Goal, Goal)
    - Is-An-Alternative-For (Alternative, Decision Problem)
    - Is-A-Sub-Alternative-Of (Alternative, Alternative)
    - Facilitates (Alternative, Goal)
    - 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-Component-Procedure-Of (Procedure, Procedure)
- Question
- Procedure
  - Procedure Description
  - Executable Procedure
- Group
- Viewpoint

Figure 2. DRL Ontology



its sub-alternatives. For example, 'Obtaining STROBE with SPE' is a sub-alternative of 'Obtaining STROBE'. An ALTERNATIVE is evaluated with respect to each goal, and has the attribute, EVALUATION, which can contain either an evaluation measure (actually an evaluation object, which contains the measure together with the explanation of how the measure was computed) or a pointer to unresolved objects (actually an instance of the object, UNRESOLVED, which contains the pointer) which have to be resolved to compute an evaluation measure.

GOAL

A goal object represents a condition that one wants to satisfy in making a decision. A GOAL may have other GOAL's as subgoals, which can be grouped into a conjunctive set (i.e. all of them have to be satisfied) or a disjunctive set (i.e. satisfying one of them makes it unnecessary to satisfy the others in the set). Also, we need a distinction between a goal which is completely specified by its subgoals (i.e. satisfying its subgoals is equivalent to satisfying the parent) and one which is only partially specified (i.e. satisfying its subgoals is not equivalent to satisfying the goal because it might have some other conditions that are not yet completely exhausted by the conditions specified by the subgoals). These distinctions are captured in terms of SET and the relationship such as CONJUNCTIVE, DISJUNCTIVE, and EXHAUSTIVE.

DECISION PROBLEM

A DECISION PROBLEM represents the top level goal. Hence, it is a special case of goal where the goal is of the form, "Choose X for Y." The other goals are the subgoals of the decision problem, which elaborates what it means to satisfy this top level goal.

CLAIM

A CLAIM represents a fact or a claim asserted as relevant to decision making. DRL makes no distinction between facts and claims. In DRL, any statement is defeasible.

If a statement is very likely to be true, eg. "The Earth revolves around the sun.", the CLAIM representing it would have a very high PLAUSIBILITY-- an attribute that every CLAIM has. The plausibility of a CLAIM depends in part on other CLAIMS that SUPPORT, DENY, and QUALIFY it. This decision not to make the distinction between a fact and a claim is based on the difficulty of making the distinction consistently. What has been regarded as FACTs often turn out to be wrong, as illustrated in the history and philosophy of science.

DRL RELATIONS

All DRL relations are subclasses of CLAIM. For example, 'B IS-A-SUBGOAL-OF A' is interpreted as 'I claim that B should be a subgoal of A'. As such, that claim can be supported, refuted, qualified, or questioned just like any other claims. More discussion of the relations follow in the next section.

QUESTION

A QUESTION represents an uncertain state which requires more information to determine its outcome uniquely. A question might be a simple request for explanation-- eg. Why is G a goal of DP? Or it may represent a major uncertainty whose different potential outcomes might lead the decision in different ways-- eg. Would the next version of X be released in time? The evaluation of X, hence its relative ranking with respect to other alternatives, might depend on how this question is answered-- eg. The next version of X might contain the hypertext feature in need. Even in the first case of a question representing a simple information request, the different outcomes might influence the alternative evaluation in different ways. For example, it might turn out that the simple question cannot be answered satisfactorily and as a result the question can become a refuting claim and the plausibility of the questioned claim can be reduced.

To deal with the different possible outcomes, each question has the attribute, Viewpoints, whose value is a set of multiple VIEWPOINTs that represent possible outcomes. Within each viewpoint, a possible outcome is assumed to hold and the evaluation takes place with that assumption. If and when the outcome is uniquely determined, the viewpoint that corresponds to that outcome is 'realized'. A question may be also related to a procedure via IS-AN-ANSWERING-PROCEDURE-OF (PROCEDURE, QUESTION). This relation is used to record a procedure that can be used to answer the question either in the present context or in another context (i.e. as a precedent, cf. Precedent Management).

PROCEDURE

A PROCEDURE represents either an actual executable procedure or a description of a procedure as may be in the case of human support context. For the moment, a PROCEDURE object is used to represent a procedure that can be used to answer a QUESTION so that this information can be used to execute or have a human being execute the procedure in answering the question. A PROCEDURE may be related to other procedures through IS-A-SUBPROCEDURE-OF or IS-A-COMPONENT-PROCEDURE-OF. IS-A-SUBPROCEDURE relation describes the generalization/specialization relationship among procedures, and is useful when one needs a procedure that is similar but not exactly the same as another procedure. IS-A-COMPONENT-PROCEDURE-OF relation describes the part/whole relationship among procedures, and is used when one wants to describe a procedure in terms of the component procedures that implement it.

PROCEDURE DESCRIPTION
EXECUTABLE PROCEDURE

GROUP

A GROUP represents a set of objects, 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.

VIEWPOINT

A VIEWPOINT groups objects that are under the same set of assumptions. The kinds of objects that one might want to group under a viewoint are:

claims assuming different possibilities
the same set of claims with different plausibilities
different sets of goals
the same set of goals but with different weights
objects at different points in time

2.2 DRL Relations

All the relations discussed here are subclasses of RELATION, which in turn is a subclass of CLAIM. So any relation itself can be related to another claim via the relations that take a claim as an argument (i.e. the relation itself can be supported, refuted, qualified, and so on.)

IS-A-SUBDECISION-OF (DECISION PROBLEM, DECISION PROBLEM)

A decision problem D1 is related to another decision problem D2 via IS-A-SUBDECISION-OF relation if D2 requires solving D1. For example, choosing the best computer environment for one's project has as its subdecisions choosing the hardware, choosing the operating system, and choosing the programming language among others. When that is the case, the alternatives of the parent decision consists of combination of the alternatives from its subdecisions. For example, the alternatives for choosing the best computer environment are:

Sun, Unix, C
Sun, Unix, Lucid Common Lisp
Mac, A/UX, C
Mac, A/UX, Allegro Common Lisp
Mac, MacOS, C

Mac, MacOS, LIsp

As we can see, not all combinations are valid. The user should be able to specify the compatibility among these objects in the form of dependency relationship (Cf. Dependency Mangement).

IS-A-GOAL-FOR (GOAL, DECISION PROBLEM)

A goal C IS-A-GOAL-FOR a decision problem DP if C is a state that the decision maker wants to achieve by solving DP.

IS-A-SUBGOAL-OF (GOAL, GOAL)

A goal C1 IS-A-SUBGOAL-OF another goal C2 if achieving C1 contributes achieving C2. This relation does not yet capture some of the distinctions that need to be captured in the subgoal relationship. For example, a goal might be exhaustively specified in terms of its subgoals or only partially specified. And a goal might have conjuntive subgoals in the sense that they all have to be achieved to achieve the parent goal; or it might have disjunctive subgoals in the sense that achieving one of the subgoals would make the other Goals unnecessary. I have yet to think about how these distinctions can best be introduced in DRL.

IS-AN-ALTERNATIVE-FOR (DECISION PROBLEM, ALTERNATIVE)

An Alternative A IS-AN-ALTERNATIVE-FOR a decision problem DP if A provides an option for solving DP. As such, IS-AN-ALTERNATIVE-FOR relation is a special case of FACILITATES described below because it is to be interpreted as the claim that A facilitates satisfying the top level goal.

IS-A-SUBALTERNATIVE-OF (ALTERNATIVE, ALTERNATIVE)

An alternative A1 IS-A-SUBALTERNATIVE-OF another alternative A2 if A2 is compatible with several choices within its framework and A1 represents one such choice. For example, Buying a Sun 386i IS-A-SUBALTERNATIVE-OF Buying a Sun.

FACILITATES (ALTERNATIVE, GOAL)

FACILITATES is a relation that links an alternative A with a goal C. It is to be interpreted as the claim that the ALTERNATIVE facilitates satisfying the GOAL. As a claim, of course, it can be argued about -- supported, denied, and qualified. Whenever a new alternative is created for consideration, it is automatically linked to all the existing Goals via a FACILITATES relation with its plausibility attribute set to UNEVALUATED. Then as claims are added that supports or denies this claim, its plausibility measure will change accordingly.

SUPPORTS (CLAIM, CLAIM)

A claim C1 SUPPORTS another claim C2 when the plausibility of C2 is enhanced by C1.

DENIES (CLAIM, CLAIM)

A claim C1 DENIES another claim C2 when the plausibility of C2 is reduced by C1.

QUALIFIES (GROUP(CLAIM), CLAIM)

A set of claims {Ci} qualifies another claim C if the plausibility of C depends on which of the claims Ci being true. For example, one might have a claim that LOOPS IS-AN-ALTERNATIVE-FOR the decision problem of choosing a knowledge representation language. However, another qualifies that claim by saying that LOOPS will be an alternative only if the hardware chosen is a Xerox D-Machine. One would represent this qualifying claim by linking that relational claim via QUALIFIES relation to the latter claim that we choose Xerox D-machine hardware.

ARE-ARGUMENTS-FOR (GROUP(CLAIM), CLAIM1)

All the claims that influence the plausibility of CLAIM1, in particular the claims supporting or denying it, are grouped and related to it through this relation.

QUERIES (QUESTION, OBJECT)

A question Q QUESTIONS an object O when Q points to an unclear aspect of O. So if it is not clear what a claim (or any object like goal, alternative, etc.) says, one can create a question object and link it to the object being questioned via this relation. Or if one does not understand why an object is related to another through a given relation (eg. why is C1 a goal of the decision problem DP1), then one can create and link a question object to the relation object (IS-A-GOAL-OF).

INFLUENCES (QUESTION, CLAIM)

A question Q INFLUENCES a claim C if the plausibility of C depends on the answer to Q. In that sense, the group of possible answers to Q, as claims, would be related to CLAIM via a QUALIFIES relation (Cf. Figure 2).

IS-AN-ANSWERING-PROCEDURE-FOR (QUESTION, PROCEDURE)

A procedure (or a description of procedure) P IS-AN-ANSWERING-PROCEDURE of a question Q if it is believed that executing P would produce an answer to Q.

IS-A-RESULT-OF (CLAIM, PROCEDURE)

A claim C IS-A-RESULT-OF a procedure P if C is the information obtained as a result of executing P. This definition is somewhat vague as it stands because 'obtaining as a result of' can be ambiguous. But intuitively it seems clear, and the definition will be elaborated as the needs arise.

ARE-POSSIBLE-OUTCOMES-FOR (GROUP(CLAIM), QUESTION)

A set of claims can represent possible answers to a question. A claim C which IS-A-RESULT-OF a procedure which IS-AN-ANSWERING-PROCEDURE for a question Q may belong to the set which FORMS-POSSIBLE-OUTCOMES for Q unless the result was some bookkeeping information such as errors from the procedure.

