CMU-CS-83-116

The QBKG System: Knowledge Representation for Producing and Explaining Judgements[1]

David H. Ackley and Hans J. Berliner

ABSTRACT

The QBKG system plays backgammon and produces critical analyses of possible moves for a wide variety of backgammon positions, using a hierarchically structured, non-discrete form of knowledge representation. The largely non-searching control structure emphasizes *judgemental* processes at the expense of *reasoning* processes, meaning that the system's behavior is determined by the estimated usefulness of its immediate actions rather than upon hypothesized longer-term results such as would be produced by a tree-searching algorithm. This report describes some of the principles by which knowledge can be represented so as to facilitate high-quality judgements in a domain, discusses issues arising from the need to be able to explain how a particular judgement was reached, and argues that sophisticated judgemental ability is a critical feature for systems operating in complex, incompletely understood environments.

[1]This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-81-K-1539.

Table of Contents

1. Introduction 1
  1.1. Background 1
  1.2. Generating explanations 1
  1.3. Non-discrete knowledge representation 2
  1.4. Reasoning and Judgement 3
  1.5. Structure of the paper 4
2. The structure of the system 5
  2.1. Generalized model of QBKG 5
  2.2. The QBKG implementation 9
3. Constructing evaluation functions 11
  3.1. The task of an evaluation function 11
    3.1.1. Linear and non-linear evaluation functions 11
    3.1.2. Heuristic knowledge 12
  3.2. Representing collections of heuristics in an evaluation function 13
    3.2.1. Representing the limits of linearity 15
  3.3. A sample evaluation function 17
    3.3.1. Evaluation surfaces and hill climbing 18
  3.4. Boundary cases and the blemish effect 22
  3.5. Language description 25
4. Generating explanations 29
  4.1. Finding relevant issues 29
  4.2. Qualifying differences in magnitude 30
    4.2.1. Judging the absolute status of the game 32
    4.2.2. Judging the relative overall quality of the two moves 32
    4.2.3. Judging the magnitude of a difference in a component of a SUM node 33
    4.2.4. Judging the magnitude of the difference in a collection 34
  4.3. Some examples 35
5. A wider view 39
  5.1. Reasoning and Judgement 39
  5.2. A hybrid organization 42
  5.3. The signal-to-symbol transformation 44
6. Limitations and extensions, summary and conclusion 46
  6.1. Weaknesses and limitations 47
    6.1.1. The QBKG system 47
    6.1.2. The general model 49
  6.2. For further research 50
    6.2.1. The quality of a judgement 50
    6.2.2. Learning to make judgements 51
    6.2.3. Network organization 52
    6.2.4. Architectures for hybrid systems 53
  6.3. Summary of main points 55

6.4. QBKG and the tradeoff of reasoning and judgement 56

List of Figures

Figure 2-1: Overview of the functional organization of QBKG 6
Figure 3-1: Schematic view of portions of the QBKG Judgement Structure. 14
Figure 3-2: Method of combining heuristic knowledge 14
Figure 3-3: Projected view after purchasing three oranges 19
Figure 3-4: Projected view after purchasing eight oranges 19
Figure 3-5: Landscape modelling the non-linearity in orange prices 20
Figure 3-6: Landscape corresponding to Equations 1 21
Figure 3-7: Landscape corresponding to Equations 2 21
Figure 3-8: Landscape corresponding to Equations 3 22
Figure 3-9: Sample concept definitions 25
Figure 4-1: Determining relative goodness of a comparison move 33
Figure 4-2: Determining significant differences in components of SUMs 34
Figure 4-3: Determining significant differences in collections 34
Figure 4-4: Sample position, White to play 6,1 36
Figure 4-5: QBKG's commentary on two moves in Figure 4-4. 36
Figure 4-6: White to play 6,5 37
Figure 4-7: QBKG's commentary on two moves in Figure 4-6 37
Figure 5-1: The Big Elephant Argument 43

List of Tables

Table 1-1: Comparison of reasoning and judgement 4
Table 3-1: Properties of QBKG concepts 26
Table 3-2: Language Operators 27

1. Introduction

The origin of the QBKG system from the BKG backgammon program is explained. The issues involved in generating explanations of behavior based on a non-discrete knowledge representation are introduced. A characterization of the notions of "reasoning" and "judgement" as being two fundamentally different methods of producing behavior is presented. The organization of the report is sketched.

1.1. Background

In 1979, the backgammon program BKG 9.8 [3] won an exhibition match in Monte Carlo against the then-world-champion Luigi Villa by a score of 7-1. Although analysis of the match makes it clear that BKG had some good luck with the dice, and made several small errors, there is little doubt that the program is a very good backgammon player.

The BKG program does not rely primarily upon tree searching to select a move, but upon a large base of backgammon knowledge coded in the form of an evaluation function. This function is applied to the positions resulting from each legal move in a given position to determine the best move to make, resulting in a fixed-depth, 1 ply search with terminal evaluation. During the course of the development of the evaluation function, several key ideas emerged as central to the success of the endeavor. Collectively, these ideas were referred to as the SNAC method of evaluation, standing for Smoothness, Nonlinearity, with Application Coefficients. [3]

The goal of the BKG project was to develop a fast program with a very high level of competence. To achieve speed, the evaluation function was written in a compiled language and in such a manner as to avoid expensive or unnecessary constructions wherever possible. The original goal of the QBKG[2] project, which was begun in 1980, was to augment the BKG system with an explanation-generating component, to allow it to defend its choice of moves and comment on suggested moves.

1.2. Generating explanations

It has become apparent that in order for a computer program to be considered genuinely expert in some field, it is insufficient for it to only be able to make expert-quality decisions. The expert program must in addition be able to justify its decisions in a manner that (at least) other experts in the field can understand. The reasons for this requirement are discussed in [5] with respect to the MYCIN medical diagnosis system, and are quite generally applicable.

[2]Query-BKG

The MYCIN work focused primarily on two sorts of justification: explaining why any particular bit of information is needed in the process of a diagnosis, and explaining how any particular conclusion was reached. Since MYCIN's control structure is a depth-first search through an AND/OR tree, answering the first question amounts to moving up the tree toward the root and explaining what the requisite information was a subgoal of, and answering the second question amounts to moving down the tree and explaining what subgoals were used in satisfying a particular rule. This intuitively-pleasing scheme was also used in [21] in the question-answering component of the SHRDLU blocks world program.

Since QBKG does not employ a search procedure of this sort, it is not immediately clear how an analogous question-answering method might be devised for such a system. In order for QBKG to perform competently, however, the information which can be gained by tree searching must somehow be represented in QBKG's evaluation function. The move to make, then, is to replace analysis of the search tree with *analysis of the evaluation function.* In order to do this, the system needs to have access to the various components of the evaluation function that combine to form the overall judgement. Whereas in BKG only the resulting *heuristic value* of a possible move was relevant, in QBKG the internal structure that computes the heuristic value is relevant as well. Given that internal structure, some means must be devised for expressing essentially continuous data in the discrete, symbolic fashion of natural language. Furthermore, in order for the generated explanations to be comprehensible, the internal structure of the evaluation function must correlate in certain direct senses with the structure of backgammon knowledge as humans comprehend it. These constraints led to the development of a set of guidelines for constructing evaluation functions, and, implicitly, a scheme for knowledge representation and use which is quite different in appearance and structure from most schemes that have been explored in AI research.

1.3. Non-discrete knowledge representation

Throughout this report we argue the dangers of discrete and extoll the virtues of non-discrete knowledge representation, so it is important to make clear what we mean by the terms and to set the context for what we are trying to say. The basic distinction is between *few-valued* and *many-valued* representations, with two-valued boolean logic at one extreme and uncountably infinite-valued differential equations at the other. By a non-discrete representation of a proposition, say, we mean allowing for a large set (twenty or fifty or thousands, depending on the situation) of possible degrees of confidence or belief in the proposition. Thus a statement about chess positions such as "x is an endgame" would be asserted with many differing degrees of confidence depending on how well each particular x fits the known constraints on being an endgame.

When we speak of "continuous" values we mean precisely this sort of many-valued representation. We do not mean to suggest that there is something going on in mathematically continuous functions that cannot be adequately modeled in a finite-valued system. Nor do we mean to suggest that discrete symbol manipulation in a two-valued system is by any means useless. When a distribution of continuous values is largely bimodal, as many are, making the assumption of two values affords a great simplification in the problem space and allows greater progress with a given amount of resources. In particular, solving the credit assignment problem to perform learning is greatly facilitated by discretizing the problem space and hypothesizing cause-and-effect relationships between the resulting states, *provided* that the discretization is performed in a manner appropriate to the domain and the goals of the problem-solving system.

What we will stress is that determining such appropriate discretizations is an inescapable, non-trivial task for intelligent systems operating in incompletely understood, complex domains, a task which must be performed in a domain-and-goal-dependent manner if discrete reasoning is to provide useful results. Furthermore, we will emphasize the serious limitation of few-valued reasoning: the occurrence of "boundary cases", where the few-value assumption leads reasoning astray. We will show that this cannot be avoided when a system does not completely understand its domain, and must be handled with a many-valued representation.

1.4. Reasoning and Judgement

One of the outgrowths of this work has been the realization that the everyday notions of reasoning and judgement find apt expression within this structure. This is discussed at some length in Section 5, but it is worth taking a moment here to describe what we mean by the two terms.

In very general terms, *reasoning*, as we see it, is the process of *imagining the environment to be other than it is.* The *sine qua non* of reasoning is the existence of an internal world model which can be perturbed to perform this imagining. By this definition, any sort of search in a problem space is reasoning, whether it be tightly constrained knowledge-intensive best-first search or a random walk. The process by which the QBKG system tries out possible moves on an internal model of the backgammon board is a reasoning process.

Judgement, in similarly general terms, we see as the process of *forming an interpretation of the environment with respect to a goal.* A thermostat which classifies sensed temperature in terms of "too hot", "too cold" or "OK" is making simple judgements. The fundamental characteristic of judgement is that it is purposive — that its results relate to some goal — whereas reasoning, in the abstract, may not be. The QBKG system possesses an evaluation function which interprets its environment, backgammon positions, with respect to the goal of winning the game.

By this characterization, it should be clear that our position will be that all problem solving systems are to be viewed as performing both reasoning and judgement in varying degrees and combinations. Search strategies are methods of using judgement to control reasoning. In best-first searching, for example, large portions of the problem space may be ignored because it is judged that the best value obtainable in those portions is not as good as a value already at hand.

It is not our primary purpose to defend these uses of the terms here, but to give a flavor for the distinction that will be made throughout this report. Table 1-1 summarizes some of the distinguishing aspects of reasoning and judgement as we see them.