ANSWERS (CLAIM, QUESTION)

ANSWERS relates a claim to a question in the special case where there was no need to represent several possible answers to the question, but only one.

IS-A-COMPONENT-PROCEDURE-OF (PROCEDURE, PROCEDURE)

A procedure P1 IS-A-COMPONENT-PROCEDURE-OF another procedure P2 if (it is believed that) executing P1 can involve executing P2. For example, Getting to an airport is an component procedure of Getting to LA by plane.

IS-A-SUBPROCEDURE-OF (PROCEDURE, PROCEDURE)

A procedure P1 IS-A-SUBPROCEDURE-OF another procedure P2 if P1 is a special case of P2 in the sense that some components of P1 are subclasses of P2. For example, Getting to Logan airport IS-SUBPROCEDURE-OF Getting to an airport.

3. DRL Environment

In this section, I characterize the services that form the DRL environment. These services use DRL to provide various bookkeeping management services and support decision making processes. They are described here in a rather general way. Eventually, however, for each of these services, I will identify a set of operations which will serve as its interface. In Appendix 3, I give partial lists of such operations, but they are far from complete. They need to be refined, modified, and expanded.

Plausibility Management

In the course of decision making, people make claims. We need some way of indicating how plausible a claim is. Different kinds of sources exist for plausibility. Rescher (76), for example, classifies them into the following categories: the observational data, personal report (experts, eyewitness), depersonalized historical sources (newspapers, tradition), intellectual resources (assumptions), and cognitive principles (simplicity, regularity). At the moment, we can be neutral wrt what the source is, although later we may incorporate this information. However, since the plausibility of a claim is partially a function of the plausibilities of other claims related to it, we need to define the semantics of how plausibilities should be combined and propagated over the different relations.

There are many studies of this confirmation process -- i.e. the way that a hypothesis should be confirmed or denied as a result of evidence.

I do not want to develop yet another method of confirmation. Instead, it would be ideal if we can provide a hook to use whichever method one wants to use for the propagation of plausibilities. But a problem with many of the existing methods such as Bayes or Dempster-Shafer is that they are based on assumptions that are not likely to be true in many realistic situations -- such as the mutual exclusiveness and exhaustiveness of the hypotheses and the conditional independence of the evidence under a hypothesis. So I need to study whether there are other methods that get around these assumptions. For example, there are some purely qualitative theories of uncertainty management such as Doyle's reasoned assumptions and Cohen's theory of endorsement, which hopefully avoid making these assumptions. Also, since some of these methods have one relation defined between hypothesis and data, namely is-an-evidence-of, I need to think about how this relation maps to the set of finer relations I want to capture such as not only supports or refutes but also qualifies or influences. Once these problems are solved, I will try to isolate the common requirements of these methods so that the common interface can be built for them. For example, we could have different jury modules which provide the same interface but deliberate differently inside.

Although plausibility management is an essential part of decision making processes, working out this part is not a top priority. In the worst case, I feel that I can always leave it for people to actually make the decision. For example, if we represent the different relations among the objects represented explicitly and perspicuously, then I think we can expect people to propagate plausibilities themselves in the sense that they can start out explicitly assessing claims, based on the different sources mentioned above, that are at the very periphery, and then in turn recursively assess the plausibility of the claim that the assessed claims are related to based on the different sources including the assessed plausibilities.

Dependency Management

When changes are made to existing objects, some consistency must be maintained over the knowledge base containing the objects. What kinds of consistency we want to maintain depend on applications. One obvious type of consistency to maintain is the truth-value dependency. Often many arguments depend on a certain condition being true. For example, one might argue that C++ is a good language to use provided that it has a source level debugger, or that Interlisp is a viable candidate language as long as the only company marketing it now survives. We want to represent this sort of dependency and propagate the consequences of a condition when we know its truth value. Or if we have to make a decision without knowing whether the condition is true or not, we would like to associate with the condition certain actions to be performed in assessing the plausibility of other claims related to it -- such as assuming the condition true, calculating the plausibility using the probability assigned to that condition, and so on.

Another type of consistency that one might want to maintain is the compatibility among objects. Suppose one is trying to choose a computer environment and partitioned the problem in terms of choosing a hardware, an operating system, and a programming language. The actual set of alternatives would hence be combinations of an hardware, an OS, and a programming language -- e.g. (Sun, SunOS, C) (Mac 2, A/UX, C). However, some of these combinations are incompatible. One may want to indicate to the system these dependencies among the different components so that certain incompatible combinations would not be considered at all (e.g. Sun and MacOS)

Viewpoint Management

I define viewpoint management as the set of services responsible for creating and manipulating a viewpoint, which is in turn defined as an object that groups objects which presuppose the same set of assumptions. In this sense, dependency management can be viewed as a special class of viewpoint management, because dependency can be maintained by explicitly creating viewpoints. The examples I gave above of the dependency management, however, are the things that you might want to do without having to create separate viewpoint objects. Viewpoint management is one aspect that I want to focus on in my thesis both because providing multiple perspectives is such an important problem that I come across whereever I turn and because it is a stepping stone to the precedent management, which was the original motivating force behind DRL. The other services such as plausibility, selective retrieval and presentation are no doubt very important parts of representing decision processes, but for the purpose of my thesis I would be content with providing the minimal features that are necessary for the functioning of the DRL system.

Viewpoint management was characterized in Lee [1989a] as also being responsible for maintaining a semantically consistent state over the knowledge base in the face of change. I distinguished viewpoint management from dependency management in terms of modification versus elaboration. Viewpoint management was said to be concerned with the types of change that require modification of existing objects, and dependency management with the types of change that introduce additional objects into the existing structure. The rationale was that when you want to modify an object, you would want to create a new viewpoint. I no longer think that this distinction between modification and elaboration is important. I think there are many cases where you want to modify existing objects but do not want to create a separate viewpoint or you simply add new objects but under a different viewpoint.

It is important to capture and relate different viewpoints. The system should allow people to argue under a specific assumption and group those arguments so that we can compare the consequences of different assumptions and cancel the whole collection of arguments based on certain assumption when that assumption is known false. The system should allow different viewpoints to be created when the goals get assigned different importance. Different viewpoints would be needed when the arguments are given different plausibilities. One should be able to freeze the present state into a viewpoint so that you can later see the way that the decision making unfolded or even revert back to this state if some assumption was wrong. Here, viewpoints would provide the version mechanism, but the relation between the viewpoints would be more than simply that of chronological one as between versions. The relations among them should include:

Is-A-Next-Version-Of
Elaborates
Restricts
Has-Different-Importance
Has-Different-Plausibilities.
Is-A-Child-Of

The ability to establish mappings between objects in different viewpoints would be also important. Such mappings would show how the problem representation has evolved in the course of a decision making process. (e.g. Subalternatives become flattened alternatives in the next version, What was a question in one version becomes a refuting claim in the next.) It would also be useful if we have to merge viewpoints. For example, if the marketing group and the design group were both evaluating alternatives for a computer environment, it would be nice if they do so within their own viewpoints and then later merge them for an overall evaluation. (What such merging would require is yet to be worked out, but I am thinking about a way to link this work to the work I did earlier on Partially Shared Views scheme for translating between different type hierarchies [Lee & Malone 88] and on Knowledge Base Integration [Lee 87].)

Another feature that we want, and potentially related to the concept of viewpoint, is a measure of consensus. As the DRL system is designed to be used by many people contributing their knowledge, it should be able to indicate what the consensus is on a particular claim or on an alternative overall. This measure of consensus can be incorporated into the plausibility measure or the evaluation measure; or it could be a separate measure. I plan to look at some related literature such as Lowe's SYNVIEW [1986] that discusses some measure of consensus among people who participate in a debate.

Precedent Management

There must be some operations that take the description of the present decision problem and find past decision problems that are relevant to the present problem. It would be even nicer if there were operations that allow extraction of specific parts that are relevant from the retrieved precedents. The language, DRL, should of course allow representing whatever it is that is being used for judging relevancy. For example, it seems reasonable to use similarity among goals to determine relevance. Then it is important that one should be able to represent in DRL not only the goals that come up in a decison problem but also a structure (say, a lattice) in which the goals are related to other goals in various ways (e.g. IS-A-SUBGOAL-OF, CONFLICTS-WITH).

Object Editor

We should be able to create, delete, modify, and relate objects that will eventually represent what we consider is important in decision making processes. In particular, we should be able to represent such qualitative aspects of decision making as the different ways that objects can be related to one another. Claims support, refute, or qualify others. Some goals are subgoals of another and in different ways. It may be that satisfying a goal means satisfying one of subgoals or all of its subgoals. Or all the subgoals do not specify a goal completely so that satisfying all of them does not amount to satisfying the parent goal. The language has to provide the vocabulary expressive enough to capture these distinctions.

Another feature that the language has to provide is the ability to represent meta-claims that arise in decision provesses. For example, one might want to say that we should stop exploring further alternatives. A claim of this sort is not about the evaluation of a particular alternative, hence has a status different from other claims which do evaluate alternatives. The way that these meta-claims are related to the other claims and the role they play in the decision making process has to be sorted out and represented.

Selective Retrieval

We want to define and look at different subsets of the objects represented.

***** show all the subgoals
***** show all unresolved objects and their status
***** show all the claims that depend on this claim
***** show all the claims that refute this claim
***** show me all the claims that have been added since the last time I looked.

We need a query language for specifying these kinds of requests.

Presentation

Presentation is an important part of the system because without good presentations people are very reluctant to use the system. Although presentation will not be a main part of this work, we cannot avoid thinking about the different ways of presenting information to the extent that we want the system actually to be used by people. There will be more specific discussion of different formats of display and associated operations in the user-interface section.

4. Related Work

In this section, I provide a basis for judging the DRL model by relating it to other relevant works. The studies I discuss are: the analytic decision model (rational actor model maximizing utility), Toulmin's theory of arguments, Doyle's model of deliberation, and gIBIS (graphical Issue Based Information System). It would be ideal if I could represent each of these works in a common underlying model and compare them. However, that is difficult to do for a couple of reasons. Most of these works are not precise enough to have a model. For those which do have some sort of models (e.g. decision theory, Doyle's theory), it is not clear what the common model should be.

Short of that, I am going to relate the DRL model to other models by first characterizing the things that we want to represent and then see how each model can or cannot represent them. Doing so helps us see the different ways that the same structure is represented or cannot represented at all. So first, I give an intuitive characterization of the things that you would want to represent in a decision making process. After having done so, I will describe how DRL represent them. Then, for each of the work mentioned above, I will discuss why it was chosen for comparison, provide a brief overview, and describe how it can or cannot represent the things discussed above.

4.1 Desiderata

The following is an intuitive characterization of the things in decision making processes that we want to represent.

** Alternatives

Alternatives are those that one is trying to choose from. E.g. the different workstations like Sun or MacIntosh. In fact, an alternative does not stand for objects, but a proposition like "We use a Sun." or "We use a Mac."

** Goals
I am using the term Goal broadly to refer to any condition or property that one wants an ideal alternative to satisfy. Thus, goals include what some people might want to call constraints or desirable properties. e.g. "Implement DRL." "Has A Good Debugging Environment."

*** Relationship among Goals
The goals are related in different ways. The following lists a few possible relations.

**** Sub-Goals
A goal can be one way of achieving another goal. These subgoals, in turn, can be related among themselves in the following way:

***** Conjunctive, Disjunctive
Satisfying a goal requires satisfying all the members of a given set of other goals.

***** Completely Specifying, Partially Specifying
Satisfying a goal requires satisfying only one of a given set of other goals.

**** Conflicting Goals
Satisfying a goal can prevent or make it more difficult to satisfy another goal.

**** Concording Goals
Satisfying a goal can facilitate satisfying another goal.

** Factors Relevant for Evaluating Alternatives

Examples of such factors are: facts, claims, opinions, possible states of nature, and informed guesses about them. These factors can be related among themselves in different ways. Some make others more likely or believable or have the opposite effect.

** History

We want some way to represent the history of decisions that led to the current state so that we can do, for example, credit-assignments. In a way, that is the motivation of the whole DRS, but we want to do that at local levels as well.

** Questions
** Answers
** Meta-Comments

There will inevitably be questions, answers, and meta-comments, i.e. comments not about alternatives but nevertheless relevant to decision making -- e.g. comments about whether we should go on searching for more relevant factors or make the decision now with what we have. It would be nice to integrate these objects to the representation of other objects discussed above.

** Status on Objects (the ranking of alternatives at the moment, etc.)

Often, we need to know the status of certain objects are. Different types of objects will have different types of status information. For example, associated with an alternative can be information about its current ranking. Associated with a claim can be its current plausibility. Associated with a question can be the status about whether it has been answered or not.

** Specialize/Generalize Relation among Objects

Sometimes, an alternative has to be specialized. E.g. Sun 4 specializes the alternative Sun. The sub-goal relation discussed above is also an example. The whole decision problem sometimes have to be specialized. E.g. Choosing a computer environment needs to be broken up into choosing a hardware, choosing an operating system, and choosing whatever software one needs.

** Different Views

We often want to look at the information collected from different perspectives. We might want to evaluate alternatives with only a subset of the goals, with the goals weighed differently, with different probability assigned to the possible state of affairs, and so on. It would be nice if we could not only look at them from different perspectives, but also explicitly represent these perspectives so that we can have a meta-level view of how these perspectives themselves are related.

** Different Degrees of:

*** Importance
Some goals are more important to satisfy than others. And some are absolutely necessary.

*** Uncertainty
Some situations are more or less likely to happen than others. Or at least we are more sure about some events than others.

*** Plausibility
Some claims are more plausible than others.

*** Performance
We need to represent the degree to which an alternative satisfies a given goal.

*** Evaluation

We need to represent the overall extent to which an alternative satisfy a set of goals. For example, alternatives would be ranked on the basis of this evaluation measure.

4.2 DRL

The objects of DRL has been discussed in Section 2. Here, I illustrate how they are used to represent the desiderata discussed above by walking through an example.

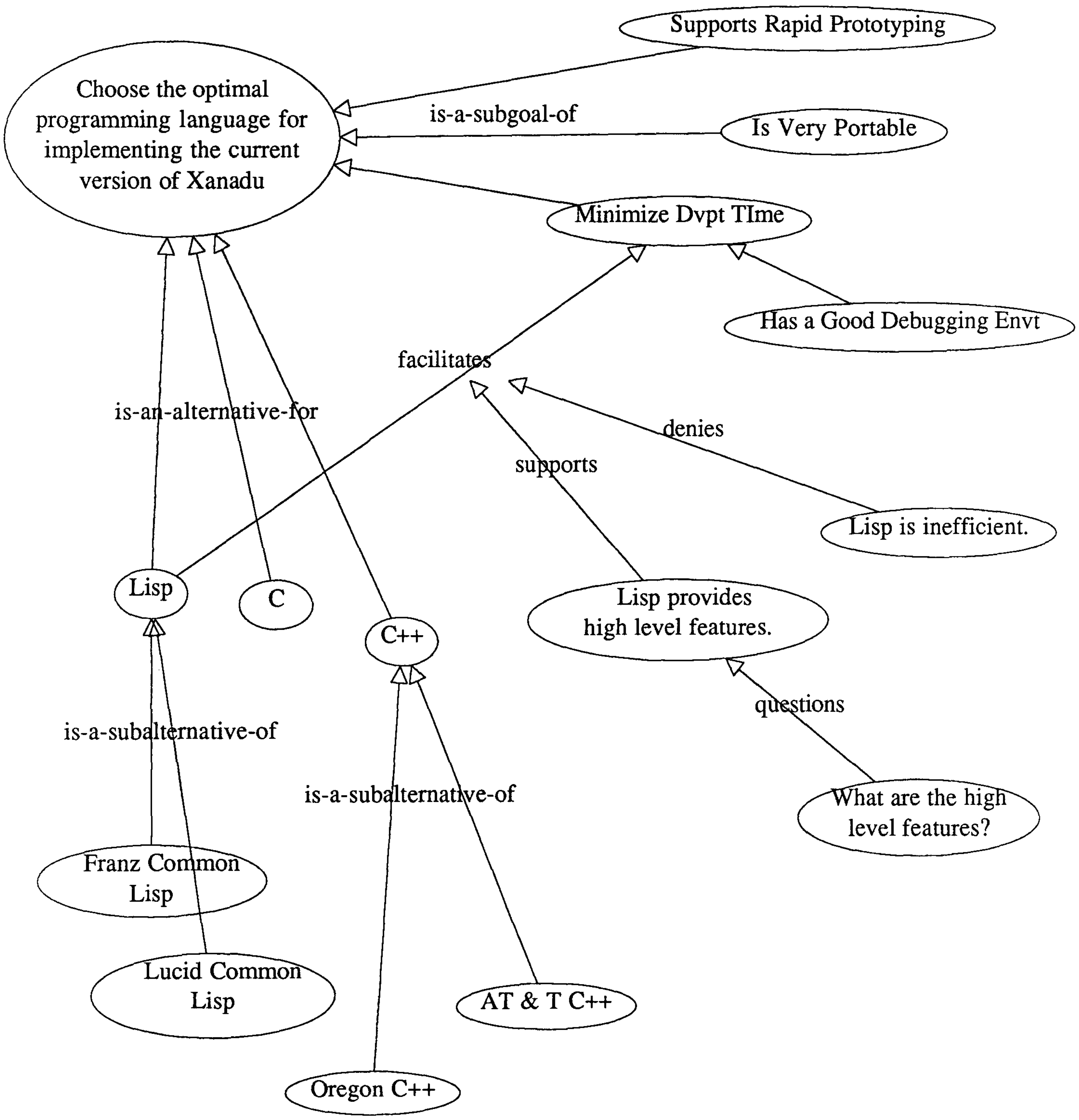
Suppose we want to make a decision about which programming language to use for implementing a project called Xanadu. First, we would create an instance of Decision Problem called "Choose the best programming language for implementing Xanadu.." Then, we represent the relevant goals as instances of Goal. We would, for example, create the following instances of Goal": "Supports Rapid Prototyping", "Minimizes Development Time", "Is Very Portable.", "Has a Good Debugging Environment", and so on. If satisfying a goal facilitate satisfying another goal, the two goals are related via the IS-A-SUBGOAL-OF relation. Hence, "Has a Good Debugging Environment" is a subgoal of "Minimize Development Time". The different relationships among the subgoals are represented in terms of GROUP. A GROUP allows us to group a number of objects and specify the relationship among them. So we can specify a set of subgoals to be members of a set whose Member-Relationship property can be such as the ones mentioned in the previous section: conjuntive, disjunctive, exhaustive, mutually exclusive, and so on.

The alternatives from which we want to choose from are represented as Alternatives and linked to the decision problem via IS-AN-ALTERNATIVE-FOR: C, AT & T C++, Oregon C++, Lucid Common Lisp, and Franz Common Lisp. [Strictly speaking, the alternatives should be not objects (C or Lisp) but propositions ('We use C' or 'We use Lisp.'). Cf. Section 3. But, with that understanding, I'll sometimes talk as if the alternatives are objects themselves.] The relation IS-AN-ALTERNATIVE-FOR (ALT, DECISION-PROBLEM) is interpreted as "ALT should be considered as an alternative for DECISION-PROBLEM", which in turn means "ALT facilitates satisfying the goal represented by the DECISION-PROBLEM instance." This decision problem is shown in Figure 3.

Each alternative, when created, is automatically related to each goal via a FACILITATES relation. In DRL, every relation is a CLAIM. The IS-AN-ALTERNATIVE-FOR relation, as we discussed above, was interpreted as the claim that "ALT facilitates satisfying the goal represented by the DECISION-PROBLEM instance". The FACILITATES (ALTERNATIVE, GOAL) relation is interpreted as the claim that ALTERNATIVE facilitates satisfying the GOAL. As such, these relations can be supported, refuted, or qualified as other claims can. Of course, the extent to which the alternative satisfies the goal is a matter of degree, and the PERFORMANCE measure of FACILITATES relation reflects this extent. This extent is, in turn, a function of the arguments that people put forward as relevant for evaluating the alternative with respect to the goal.

In DRL, an argument is represented as a set of CLAIMs. For example, suppose one wants to say that we should use Lisp because it provides high level features . One would first create the claim, "Lisp provides high level features." One would then find one or more goal that gets satisfied by virtue of that claim being true. There must be such a goal because goals are supposed to reflect all the desirable properties that an alternative should have. Hence, if such a goal is not found, then the list of goal is not complete; and one should be added, possibly as a subgoal of an existing one. In our example, the goal being satisfied is "Minimizes Development Time."

Fig. 3 An Example of DRL Decision Graph



Once the relevant goal(s) are found, one links the claim via SUPPORTS relation to the FACILITATES relation between the alanine alternative and the found goal. (IS-AN-ALTERNATIVE-FOR relation is in fact a special class of FACILITATES relation.) In DRL, all the factors that enter into an argument are represented as instances of CLAIM. DRL intentionally makes no distinction between facts, claims, opinions, or possible states of affairs because I believe, like many philosophers of science, that these distinctions are only matter of degree to which the assertion is accepted. Introducing these distinctions introduce the problem of ambiguity about how to classify a given assertion. It seems better to have a uniform type with an attribute, called Plausibility, for indicating the degree to which it is accepted.