Reasoning is	**Judgement is**
Qualitative	Quantitative
- few-valued, discrete	- many-valued, continuous
- response either unchanged or drastically changed by small differences	- graded response to small differences
Causal	Correlative
- therefore, ultimately	- therefore, potentially
Sequential	Parallel

Table 1-1: Comparison of reasoning and judgement

1.5. Structure of the paper

This report describes the knowledge representation and control system that emerged from the QBKG project. Section 2 presents a generalized model of the structure employed in QBKG and discusses the domain of backgammon with respect to the model. Section 3 discusses some considerations involved in designing evaluation functions to allow for large amounts of knowledge and avoid several pitfalls that arise. Many of these concerns are applicable to evaluation functions designed for more general tree-searching models as well as the QBKG-type shallow search hill-climbing models. Section 4 describes the algorithms for interpreting the evaluation function to generate explanations. Section 5 steps back somewhat from the QBKG system and concerns itself with the nature of reasoning and judgement and considers issues arising from the making of such a distinction. Finally, Section 6 summarizes this report, discusses some limitations of the system and the model, and speculates about directions in which the work could be extended.

This report may be slightly unusual because of its rather uniformly "vertical" organization — it attempts to be, in turns, a report in some detail on a specific system, a short treatment of some important concepts and principles for designing knowledge manipulation systems, and a brief high-level discussion of more "philosophical" concerns stemming from the research. We have taken this route, rather than producing two or three separate reports or relegating the higher-level concerns to a paragraph or two in a "Discussion" section, to try to make explicit a particular *point of view* from which the work can be interpreted; a point of view, still being developed, which we have come to over the course of time and believe to be a useful vantage point for understanding system organizations and behavior.

2. The structure of the system

The representation and control system of QBKG is generalized and the domain of backgammon is discussed with respect to the generalization. The implementation of the QBKG system is sketched as an example of the method and to provide a framework for more detailed discussions.

2.1. Generalized model of QBKG

The organizational structure of the QBKG system can be readily generalized as having a few basic functional parts as listed below. The purpose of this generalization is two-fold. First, it highlights certain difficult problems that the QBKG system does not have to face due to the nature of the domain of backgammon. Moreover, it provides a high-level breakdown of the task faced by information-processing organisms in general and thus provides a framework for comparing structures. We make no claim that this breakdown is *the* model by which all systems should be organized, nor that such a *functional* description should necessarily imply an analogous *architectural* description.

1. The *Real World*, from which measurements can be taken and upon which actions can be effected. We explicitly include the Real World as a component of the system to emphasize the fact that the particular demands of any given Real World heavily impact system design trade-offs, and that no system can be discussed effectively except with respect to a Real World (or class of Real Worlds) in which it can function.

2. The *World Model*, from which estimates of results of actions on the Real World can be obtained. The World Model must attempt to provide answers to questions of the form "In the current situation, if I do x, then what will my measurements of the Real World be?" Having a World Model in the system is the fundamental prerequisite for allowing any reasoning process to exist.

3. The *Judgement Structure*, which contains the system's beliefs as to how measurements of the Real World correlate with the goals of the system. By correlating measurements with goals we gain the ability to hypothesize various possible actions using the World Model and compare the usefulness of their expected results. The Judgement Structure provides the system with a means of estimating future consequences of situations with respect to the system's goals without performing a search.

4. The *Action Set*, which is a description of the possible actions of the system, and which can be either *hypothesized* (performed on the World Model) or *actualized* (performed on the Real World). In general, it is assumed that the contents of the Action Set must be derived anew from the World Model (or the Real World) after each action is taken.

5. The *Reasoning System*, which implements the hypothesize-act cycle, and is responsible for updating the World Model as actions are actualized. The reasoning system operates in the following fashion:

   a. Hypothesize each of a selection of actions from the current action set and apply the Judgement Structure to the estimations produced by the world model.

   b. Actualize the action corresponding to the highest evaluation produced, and update the World Model.

   This particular control structure is closely coupled to the actual organization in the QBKG system, and there are many others that could be fit in without changing the rest of the model, such as alpha-beta search or branch-and-bound.

6. The *Analysis System*, which interprets the results of the Judgement Structure and the Reasoning System. The analysis system is used to generate explanations of behavior and would also play a central role in guiding system learning by noticing when the estimations produced by the World Model fail to match the observed results (World Model in error), or when favorably judged situations too frequently lead to undesirable consequences (Judgement Structure in error).[3]

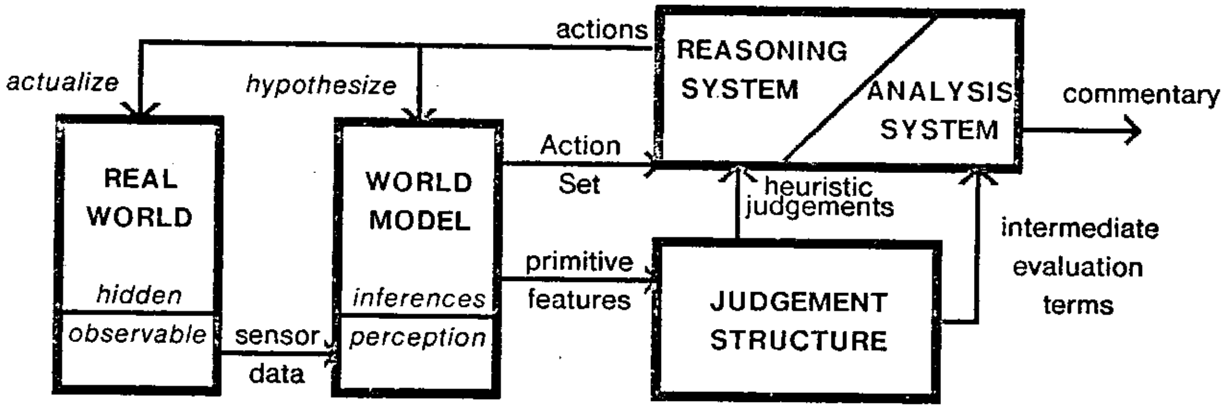

Figure 2-1: Overview of the functional organization of QBKG

Figure 2-1 shows how these functions interact in the QBKG organization. The reader should not imagine

[3]A major weapon in the practical joker's armament is to systematically cause World Model errors in other people. An invitation to appreciate the scent of a water-squirting fake flower, for example, trades on the victim's World Model erroneously assuming the flower is real. If the victim *realized* that the flower was a trick and inexplicably moved within range anyway, one might suspect an error in the Judgement Structure.

that Figure 2-1 is supposed to encompass all information-processing organizations. At the very least, systems significantly more powerful than QBKG would probably have more data paths than are shown in the diagram. For example, a learning system would have a path from the Reasoning System to the Judgement Structure, among others.

There are two feedback loops in the overall process. The external loop passes through the Real World and corresponds to emitting behavior and observing results. The internal loop operates directly through the World Model and performs the hypothesization function which allows for exploring problem spaces without actually emitting behavior. Performing search via hypothesization in the World Model has two important advantages and one important disadvantage. On the positive side, hypothesizing actions is much faster than actually performing them, and, perhaps more importantly, states of the problem space can be judged without first having to actually reach those states in the Real World. On the negative side, of course, is the fact that the World Model is only an approximation, and a judgement of an hypothesized situation may drastically fail to agree with the judgement produced when the system actually achieves the hypothesized state in the Real World, due to errors in the World Model.[4]

The main point we wish to stress with respect to Figure 2-1 is the separation of the Reasoning System and the Judgement Structure. In some organizations, all the knowledge necessary to determine behavior is centralized and determining actions is performed, conceptually, by a single process — determining the next production to fire in a production system, for example. Having a single uniformly-structured knowledge base seems quite natural and convenient, so unless there is a pressing need for a more heterogeneous structure, organizations such as shown in the figure seem uselessly complicated.[5]

The reason for making the distinction is that, as suggested by Table 1-1 and discussed in Section 5, the natures of reasoning and judgement are more complementary than similar, and the kinds of representations and processes which are suitable for performing such functions *rapidly* appear to be quite different. To expect a single-level, uniform structure to be adept at both seems, to us, to be a mistake.[6]

The QBKG system manifests certain components of the general model more prominently than others. Most significantly, difficult questions dealing with the production of high-quality estimations by the World

[4]In the general case, as shown, not all of the Real World is assumed to be observable by the system, and the World Model must be able to make plausible inferences about the hidden aspects of the Real World.

[5]Furthermore, there is no "theoretical" benefit in separating these functions, in the sense that the QBKG organization is still only Turing-equivalent, and so is isomorphic to a production system, up to space and time requirements.

[6]Although exploring such uniform organizations can have great value in delineating what they can and cannot do, and why.

Model are largely finessed in the QBKG system, by virtue of its domain being a perfect information game. In such cases, all of the Real World is observable, so the World Model can simulate it perfectly. Applying this general structure to the play of the hand in Bridge, for example, would require a great deal more complexity in the World Model. If QBKG were extended to learning, therefore, it would only have to deal with failures of the Judgement Structure.[7]

Another place where the QBKG system gains simplicity over this organization is that, in the general case, the action set will be very large, and knowledge must be applied in order to select which actions should be hypothesized. The QBKG system does not have to face this problem because the action set is small enough that it can be exhaustively investigated. Similarly, in general the action set may consist of continuously parameterized actions (such as where to place the hand in order to catch the ball) rather than discrete ones, causing difficulties that the QBKG system does not have to face by virtue of its domain.

Balancing this simplicity provided by backgammon is the existence of chance factors represented by the roll of the dice, making extensive hypothetical reasoning infeasible due to the explosive growth of possible future states. The Judgement Structure is faced with the difficult task of computing the *expected* heuristic value of a position as best it can, given the nature of the random choices. Since searching more than a few moves into the future is impractical, systems for playing backgammon must rely heavily on the quality of the estimations produced by the Judgement Structure. The resulting Judgement Structure for backgammon is large and complex enough that the analysis system has a non-trivial job in isolating the knowledge relevant to a given question.

The structure of the QBKG system differs from most symbol-manipulation work in that it performs very little *reasoning* and a lot of *judgement* to determine its actions. The extent of the reasoning process is performed by the World Model in order to answer the "What if" questions of the hypothesization step. In general, the maintenance of an accurate World Model becomes exponentially harder the further from the Real World the hypothesizations become, due to the compounding of chances for error. The one-ply hypothesize-act cycle employed in QBKG represents a near-extremal point in the spectrum represented by the trade-off of reasoning and judgement. The degenerate case where no reasoning is performed at all leads to a stimulus-response system where the Judgement Structure, rather than producing a measure of the utility of the situation, produces a coding for an action to be taken.

The choice as to what part of the decision-making process should be delegated to reasoning processes and

[7]Although a more sophisticated World Model might, for example, make plausible inferences based on observed weaknesses in an opponent, in which case World Model learning would be necessary.

what part to judgemental processes is heavily dependent upon the particular constraints on and abilities of a system, and it is frivolous to hope that one particular optimal breakdown of responsibility exists for all possible environments and organisms. Section 5 discusses some of the general issues involved in the tradeoff.