Claims can SUPPORT, DENY, or QUALIFY other claims. So the claim that "LISP provides high level features." SUPPORTS the FACILITATES relation (, which is itself a claim) between the alternative "Use Lisp" and "Minimize Development Time." Another claim can DENY this claim by saying that we do have some knowledge about the current situation and pointing to a claim that does represent such knowledge. A claim can be QUALIFIED by a CLAIM or a SET of CLAIMs. The semantics of QUALIFIES (CLAIM1, GROUP of CLAIMS) is that the plausibility of CLAIM1 depends on which of the claims in GROUP of CLAIM is true. The relationship among the claims in a group may be related by the relations one specifies in the Member-Relationship attribute of the group: e.g. mutually exclusive, exhaustive, and so on. If the truth of the claim is not known, then the plausibility of the claim qualified remains tentative. If the decision has to be made with the qualified claims whose plausibility is tentative, the plausibility of the claim is calculated with appropriate weights given to the possibilities -- in this case, the plausibility of the claims reflect the probability of the possibilities they represent.

Another relation that a claim can have is being QUESTIONed. In fact, any object, not just a claim, can be QUESTIONed. A QUESTION QUESTIONS a claim (Note that the object QUESTION is not the same as the relation QUESTIONS) if somebody wants a more information about the claim. Hence, the presence of a QUESTION object indicates the need for clarification. For example, if one does not know what high level features that Lisp provides, one would create a question object and link to the object in question, in this case the claim object "Lisp provides high level features". People answer a question by creating an Answer object and link to the question object via ANSWERS relation. Or if the procedure of answering a question is involved and worth recording, then the procedure or its description is linked to the question object via IS-AN-ANSWERING-PROCEDURE-FOR. A PROCEDURE can be an EXECUTABLE PROCEDURE or just a PROCEDURAL DESCRIPTION, i.e. a description of a procedure. A PROCEDURE can be related, via the IS-A-COMPONENT-PROCEDURE-OF relation, to other PROCEDURES, which describes the original procedures in detail. On the other hand, a procedure is related to another procedure via the IS-A-SUBPROCEDURE-OF relation if the first procedure generalizes the second procedure, though neither is part of the other. For example, Evaluate the Debugging Environment of Oregon C++ would be a component procedure of Compare the Debugging Environments of the Alternatives, but not its subprocedure.

In DRL, a VIEWPOINT represents a collection of other objects and their relations. VIEWPOINTs are first-class objects so you can browse them, relate them, or perform operations defined on them. Thus, if you want to consider only a subset of goals, say, related to implementation, then you would collect only those objects linked to the goals in the subset into a viewpoint called Implementation Viewpoint. When you decide to incorporate other goals about, say, Cost, you can create another viewpoint, called Implementation with Cost Viewpoint, which incorporates all the objects linked to relevant goals. You can then relate this viewpoint to the previous viewpoint, Implementation Viewpoint, by one of the possible relations that can hold among viewpoints. Such relations include: Elaborates, Elaborates Alternatives, Elaborate Goals, Restricts, .., Is-A-Next-Version-Of, Has-Different-Weights-on-Goals, and so on. This way, whatever different parts of the graph or different assumptions one wants to make, can be made explicit and considerations pursued within a local environment.

The different degrees of importance, uncertainty, plausibility, performance, and evaluations discussed in Section 1 are all represented as an attribute value of the relevant objects. For example, a goal has the Importance attribute, which indicates how much weight we should give in evaluating an alternative with respect to the goal. A claim has the Plausibility attribute, which indicates how plausible it is. This plausibility is based on many factors -- the credibility of the source from which the claim originates, uncertainty of the event that the claim represents, as well as the plausibilities of other claims that are related to it. Hence, DRL has no separate measure for uncertainty because no event is directly represented in it; instead, whatever uncertainty is reflected in the plausibility of the claim representing the event in question. A FACILITATES (ALTERNATIVE, GOAL) relation is a subclass of Claim, hence it has the Plausibility attribute. In addition, it also has Performance attribute, which indicates the extent to which ALTERNATIVE facilitates satisfying the GOAL. This value is a function of the plausibilities of all the claims that are linked, directly or indirectly, to the FACILITATES relation. An Alternative object has the Evaluation attribute, which indicates the extent to which the alternative facilitates satisfying all the goals that have been specified for the given decision problem. This value is a function of the evaluation measures of the alternative with respect to all the goals, properly weighed with their importance.

We also want to associate the history of how a decision was made with the outcome of the decision. In our example, that means associating a particular sequence of amino acids or a set of them with the history of how it was arrived at. This can be achieved by storing in the History attribute of the object the technique used to obtain the object together with the previous state to which the technique was applied as well as the pointer to the records of the decisions that have led to the choice of the technique.

4.3 Analytic Decision Theory

Since DRL attempts to represent and support decision processes, it is worthwhile to compare it to the analytic decision theory, which has similar goals but has been around much longer.

Analytic decision theory starts with a set of Actions and a set of possible States. Each of the possible states is associated with the probability of its actual occurrence. The theory also assumes that we can assign a utility measure for each Consequence, which is a state resulting from an action applied over a possible state. Then, the theory judges an Action to be optimal when it maximize the expected utility over all possible states.

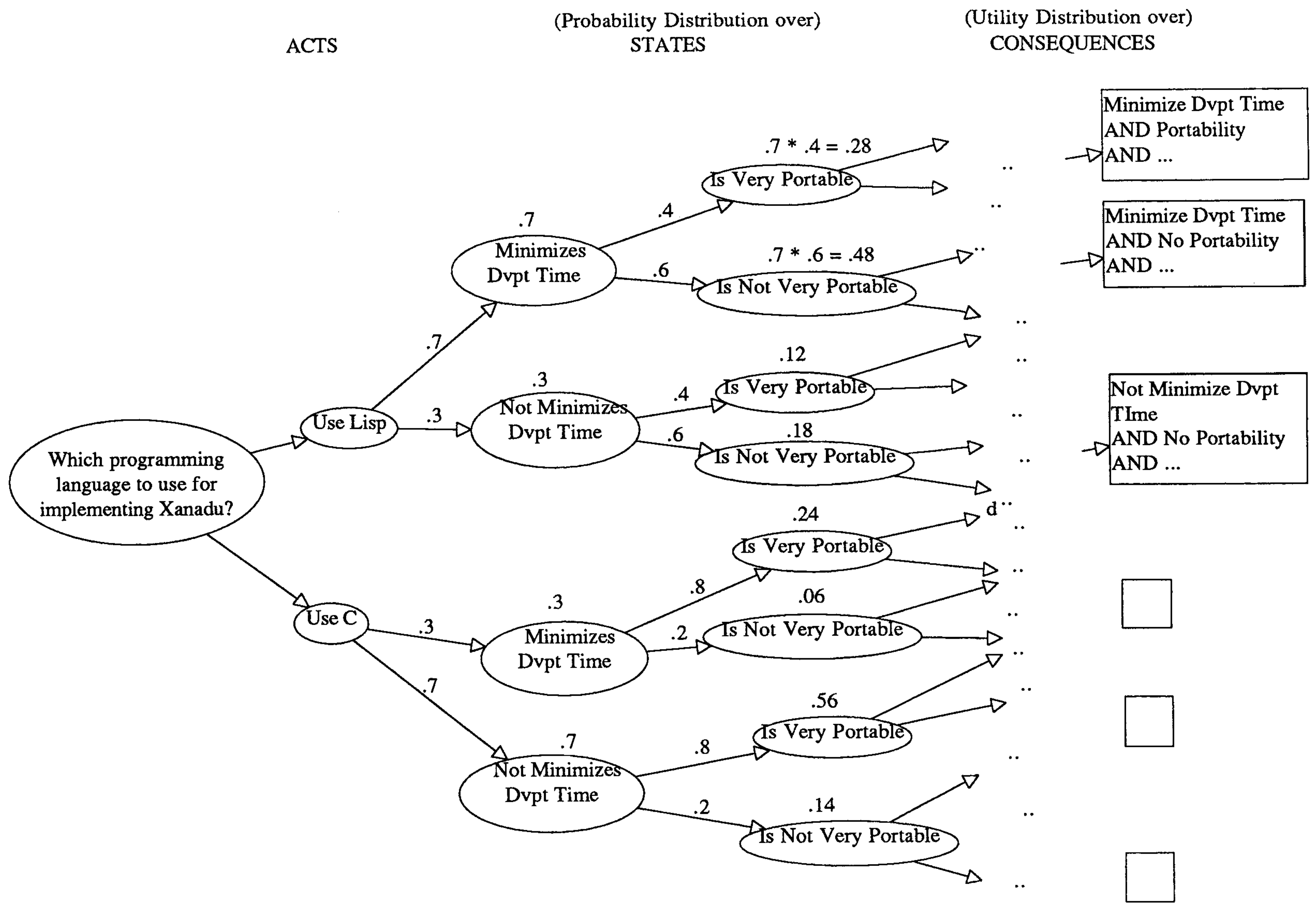
In the case of our example, using the different programming languages correspond to the set of actions (Cf. Figure 4). The relevant consequences are such as 'Finding Out that the Chosen Language is a Disaster for Xanadu', 'Finding It Adequate', and 'Finding It Perfect'. The states are the conditions that matter in evaluating the consequences of the different actions. In our example, they include: whether a language supports rapid prototyping, whether it minimizes development time, whether it is portable, and so on. Hence, the states correspond roughly to the subgoals. The analytic decision theory assumes that we know what these relevant conditions are and the probability of the conditions being true for each of the actions taken. In that sense, it presupposes the whole qualitative structure that others like DRL or Toulmin's tried to capture. Once we know them, however, the theory dictates that we choose that action that maximizes the expected utility.

Of the features listed in the beginning, the only ones represented explicitly in the decision theory are:

ACTS represent alternatives;
POSSIBLE STATES is one of the factors that enter into an argument;
PROBABILITY assigned to these states represent the different degree of uncertainty;
CONSEQUENCES represent goals, but only implicitly.
UTILITY represents the different degree of evaluation of the alternatives, which, in turn, is a result of combining all the other factors listed above.