The focus of the QBKG work, then, is in examining the issues involved in creating, exploiting, and explaining high-quality judgements in a complex domain. The domain of backgammon and the particular instantiation of the model emphasize the judgemental issues that have received little explicit attention in AI research.

2.2. The QBKG implementation

For purposes of gaining perspective and bringing the discussion down to a more detailed level (at least temporarily), it is worthwhile to look briefly at the "guts" of the QBKG program and see how it is organized. The breakdown is basically in terms of the general structure presented in the previous section, although the program was not formally designed that way when it was first constructed.

The *Real World* consists of thirty-two integers of appropriate sizes representing the contents of the twenty-four points of a backgammon board, the contents of each player's bar and home, the values of the two dice, the position of the doubling cube, and which side is to move next.[8] There are constraints on the values of these integers corresponding to the rules of backgammon.

The *World Model* is essentially a duplication of the Real World, but it is transformed "egocentrically", so the player who is to move always appears to be playing White and moving in the same direction around the board. The World Model is such that it can always be derived as needed from the current Real World, so that the program has no sense of history; for example, it is not possible for the QBKG program to observe and exploit weaknesses on the part of an opponent.

The *Judgement Structure* comes in two basic parts. There are a set of support routines that extract useful information from the World Model. These routines compute a wide variety of functions that have been found useful for guiding actions, from simple matters such as the number of points a player has made in his home board to more complex and esoteric functions such as the second moment of inertia of a position about the center of gravity, considering the men as point masses.

The second part of the Judgement Structure is the hierarchically structured evaluation function that

[8]There is additional state involved in match play that is not considered in this report. Also, state is maintained in some cases where it is computationally cheaper to incrementally update a previous value than to recompute it each time; e.g., the pipcounts.

produces a measure of the goodness of a given position, using the World Model and the results of the support routines as input. This component is at the heart of this research and is considered in Section 3.

This division of the Judgement Structure is in some sense unnecessary, since the language in which the structured evaluation function is expressed is powerful enough to compute the functions produced by the support routines directly from the World Model. However, to do so would lead to a significant slowdown in the speed of the system, due primarily to the cost of unrolling loops used in the support routines.[9] The penalty for using support routines such as this is that the analysis system is unable to make any explanation for why the value of a support routine is what it is. A good way to view the support routines is that they provide an "Enhanced World Model" in which determining behavior is easier and works in more powerful terms. That the system is unable to explain the enhancements in lower level terms is no more upsetting than the fact that humans are typically unable, for example, explain how they recognize a familiar face.[10]

The *Action Set* for QBKG is simply all legal moves from a given position as determined by the constraints forming the rules of backgammon. In the implementation, this set is never directly collected; instead the move generator proceeds around the board looking for legal ways to play the dice, and whenever one is found it is played in the World Model and the Judgement Structure is applied to it. The incrementally best move is kept along with its heuristic value until all moves have been examined. This move generation and evaluation procedure forms the entirety of the *Reasoning System* in the QBKG implementation.

The *Analysis System* is entered on the user's request after a move has been played. The task of interpreting the Judgement Structure for purposes of generating explanations is performed by comparing a suggested alternative move with the move just played. The Analysis System must distinguish significant from insignificant factors in a comparison, and for those issues judged worth mentioning, must produce some indication of the relative importance of each. The Analysis System is discussed in Section 4. Since QBKG is not a learning system, the Analysis System proceeds under the assumption that the Judgement Structure is perfect. An Analysis System for a learning program would need to have access to statistical performance data on the program's behavior, to be able to discover errors in its World Model and Judgement Structure.

[9] At least on a sequential machine. A parallel implementation of this scheme need not suffer the same fate.

[10] And no less upsetting: for a learning system such support routines might well be unalterable or nearly so, and an inability to get to the "raw inputs", or at least an inability to understand the connection between the raw inputs and the enhancements, could constitute a significant limitation.

3. Constructing evaluation functions

The approach to representing heuristic knowledge as an arithmetic function is presented, stressing the necessity and the danger of non-linearity. The notion of an ***application coefficient*** *is described. An evaluation function for a simple task is developed as an example of the method and the pitfalls. The phenomenon of the* ***blemish effect*** *and the consequent unavoidability of "boundary cases" is discussed. The language devised for the QBKG system is presented.*

3.1. The task of an evaluation function

The basic task of the knowledge representation scheme for judgements is to *approximate the relationship between the sensory data and a utility metric.* The utility metric must specify how well a given set of sensory inputs is expected to satisfy the goals of the organism. The QBKG system defines a language in which its Judgement Structure is written. The language is described in some detail in Section 3.5; here, we will consider some of the issues that led to the language being defined as it was.