Goals are represented only implicitly by Consequences and States. In the above example, one might argue that the Consequences and the Goals -- 'Choosing the optimal programming language for implementing Xanadu' and 'Finding the chosen

Fig. 4 Analytic Decision Theoretic Representation



ACTS
(Probability Distribution over)
STATES
(Utility Distribution over)
CONSEQUENCES
Which programming language to use for implementing Xanadu?
Use Lisp
Use C
.7
.3
Minimizes Dvpt Time
Not Minimizes Dvpt Time
.4
.6
.8
.2
Is Very Portable
Is Not Very Portable
.7 * .4 = .28
.7 * .6 = .48
.12
.18
.24
.06
.56
.14
Minimize Dvpt Time AND Portability AND ...
Minimize Dvpt Time AND No Portability AND ...
Not Minimize Dvpt TIme AND No Portability AND ...
d

programming languge perfect for implementing Xanadu' -- are the same things. Often, however, they are not the same. For example, when we decide whether to buy an insurance, a goal is to minimize cost. However, the consequences are specific states such as having an accident but covered by the insurance, or having an accident and having to pay for the damage yourself. The goal of minimizing cost is implicitly reflected in the utility measure assigned to these consequences, but it, by itself, is not explicitly represented independent of any particular realization.

However, there are good reasons for making goals explicit. The explicit representation of goals allow modular representation of arguments. For example, if goals remained implicit as in the above example and if a goal were to change -- e.g. portability were not so important because the current version is only going to be a prototype, it would be difficult to determine which part of the arguments would be affected by this change. Also, the explicit presence of goals forces people to articulate the goals as well as their subgoals. Furthermore, by making goals explicit, we can argue about them -- whether they should be the goals or how important they are.

Goals in DRL also serve as a basis for determining the relevance of past decisions to the decision in question. Those precedents which share more goals are more similar, and these shared goals can be used to lift out, so to speak, portions of those precedents in order to transfer the knowledge to the current context. Furthermore, making goals explicit allows different viewpoints possible; if there are disagreements as to which goals should be pursued, the decision process can be factored with respect to multiple subsets of goals and the alternatives evaluated with respect to each of them.

All the other qualitative aspects -- factors that enter into arguments, questions, meta-comments, in fact anything that is relevant in evaluating alternatives -- are merged into the utility measure. Of course, that means that all these factors are not represented explicitly. Hence, the theory will not help you extract any qualitative knowledge that are cumulated in decision making processes, but only will help you make a choice from the existing options.

4.4 Toulmin's Theory of Argument

A British philosopher, Stephen Toulmin, has proposed a model of argument in his book, The Uses of Argument [Toulmin 69]. Since then, this model has been adopted by many people in modeling dialectical processes. It has been used for computationally modeling arguments ([Birnbaum et. al. 80] [Lowe 86], in interpreting literary text [Wunderlich 74] as well as general text [Huth 1975]. It has been also related to the theory of truth [Habermas 73] and legal argumentation [Pratt 70].

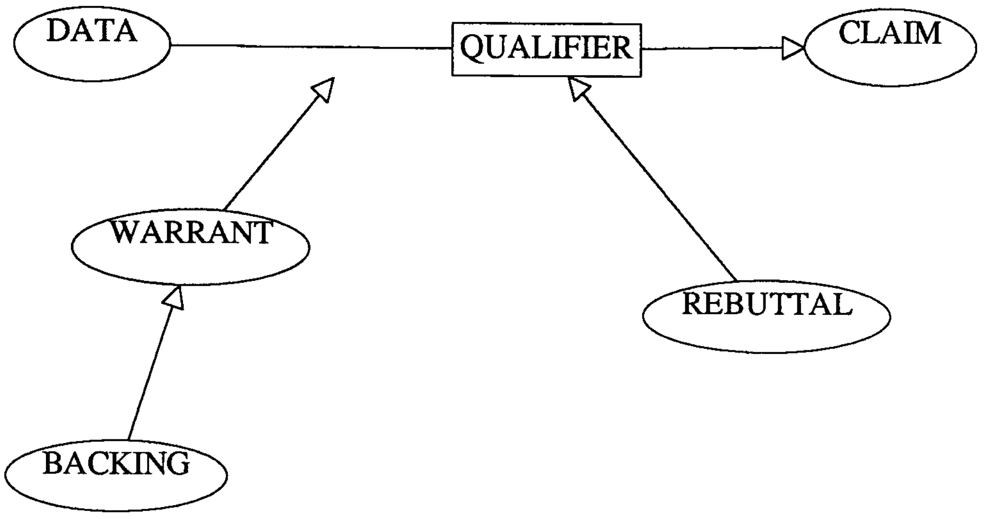
Because it has been so widely used, it is worth comparing this model with that of DRL. First of all, it is worth noting again that Toulmin's is a model of argument, which is a subset of what DRL is trying to represent. Hence, I will compare Toulmin's model with only those aspects of DRL that is trying to represent arguments. First, I describe Toulmin's model, illustrate with an example, and discuss why the DRL model is different from Toulmin's.

Figure 5(a) shows the object types in Toulmin's model. A CLAIM is the main assertion being made, DATA is what supports the claim, WARRANT is the basis on which DATA is said to support the claim, BACKING is what, in turn, supports WARRANT, QULAIFIER is what qualifies the extent to which the data supports the claim, and REBUTTAL is what gives a reason for QUALIFIER.

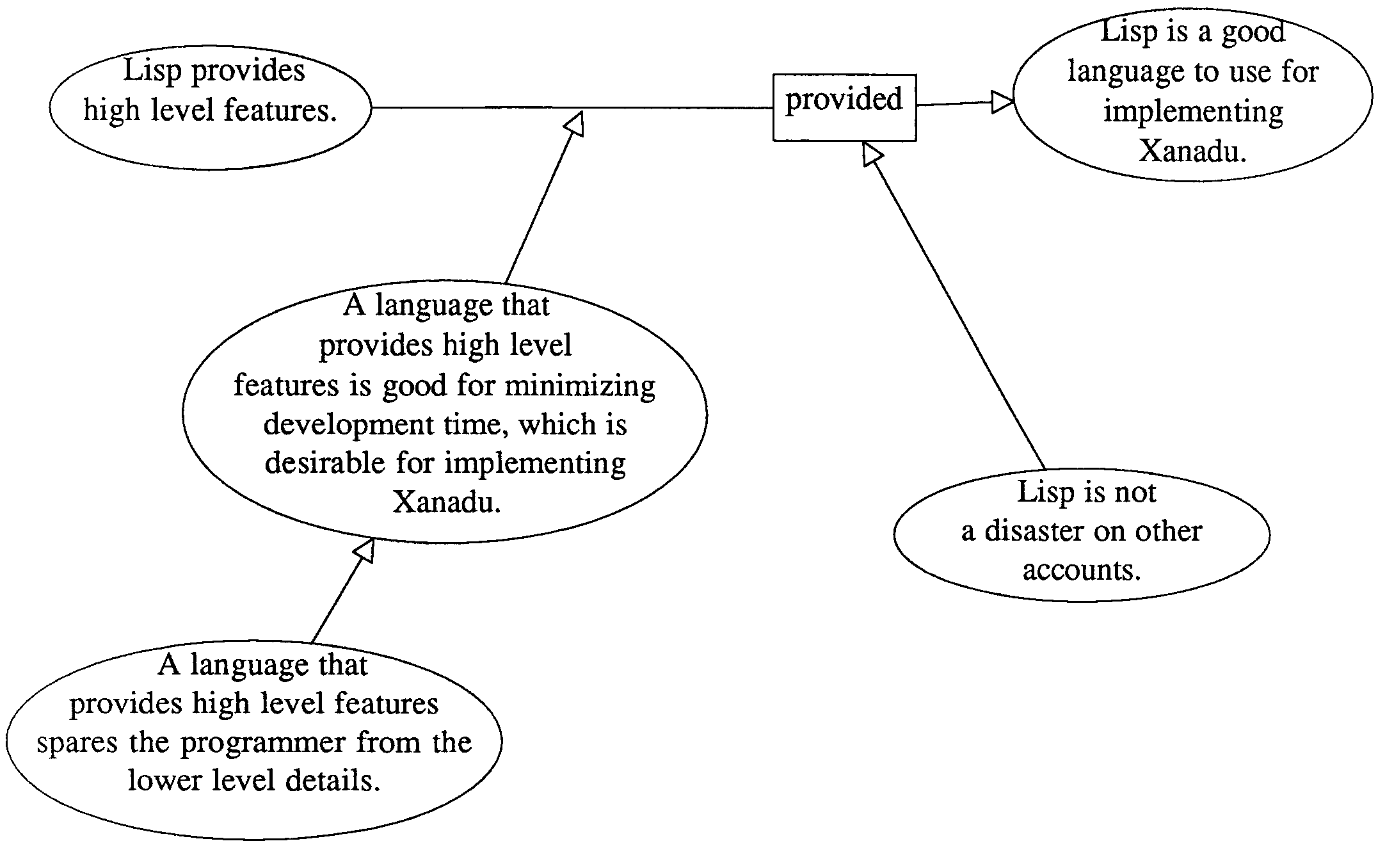
Figure 5(b) shows the representation of our example in Toulmin's model. The CLAIM is "Lisp is a good language to use for implementing Xanadu." A DATUM may be "Lisp provides high level features." A WARRANT may be "A language that provides high level features is good for minimizing developing time, which is desirable for implementing Xanadu." A BACKING is "A language that provides high level features spares the programmer from the lower level details." A QUALIFIER can be something like "provided that Lisp is not a disaster on other accounts."

Fig. 5 Toulmin's Model of Argument and an Example

(a) The Model



(b) An Example


Lisp provides high level features.
provided
Lisp is a good language to use for implementing Xanadu.
A language that provides high level features is good for minimizing development time, which is desirable for implementing Xanadu.
Lisp is not a disaster on other accounts.
A language that provides high level features spares the programmer from the lower level details.

One problem with this model is that some of its distinctions are arbitrary. What makes "Lisp is a good language to use for implementing Xanadu." a CLAIM, but not "Lisp provides high level features." or "A programming language that provides high level features is good for minimizing developing time, which is desirable for implementing Xanadu."? And what makes "A programming language that provides high level features spares the programmer from the lower level details." a BACKING rather than a DATUM? One answer is that it is context-dependent so that a CLAIM is what is being argued for, a DATUM is what supports or refutes it, and so on. In other words, the same thing can play different roles in different contexts.

This context-dependent typing, however, undermines the uniformity and extendability of the representation. For example, Toulmin's model only allows a CLAIM to be backed up, qualified, or rebutted but not a DATUM or a WARRANT. What shall we do if we want to express disagreement with "Lisp provides high level features." or "A programming language that provides high level features is good for minimizing developing time, which is desirable for implementing Xanadu."? Also, in Toulmin's model, WARRANT is what allows DATA to support CLAIM, and BACKING is what supports WARRANT. What do you then to support BACKING?

Many other desirable distinctions that we discussed above cannot be expressed in Toulmin's model. That's partly because Toulmin's model is that of argument proper, but not entirely so. One can refute an existing claim only in a round-about way by supporting its negation (which does not really mean the same thing anyway). One can qualify a claim, but whether that qualification is due to some uncertainty over situations, less than perfect plausibility of data, or the weak relation between data and claim is not clear. It does not leave rooms for expressing questions, answers, meta-comments, or different viewpoints. One might, however, argue that they are not part of argument proper, although they are bound to appear in any complex arguments.