An evaluation function is a more or less successful approximation depending on how well it is able to order the states of the Real World in terms of nearness to a goal state with respect to the actions the system can take. In general, the quality of the approximation can be traded off against the amount of searching done; at one extreme, the evaluation function only discriminates goal states and the search must blindly hunt for one, at the other extreme, the evaluation function totally orders the states of the Real World and a one-ply hill climber proceeds directly to the nearest goal state. Because of the combinatoric nature of searching, it seems desirable to try as far as possible to push towards the high-quality approximation end of the spectrum.[11]

3.1.1. Linear and non-linear evaluation functions

One of the conceptually simplest methods of evaluating the quality of a given state of affairs in the Real World is just a linear combination of the inputs to the system (such as counting points to evaluate the quality of a bridge hand, or counting material on the chess board with some weighting scheme such as "A rook is worth five pawns" and so on). A linear evaluation function attempts to provide a uniform measure of utility for every state in the problem space, providing a total ordering with respect to nearness to goal states. For sufficiently simple spaces, a linear approximation can work well, unfortunately, they have to be *very* simple. Any problem space where the utility of a given input is not constant or nearly so is too complex. Tic-tac-toe, for example, when the only inputs are the contents of the nine squares, is too complex for a linear combination of the inputs to succeed. By enhancing the original nine inputs with a few non-linear functions

[11]The fact that the better present-day chess programs push toward the other end of the spectrum suggests to us that, in general, too little is known about creating high-quality judgements. It seems clear that advances in searching technique and special-purpose hardware devices will add several more ply to the depth these systems can go before they "hit the wall", at which point improving the quality of the evaluations will take on more and more significance.

of them, such as the difference between the number of X's and O's on each possible winning line, a linear combination of this expanded set of inputs can be made to totally order the game states with respect to nearness of winning.[12]

3.1.2. Heuristic knowledge

In more complex problems such as backgammon or chess, no one has yet found any computation, other than hypothetical complete game trees, that will totally order the states of the problem spaces. What have been found are large collections of heuristics that provide partial orderings for small subsets of the problem spaces. Heuristics are frequently expressed in the form "If conditions are *x*, then try to accomplish *y*." Interpreting such a heuristic in terms of an evaluation function, our first suggestion is that a heuristic specifies a region *x* of the problem space within which the relation between *y* and utility is approximately linear. A collection of heuristics breaks a problem space up into a collection of (frequently overlapping) regions within which specified linear relationships between features and utility hold. Thus, the heart of the representation scheme is computations of the form:

Heuristic = (degree to which the current state is in region *x*)*
(constant of proportionality between utility and *y*)*
(degree to which *y* is accomplished)

However, most heuristic knowledge does not relate directly to any abstract notion of utility or goodness. More commonly, a heuristic relates quantities which only indirectly bear on utility. Consider the following beginner's chess advice:

> When there are 13 points worth of material on the board, or less, it is an endgame position.

This heuristic suggests a (rather discrete) method of estimating the "endgameness" of a chess position. It has nothing to do with utility directly, instead it helps determine to what degree you are in a region of the problem space where all the endgame heuristics are relevant to utility. Similarly, consider the backgammon blockading heuristic mentioned in Section 3.3:

> The goodness of a blockading formation is proportional to the cube of (36 – the number of rolls that allow a given man to escape across the blockade).

The addition of this heuristic caused a dramatic improvement in the quality of the BKG program's play. It does not relate directly to utility, since in some circumstances blockading enemy men is an undesirable goal,

[12] Similarly, the simple bridge and chess functions are always improved in a non-linear way; in bridge for example, points are added for short and long suits, and in chess, bishops are more valued than knights in open positions and vice-versa in closed positions. In Samuel's checkers program [18], he created a set of features which were non-linear in terms of the board position and then created linear combinations of them.

nor does it help determine a region in which other heuristics apply. What it does do is suggest a means of determining how well the goal of trapping enemy men is being accomplished. Other heuristics then specify how the goal of trapping enemy men relates to utility.

In general, this sort of heuristic knowledge may relate quantities in a region, or relate quantities to degree of being in a region, or relate quantities to degree of accomplishment of a goal. Thus, the basic scheme for representing a bit of heuristic knowledge about some concept w is:

$$
\begin{aligned}
(\text{degree of } w) = \; & (\text{degree to which the current state is in region } x) * \\
& (\text{constant of proportionality between } w \text{ and } y) * \\
& (\text{degree to which } y \text{ is accomplished})
\end{aligned}
$$

The picture of the representation scheme that emerges from this is a layered, hierarchical structure. At the very top is heuristic value or utility. Immediately below that is a layer of heuristics all of which relate directly to utility. In each successive layer are sets of heuristics which provide methods of computing the values used by the layer above. At the bottom of the hierarchy are values which are produced directly by the World Model. As an example, a somewhat schematized version of the top few branches (and a couple of the leaves) of the QBKG Judgement Structure is shown in Figure 3-1.

In QBKG, the Judgement Structure is treated in most places as if it were a tree, but in fact it is a directed acyclic graph. Since bits of heuristic knowledge are frequently useful in several different areas of the structure, there can be multiple parents for a concept as well as multiple children. **MyPipcount**, for example, is a component of several ascendancy chains leading to, among other things, considerations about how to bear men off, how much to worry about maintaining mobility and a flexible position, and whether there is enough time to perform a return play or release play.[13] Still, hierarchical structuring falls naturally out of the "patchwork" nature of heuristic knowledge, and is an important principle precisely because it supports the convenient expression of heuristic knowledge.

3.2. Representing collections of heuristics in an evaluation function

The basic method of representing heuristics presented in the previous section was designed so that a set of heuristics that all relate to utility can simply be added to derive a measure of the collective utility of a given state of affairs. The basic knowledge structuring mechanism is thus a sum of terms, where each term is composed of an *application coefficient* A specifying the region in which the heuristic applies, a *constant of proportionality* C, and a *feature* F which is to be related to utility. Figure 3-2 shows the general form.

[13] A "return" play deliberately exposes a man to be hit, when normal tactics would not suffice to win. A "release" play deliberately allows a blockaded enemy man to escape, in the hope of winning a gammon.