4.5 Doyle's Model for Deliberation

Doyle, in [Doyle 1980], has proposed "a model for deliberation, action, and introspection". It is worth relating the DRL model to his model for several important reasons. First, it is the most comprehensive and detailed model that I have seen so far of defeasible reasoning, i.e. where the assertions are non-monotonic. As Doyle points out, this provides an important framework for dialectical reasoning, i.e. reasoning based on arguments, as opposed to pure deduction. Second, it is a computational model. The goal of the model is to control or direct the reasoning and actions of a computer program by explicitly representing and justifying the reasoning process itself. As such, it has special relevance to my goal of developing a computational problem solving paradigm based on arguments. In particular, I need to justify how my model is different and why. Third, it also discusses the meta-level decision making in the context of deciding how to reason.

Doyle's theory is, as said above, very comprehensive. If I were to present all the aspects of the model, it would take as much space as the original paper, probably more because the original is quite dense. Hence, instead of explaining the concepts in abstract, let me illustrate it through an example. I'll use the same example I've been using.

The basic primitive objects in his model are Concepts, Desires, Intentions, and Beliefs. A concept represents an object that one needs to reason about. In human interactive context, we may not need concepts because human users can extract concepts out of free texts. However, in computational context, we need some way of representing whatever objects the system needs to reason about. In our example, the concepts are C, Development Time, High Level Features, Portability, etc.

Desires and Intentions represent what we have been referring to as goals so far. Doyle argues that goal is an ambiguous concept that confounds the two concepts -- desire and intention. To summarize, we can think of desire as a mental attitude which is satisfied when a certain condition is realized. Hence, wanting to find a very portable language is a desire. That desire is satisfied when we achieve the condition of having found such a language. Intention, on the other hand, is a mental attitude associated with an action and it is satisfied when the action is taken. Hence, the intention to find a portable language is satisfied when a certain action is taken toward that aim, whether that aim is satisfied or not. The more specific the intention is, the more specific the action would be. A belief is a mental attitude which is more or less justified depending on the other beliefs and the state of nature. "The higher level features a language provides, the less development time it will take." is a fairly reasonable belief though by any means a totally justified one.

Plan, Policy, and Reason are higher level structures linking other objects. "Plans are ways of describing the structure of one's desires and intentions." A plan specifies how a desire can be satisfied in terms of other desires or intentions. A plan, say X, for breaking a tie between two candiates may be "First, find out the differences other than the ones considered already. Order the differences by their importance. Then, for each difference, do so and so ... " A policy is an intention "with conditional or hypothetical statements as their aims". For example, "Consider the plan X whenever there is a tie between two alternatives." is a policy. Policies express temporal ordering relationships between intentions, embody the strengths of desires, as well as embody many other preferences of the program, such as those used in belief revision to chose one possible revision over another. The actions of a policy can be a plan as in the above example, or a primitive. An example of a policy with a primitive action is "If you have to break the tie and if you don't have much else to go by, add a pro-argument for the option of using Plan X" The action of this policy makes the option of using Plan X more attractive, whereas the action of the previous policy just added Plan X as an option. (This way, the actions of a policy either add new intentions as subordinates of the decision intention, options to the list of options, considerations to the list of considerations, or reasons to the list of reasons.)

A Reason is a record of how an attitude, such as Desire, Intention, or Belief, has been arrived at. Hence, in Doyle's model, every attitude is associated with one or more reasons, which includes previous attitudes from which the current attitude was derived and the "inference rule" (such as policy) that was used for derivation. So in the above example, the reason for adding Plan X as an option was the policy whose action was to consider this plan plus the fact that the condition for applying the policy, i.e. the need to break the tie.

Given these basic concepts, here is how a decision is made in Doyle's system. Let's pursue the same example: we are trying to find out the appropriate strategy to apply at this point. The first step in decision making is to create a decision record, which represents the decision intention (i.e. the intention to make a decision) to be worked on. A deliberation record has several parts: a purpose, a list of options, a list of considerations, a list of reasons, a list of reflections, and an outcome. The purpose of deliberation record is the intention to make the decision, i.e. about choosing the optimal programming language.

Once the record is created, policies are searched for the active intention. Suppose there is the following policy relevant to the decision intention: "If the aim of the intention is to make decisions about which language to use for implementing the current version of Xanadu, then add the following options (C, Lisp) to the list of options." A retrieved policy is not executed at this point, but only (the intention of applying it is) added to the list of Considerations at this point, whose purpose is to keep track of the items to be examined later. Another relevant policy that may be added to the list is "If the aim of the intention is to make decisions about which language to use for implementing the current version of Xanadu and if it is important that Xanadu be portable, then add a pro-claim to the option "Use C."

Next, reflect on how to proceed. Create a second order deliberation procedure to consider the problem of how to proceed with the original decision. All the policies relevant to the second order decision are retrieved, and executed one by one rather than simply being added to the list of considerations as in the case of the original decision. That is why the deliberation does not lead to infinite recursion. The second order options include: Delay, Abort, Halt by choosing the current best option, Continue, or Reformulate. In our case, suppose the second order option chosen is Continue.

There are two policies in the list of considerations now, and some other policies determine which one to execute first. In our case, the first one about adding options get executed, and then the second one of adding a pro-argument. The actions for both of them are primitive, so can be executed directly. If the action of a policy were a plan, applying the policy would involve adding new desires, intentions, options, or reasons as dictated by the plan. If there are new reasons that have been added, reflect on them in the sense of retrieving all the policies relevant to the new reason and the purpose, and then carries out each of these new considerations the same way. The process of reflecting on new reasons continues until no more reason-relevant policies can be found. This process is guaranteed to terminate. At this point, the system repeats the step of reflecting on how to proceed and applying policies, until the second-order decision is made to halt the deliberation.

There are much more detail in [Doyle 1980] than what is briefly presented above. Doyle discusses the representation (Structured Description Language) he uses for the model, the reason maintenance system that keeps track of the dependencies among reasons, as well as more detailed presentation of the distinctions such as plan, desire, and intentions.

Doyle's model certainly captures much of the qualitative aspects of decision making processes. The notion of goal, missing in many models as we saw above, is not only explicit but also is there in finer distinctions: desires and intentions. (The distinction between desire and intention has to be made in computation context because as we saw they are orthogonal concepts. But, I am not sure if the distinction is needed in human support context.) Subgoals are represented by Subordinate relation. However, there is no explicit discussion of the relations that hold among goals such as conflicting and concording, or among subgoals such as conjunctive, disjunctive, completely specifying, and partially specifying. It is not difficult to put these distinctions in Doyle's model, and it seems worth doing it. Alternatives are expressed as Options in a Decision Record. Arguments are represented by Pro and Con Reasons on Options or other Reasons. There is no distinction between facts, claims, and opinions -- I believe for the same reason that there is no such distinction in DRL. Questions and answers have no place in Doyle's model. Meta-comments, however, can be expressed as beliefs, because beliefs can be about anything not just about the evaluation of an alternative as in DRL. Doyle's model does not provide a way to deal with the different degrees of importance, uncertainty, plausibility and how they can be combined.

Doyle's model is a very ambitious one. It wants to represent every single step in reasoning explicitly. This ambitious goal is both the strength and weakness of his model. The explicit representation allows the system to reason about the steps in its own reasoning and retract them if necessary. However, it is a very expensive process in both the human and the computational contexts. People will find it too demanding to represent their decision processes in the way that the model requires. Computationally, the model would be intractable.

I would like to view DRL as a language that is compatible with Doyle's model but is non-committal about some of its aspects. Thus, the DRL model is not as explicit as Doyle's, but it then does not have to do things, e.g. making assertions, the way prescribed by Doyle's. What DRL requires is only that claims are produced and linked to other claims in some meaningful way. The claims could have been produced in any way you want, like in Doyle's model or not. Also, one may decide, for computational tractability, not to keep justifications for every step in reasoning. In that sense, DRL can be viewed as a higher level interface to something like Doyle's model, but not bound to it.

DRL is less ambitious than Doyle's. Doyle's is a model for deliberation, action, and introspection. DRL provides a model for decision making. To the extent that deliberation, action, and introspection can be all viewed as kinds of decision making, DRL may be viewed as ambitious. But the scope of DRL, at least for now, is a subset of such decision making at large -- namely, those decisions that involve evaluating well-defined alternatives against a set of goals. That restriction in scope let DRL provide higher level constructs such as the FACILITATES relations, which allow more modular representation of the arguments evaluating alternatives as well as allow such relations themselves to be argued about. This comparison is only a preliminary and the exact relation between Doyle's model and DRL has to be worked out.

4.6 gIBIS

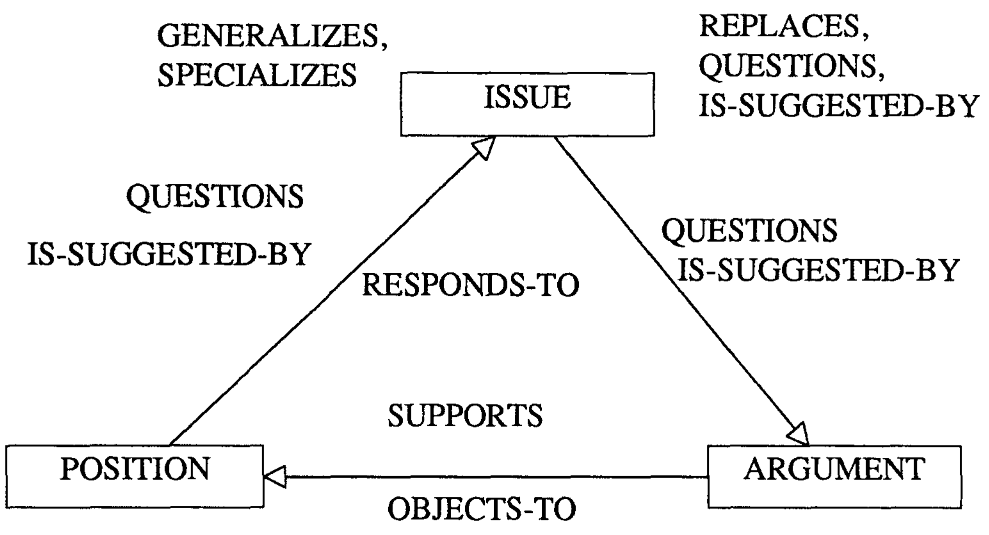
gIBIS (graphical Issue Based Information System) is a "hypertext tool for exploratory policy discussion" that is being developed at MCC [Conklin & Begeman 88}. The goal of gIBIS is to capture the design rationale: the design problems, alternative resolutions, tradeoff analysis among these alternatives, and the tentative and firm commitments that were made in the process of the decision making process. gIBIS is discussed here because its goal is similar to that of DRL and because it is a hypertext-based system that is designed to support human making decisions. The difference between DRL and gIBIS comes from the structures provided in achieving this goal and the underlying philosophy behind the structures.

gIBIS attempts to capture qualitative aspects of decision making with the following constructs (Cf. Figure 6). There are three types of objects (Issue, Position, Argument) and nine kinds of links (e.g. Responds-To, Supports, Objects-To, Generalize, Specialize, Question, Be-Suggested-By). In our example, ISSUE is "Which programming language should we use for implementing the current version of Xanadu?", POSITIONs are the different languages, e.g. "Use Lisp." An ARGUMENT Supporting this Position would be "Lisp provides high level features." An ARGUMENT Objecting-To the last ARGUMENT would be "But Lisp is not efficient."

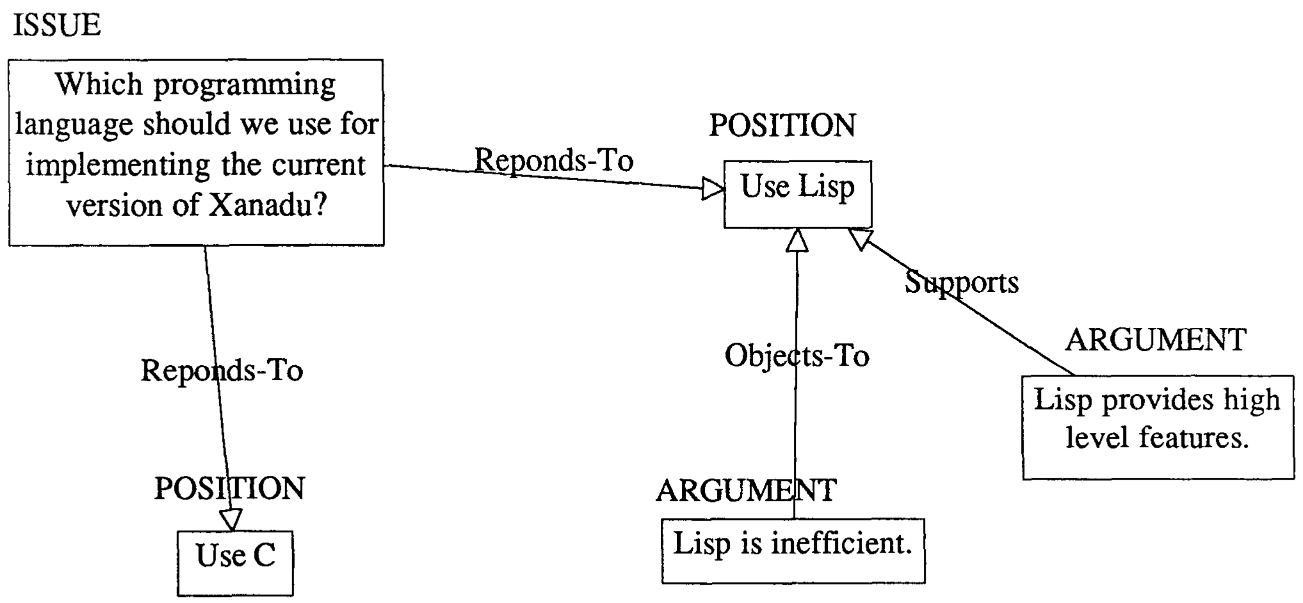
Notably lacking in gIBIS, however, is again the notion of goals. Of course, goals cannot be absent in decision making. In gIBIS, they appear implicitly in Arguments. For example, when one argues that Lisp is not a good choice because it is not efficient, an implicit goal is to minimize the development or the run time. Again, all the reasons for making goals explicit, discussed in the context of analytic decision theory, still hold here. The explicit representation of goals: allows modular representation of arguments, forces people to articulate the goals and their subgoals,

Fig. 6 The gIBIS Model and an Example

(a) The Model



(b) An Example


ISSUE
Which programming language should we use for implementing the current version of Xanadu?
Reponds-To
POSITION
Use Lisp
Supports
ARGUMENT
Lisp provides high level features.
Objects-To
ARGUMENT
Lisp is inefficient.
Reponds-To
POSITION
Use C

lets people argue about them, provides a basis for precedent management as well as multiple viewpoints.

Another major difference between gIBIS and DRL is that, in DRL, relations are objects, in particular subclasses of CLAIM, so that they can be argued about or questioned in the same way as any other claims. In gIBIS, there is no way to comment on relations themselves. Also, gIBIS is only a static representation structure; there are no services such as dependency or viewpoint, or precedent managements envisioned as in the DRL environment.

5. Future Research

The semantics for DRL needs more work, as mentioned. Doing so will clarify its relationship with other computational paradigms like production rules or prolog.

The operations need to be defined that will serve as interfaces to each set of services that I discussed in Section 3.

I would like to identify and classify the different types of dependency relations that can hold among the objects of DRL. Goals, for example, are related in ways more complex than through IS-A-SUBGOAL-OF. They could be completely or partially conflicting; they could be supportive without one being subsumed by the other; or there could be tradeoffs between them. I need to work out what the possible relations are, and how they should affect the way that evaluation get merged and propagated. It is also important to identify the different kinds of goals and how they are related, if the system is to provide precedent management. There are some works on goal taxonomy (e.g. preservation goal, achievement goal, delta goal. Cf. Schank, Wilensky). But we need to articulate more fine-grained categories of goals. Another area that I want to study is requirements analysis (e.g. Greenspan). I suspect that in specifying the requirements, much of the same issue comes up and that some work has been done on it.

I would like to find a more formal basis for describing and comparing the models discussed in Section 4. The above comparison was given by first intuitively characterizing the things that we want to represent and showing how they were or were not represented in each model It would be nicer if I can make the basis for comparison more formal or systematic.

I am also aware of some models that have not been covered in this report. One is Konolige's recent work [Konolige 89] on 'defeasible arguementation system'. Another is Jarkes' MEDIATOR [Jarkes 88, Hahn 88]. Konolige's work is interesting because it is an attempt to formalize argumentation in terms of non-monotonic logic and studying it would provide me with better understanding of the semantics of DRL. MEDIATOR is interesting because it views collaborative decisions as "complex, evolving data objects to be managed by an extended database management system". Decision problems are represented as "multispatial mappings that eventually lead from real-world observations to group utilities. Group decision processes are represented as nested transactions which collaboratively design this mapping structure, controlled by group communication norms represented as integrity constraints." MEDIATOR would be also interesting to study because it uses CML (Conceptual Modeling Language). CML is a descendant of RML (Requirements Modeling Language) used for requirement analysis, which is an area that I indicated in another report as interesting to study.

Appendix 1. A Scenario Illustrating the Motivations

Below, I present scenarios that illustrate the problems that motivate this research on DRL.

Learning from Precedents

As the head of a new project FOO, John is responsible for selecting the best knowledge representation (KR) language for Project FOO. John knows that the project BAR at ABC Inc. is similar to FOO; so he calls ABC. After many tries, John finally gets to talk to some members of the project BAR and ask them what KR language that they had chosen and why. Unfortunately, most members who had been involved in deciding the KR language for BAR are not around anymore, and others do not remember all the issues and the alternatives that had been considered. How could one transfer the knowledge cumulated at ABC to John's company? What is the best way to describe the decision processes that went on? How can one extract the relevant issues from them?

John was told that, in fact, ABC's choice of KRL was a bad one insofar as minimizing development time was concerned because it did not have a good version mechanism. It occurs to John that it would be nice if one could represent and transfer to the current problem context the experiences after the decision as well as the considerations given before the decision.

Distributive Group Decision Support

John calls several meetings of his project members to discuss the issues that arise in choosing the best KR language. However, meetings are not always the best way to discuss the issues. There are usually some people who cannot come to the meeting for various reasons but whose inputs are necessary. Also, John wants some way of allowing people to raise additional issues or arguments whenever they want to.

Dependency Management

During meetings, it becomes apparent that the choice of KR language depends on other choices. On which of the machines available-- Sun, microVax, Mac2-- should the language run and under which programming language? The choice of one would restrict, but not completely determine, the range of other choices. For example, if one chooses STROBE, the programming language has to be either Lisp or C and the hardware cannot be Mac 2. On the other hand, if one chooses LOOPS, then the programming language has to be Interlisp and the hardware has to be Sun. John finds that the choices of hardware and the programming language themselves merit discussions. For example, portability and reliability of the hardware can be important issues. John wants to represent the dependency relations in such a way that when one choice is made, he need not consider all the other choices incompatible with it.

Not only does a choice depend on other choices, but also the validity of arguments depend on other arguments or choices. For example, Lisp would be a bad choice as a programming language if it were to run on microVax. But that claim would be relevant only if microVax is chosen as the hardware and VaxLisp as the programming language. John wants some way of representing all these dependency relations among choices as well as among claims in such a way that when one changes, the others dependent on it can get updated in a consistent way.

Viewpoint Management

As the issues, arguments, and various constraints are brought forth, John feels the need to view them in different ways. He wants to evaluate KR languages with different weights on the different constraints; for example, when portability is very important vs. when it is not or when Sun is chosen as the hardware vs. microVax. John wants to be able to switch back and forth between these different perspectives and compare them as well.

Clearer Representation

John has called several meetings to talk about the issues to consider in choosing the right KR language. Notes are kept at every meeting, but John finds the notes unsatisfactory. The textual notes are too linear to show the complex interrelationship among the issues that have been raised, the arguments put forth, and the eventual evaluation of the alternatives. John feels that there must be a better way to represent all that has been discussed so that one can see more clearly their overall structure.