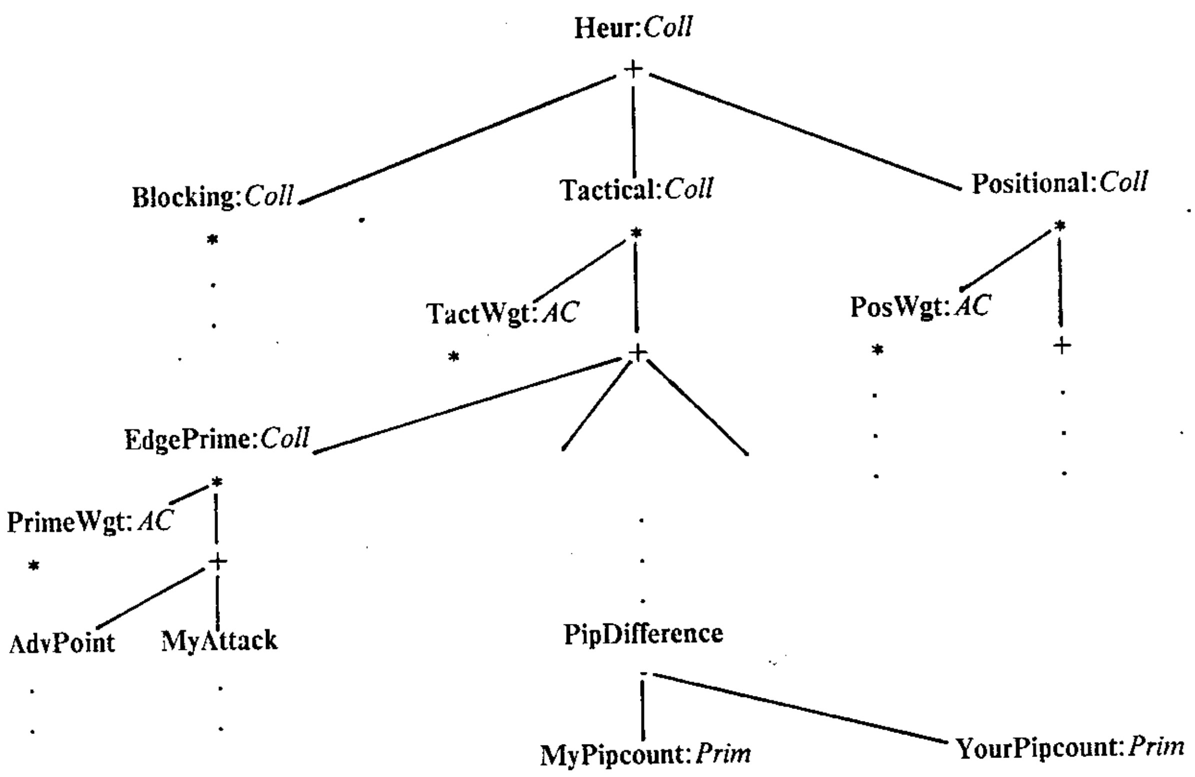

Figure 3-1: Schematic view of portions of the QBKG Judgement Structure.

$F = \sum_{f \in K} A_f C_f f$

with $F =$ the feature being defined,

$K =$ the set of features from which F is derived,

$A_f =$ a real variable in the range (0,1), the application coefficient associated with this use of feature f,

$C_f =$ a signed, real constant associated with this use of f.

Figure 3-2: Method of combining heuristic knowledge

The summation operator is the most common mechanism whereby differing knowledge sources are combined; where apples and oranges are compared. We call F a *heuristic*, *feature*, *concept*, or *node*, corresponding to viewing the Judgement Structure as a collection of heuristic knowledge, combinations of input data, a symbolic hierarchy, or a parse tree of an expression. F is said to *depend* on K. This recursive definition (in concert with similar definitions for the other operators, as in Table 3-2) terminates at the leaves of the parse tree with *primitive features* whose values are obtained directly from the the World Model.

3.2.1. Representing the limits of linearity

The *application coefficient, A*, is of critical importance in the construction of effective evaluation functions, as it is a main source of non-linearity in the function. The fundamental difficulty in using non-linear functions is the problem of *instability*: while a non-linear function may approximate utility quite closely over portions of the ranges of values of its inputs, the approximation can go drastically astray for some combinations of inputs if it is not carefully controlled. For example, IQ tests are supposed to approximate "intelligence" in some fashion. Idealistically, they are computed by the non-linear function *(raw score)/(age)*. This works adequately for ages in the range of about 6 to 16, but as the subject grows older, their IQ score by this metric begins to drop steadily. In reality, IQ scores are computed by the slightly more complicated function *(raw score)/min(age,16)*. By clipping the denominator in this fashion the tests results become much more stable.

The method (*Clip the inputs*) used to "fix up" the IQ computation is one of many ways to represent the limits of validity of an approximation. Some other common methods are:

- *Clip the result.* This technique is sometimes used in academia: Giving a test with, say, two hundred points possible, but grading on a one hundred point scale. It sacrifices resolution at the high end, and tends to be popular with the majority of students.

- *Scale the result.* This is frequently used when several results need to be linearly related, such as several exams that are supposed to have equal impact on the final evaluation. Scaling a result down sacrifices resolution over the entire range, and scaling a result up sacrifices resolution in the other results.