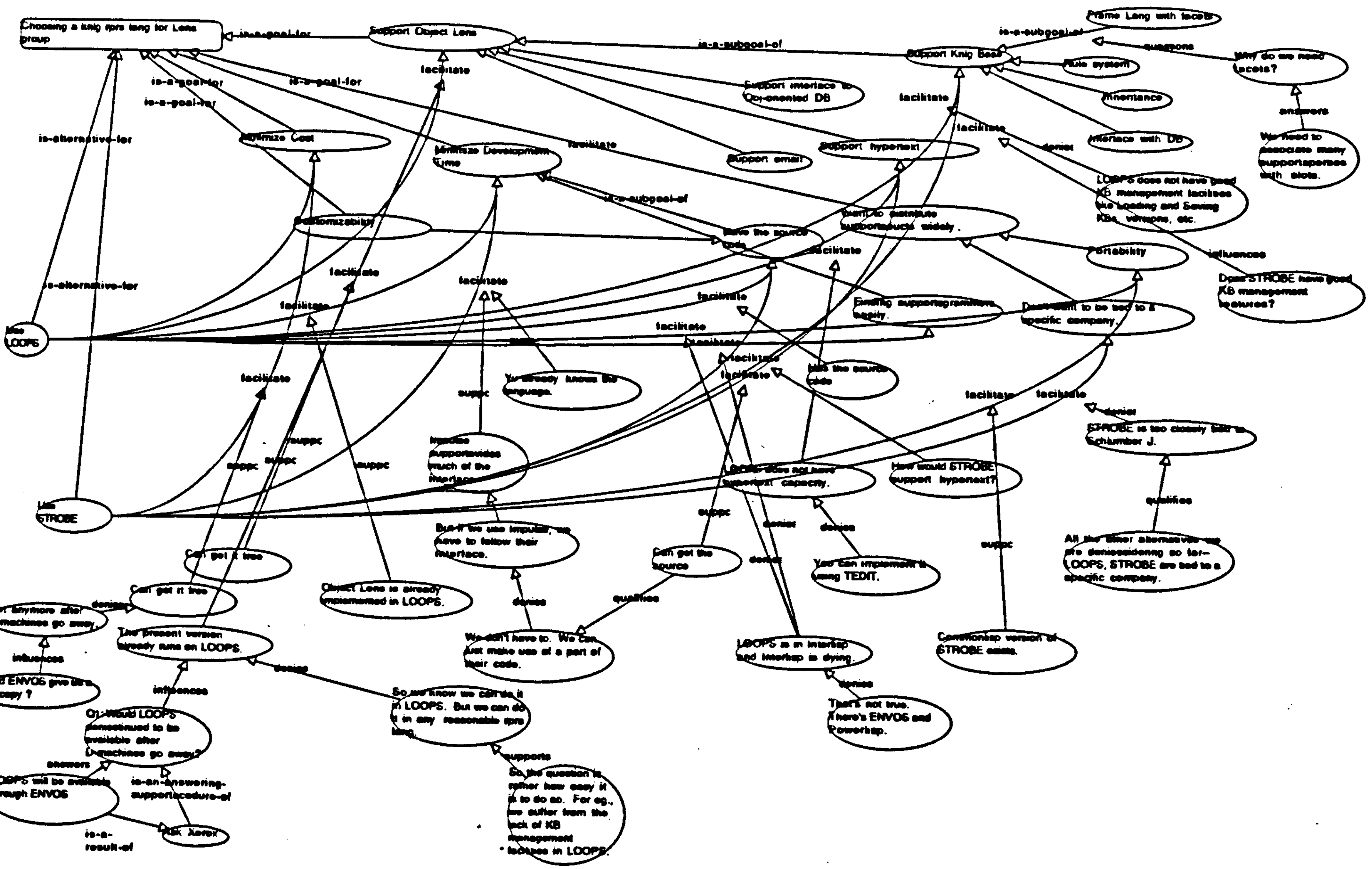
Cumulative Progress

John also feels no sense of cumulative progress from meeting to meeting. Again, the meeting notes kept provide some record of what has been discussed. But they do not provide a sense of continuity. He believes that they need a better way of representing the decision process so that the next meeting can start from where the previous one left off.

Decision Audit

Several months after John made the decision, his superior calls him and asks him why he made the decision the way he did. John collects all the notes that have been kept from the meetings and tries to reconstruct the decision processes. But he finds it not only time-consuming but also difficult to remember all the details.

Figure. An Example Decision Graph: Selecting a Knowledge Representation Language


Choosing a knlg rprs lang for Lens group
is-a-goal-for
Support Object Lens
is-a-subgoal-of
Frame Lang with facets
is-a-subgoal-of
questions
Why do we need facets?
answers
We need to associate many supportsperses with slots.
Support Knlg Base
Rule system
Inheritance
Interface with DB
Support interface to Obj-oriented DB
facilitate
facilitate
denies
LOOPS does not have good KB management facilities like Loading and Saving KBs, versions, etc.
influences
Does STROBE have good KB management features?
Support email
Support hypertext
Want to distribute supportsducts widely.
Portability
Does want to be tied to a specific company.
Finding supportsgrammers easily.
is-a-goal-for
is-a-goal-for
is-a-goal-for
is-alternative-for
Minimize Cost
Minimize Development Time
is-a-subgoal-of
Customizability
Have the source code
facilitate
is-alternative-for
Use LOOPS
facilitate
facilitate
facilitate
facilitate
facilitate
facilitate
facilitate
Has the source code
You already know the language.
suppc
Impulse supportsvides much of the interface
suppc
suppc
suppc
suppc
Use STROBE
LOOPS does not have hypertext capacity.
How would STROBE support hypertext?
facilitate
facilitate
denies
STROBE is too closely tied to Schlumber J.
qualifies
All the other alternatives we are considering so far—LOOPS, STROBE are tied to a specific company.
suppc
Commonlisp version of STROBE exists.
denies
You can implement it using TEDIT.
denies
LOOPS is in Interlisp and Interlisp is dying.
denies
That's not true. There's ENVOS and Powerlisp.
suppc
Can get the source
qualifies
But if we use Impulse, we have to follow their interface.
denies
We don't have to. We can just make use of a part of their code.
Can get it free
Object Lens is already implemented in LOOPS.
Can get it free
denies
Not anymore after D-machines go away.
influences
Would ENVOS give us a free copy ?
The present version already runs on LOOPS.
denies
So we know we can do it in LOOPS. But we can do it in any reasonable rprs lang.
supports
So the question is rather how easy it is to do so. For eg., we suffer from the lack of KB management facilities in LOOPS.
influences
Q1: Would LOOPS be continued to be available after D-machines go away?
answers
LOOPS will be available through ENVOS
is-an-answering-supportsedure-of
Ask Xerox
is-a-result-of

Appendix 3. Operations

User Interface Operations

The following are the operations available to the end user. They are organized object-oriented way. For the end user, this also means that only those operations for a given object (and its parents) will be accessible from the object. For example, when the user clicks on an object, only those operations for that object will be presented for the user as an option. From the system viewpoint, these operations for an object form the methods for the object.

*** for any object

**** Edit Object
**** Delete Object

*** for Decision Problem

**** Create An Instance (Class operation)
**** Add Goals
**** Add Alternatives
**** Show Matrix
Shows a matrix where the columns represent goals, the rows represent alternatives, and each cell represents an overall evaluation measure so far.
**** Evaluate
Updates the evaluation by propagating and calculating the impact of newly added claims, goals, or alternatives.

*** for Evaluation

**** Show Arguments
Shows the claims that are responsible for the evaluation
**** Specify Evaluation

Updates the evaluation measure for the evaluation

*** for Set

**** Create an Instance (Class)
**** Specify Members
**** Specify the Relationship

*** for Goal

**** Create an Instance (Class)
**** Specify the Importance
**** Create a Subgoal
**** Show Subgoals

*** for Alternative

**** Create an Instance (Class)
**** Specify the Seriousness
**** Show Sub-Alternatives

*** for Claims

**** Create an Instance (Class)
**** Add a Pro Claim
**** Add a Con Claim
**** Add a Supporting Claim
**** Add a Refuting Claim
**** Add a Qualfying Claim
**** Add a Question
**** Add an Influencing Question
**** Specify Plausibility
**** Propagate Plausibility

Updates the plausibilities and the evaluation mesaures

*** for Question

**** Create an Instance (Class)
**** Add a Question
**** Create an Answering Procedure
**** Connect to an Answering Claim
**** Create a Set of Possible Outcomes

*** for Viewpoint

**** Create an Instance (Class)
**** Create an Related Viewpoint
**** Relate to another Viewpoint
**** Display Sub-Viewpoints

*** for Procedure

**** Create an Instance (Class)
**** Specify Sub-Procedures (Class)
**** Specify Component Procedures

System Operations

*** Retrieve what's been represented selectively (ASK)
and possibly transform them in such a way

**** Explicitly Represented Information:
;something that a simple query language would do)
;something that requires only filtering)
e.g. show me all the claims that refute this claim

**** Implicitly Represented Information (Derived Information):
;something that requires transformation)
***** e.g. Evaluate Alternatives
***** extracting relevant precedents
****** navigating the goal lattice
***** extracting relevant parts from a given precedent

*** Selectively change what's been represented (TELL)
**** explicitly, in which case we need consistency check
**** automatically, representing and mainataining dependency relation.
***** Propagate Plausibility

*** Display

**** e.g. multiple viewpoints

<u>**References:**</u>

[Birnbaum et. al. 80] Birnbaum, L., Flowers, M., and McGuire, R. Towards an AI model of Argumentation. In Proc. AAAI, Stanford, CA. 313-315. 1980

[Conklin & Begeman 88] Conklin, J. & Begeman, . gIBIS: A Hypertext Tool for Exploratory Policy Discussion. Proc. Computer Supported Cooperative Work. 1988

[Doyle 80] Doyle, J. A Model for Deliberation, Action, and Introspection. MIT AI-TR 581. Cambridge, MA 1980

[Fahrenbach 73] Fahrenbach, H. Wirklichkeit und Reflexion. Festschrift fur Walter Shulz. Pfullingen, Gunther Neske. 1973

[Habermas 73] Habermas, J. Wahrheitstheorien in [Fahrenbach 73] 1973

[Hahn 88] Hahn, U. & Jarke, M. A Multi-Agent Reasoning Model for Negotiation Support. MIP-8804, Fakultat fur Mathematik und Informatik. Universitat Passau. 1988

[Huth 75] Huth, L. Argumentationstheorie und Textanalyse Der Deutschunterricht, Jhrg. 27, pp.80-111. 1975

[Jarke 88b] Jarke, M. The Design of A Database for Multiperson Decision Support. MIP- 8812, Fakultat fur Mathematik und Informatik. Universitat Passau. 1988

[Konolige 89] Konolige, K. & Pollack, M. Ascribing Plans to Agents. Preliminary Report. Proc. IJCAI-89 Detroit, MI

[Lowe 86] Lowe, D. SYNVIEW: The design of a system for cooperative structuring of information. Proc. Computer Supported Cooperative Work. 1986

[Pratt 70] Pratt, J. The appropriateness of a Toulmin analysis of legal argumentation. Speaker and Gavel, v.7 1970

[Schank & Abelson 77] Schank, R. & Abelson, R. Scripts, Plans, Goals, and Understanding. Lawrence Earlbaum Associates. New York, NY. 1977

[Toulmin 69] Toulmin, S. The Uses of Argument. Cambridge, Cambridge Univ. Press.

[Wilensky] Wilensky, R. Planning and Understanding. UC Berkeley, CA. 1986

[Wunderlich 74] Wunderlich, D. Grundlagen der Linguistik. Reinbek bei Hamburg, Rowohlt Taschenbuch Verlag 1974