- *Tighten applicability criteria.* This is the method most commonly used by symbol-manipulation programs, as in Winston's arch-learning program [22]. A crude definition of an arch (e.g., "three blocks") gives the wrong answer (i.e., is unstable) on many structures and is tightened up ("three blocks with two supporting the third and not touching each other", etc.) This method sacrifices resolution in the space between the old, looser criteria and the new ones. (E.g., if a system thought three stacked blocks formed an arch, but was told they didn't, then what should the system think they *do* form?) In the IQ example, it might say that unless you are between six and sixteen you do not possess an IQ.[14]

- *Break into multiple approximations.* This technique is frequently used in concert with tightening applicability criteria, to preserve the resolution that would otherwise be lost. It can be seen almost

[14] Which might not be a bad idea! But this technique does not have to be discrete; the whole notion of application coefficients is designed to allow for gradual increasing or decreasing of the degree of applicability.

everywhere you look, from the esthetics of Impressionism and Dada to quantum mechanics and Newtonian mechanics and relativistic mechanics. In theory, no resolution will be lost, however, the price is paid in the difficulty of evaluating states that are close to the "turnover points" between the approximations.

All of these methods and others have their place, circumstances in which they perform effectively. In many cases, however, statements can be made about how these techniques should be applied in order to avoid unexpected side-effects. Consider the issue of applicability criteria.

As discussed below in Section 3.3, it is in general unsatisfactory to simply supply a numeric range or a boolean predicate for a given relationship and only include it in the evaluation when the input is within the range, because this induces discontinuities at the extremes of the range. What is needed is a "degree of applicability" factor that decreases smoothly to zero as the input approaches the limits of linearity with respect to a relation, and this is precisely what application coefficients are designed to provide.

Local linearity: The fundamental condition for using non-linearity in this fashion is that the condition of *local linearity* be maintained: when evaluating positions *near* each other in the problem space, the evaluation function must be effectively linear in its inputs. Restricting non-linearity to the multiplication of features by application coefficients maintains that condition because the ACs change slowly and smoothly with respect to the rate of actions. In a small region of the problem space, therefore, they are assumed to act as constants. A consequence of local linearity is that, while sufficiently similar sets of inputs may be evaluated and compared for relative utility, when the sets of inputs are different enough that the application coefficients are significantly changed, then comparisons of the computed heuristic values will not in general yield a valid absolute comparison of the two sets of inputs.

Application coefficients provide the Judgement Structure with context sensitivity, allowing control over what information is deemed relevant in any given circumstances. It is widely recognized, for example, that the information important to the end game in chess is quite different from that needed in the middle game; it is also widely recognized that there is no well-defined boundary between the two phases of a game. An application coefficient allows for the smooth "turning down" of the importance of middle game issues and a corresponding "turning up" of the importance of end game issues as the game progresses.

Typical examples of application coefficients from the QBKG program include the pipcounts for the two players (which is an indicator of the expected length of the remainder of the game) and the degree of blockading of the opponent (which is a central strategic quantity indicating the degree to which one side is in charge of the situation.) In the evaluation function for the ***Iago*** Othello program [17], the move number is used as an application coefficient, to gradually sensitize the program to changes in context as the game progresses. This is effective because an Othello game cannot exceed sixty moves. Application coefficients are also being used in an evaluation function designed to judge plans for job-shop scheduling [7], measuring such features as the amount of free space remaining on the floor.

3.3. A sample evaluation function

To take an example, suppose the actions of a system consist of buying one apple or one orange per move, and the goal is to maximize calories while minimizing cost. The evaluation function could be phrased:[15]

$$\mathbf{Heur} = -\mathbf{CostWgt} * \mathbf{DollarValue} + \mathbf{CalWgt} * \mathbf{CalorieValue}$$

$$\mathbf{DollarValue} = \mathbf{PricePerOrange} * \mathbf{OrangeCount} + \mathbf{PricePerApple} * \mathbf{AppleCount}$$

$$\mathbf{CalorieValue} = 100 * \mathbf{OrangeCount} + 150 * \mathbf{AppleCount}$$

In this example, **PricePerOrange** and **PricePerApple** fill the role of application coefficients, which vary only slowly with respect to the rate of actions and are only indirectly (if at all) under the system's control.[16] **CostWgt** and **CalWgt** are assumed to be constants in this simple example, and together they specify what the system thinks is favorable in terms of price/calorie (so, for example, if we gave the system the added action of not buying anything, it would wait for a favorable ratio before purchasing). **OrangeCount** and **AppleCount** (as well as the application coefficients) are *primitive* features: their values are obtained directly by measuring the Real World. It is the job of the World Model to determine that the hypothesized action of buying an apple or an orange will lead to **AppleCount** or **OrangeCount** being increased by one. In addition, the World Model must be able to produce likely values for **PricePerOrange** and **PricePerApple**; since they are being taken as application coefficients it is reasonable to assume that they will be the same in the near future. If they have changed, that will be noted when the next purchase is made and the World Model is updated.

Given this, the system conducts business in a very simple way. It hypothesizes buying another orange and obtains a resulting **Heur**, does the same for another apple, and selects whichever action leads to the larger **Heur**.

As the system grows in complexity from this simple example, more and more considerations are included in the evaluation function. If, for example, the designer wants to ensure that the system obtains enough apples to make an apple pie (say, ten are required), a term would appear to reflect that desire. An obvious

---

[15] In these examples, the functions are presented as collections of mathematical equations, not as sequences of assignment statements. The equation defining **Heur** can be taken as the root of the parse tree of which the other equations form subtrees.

[16] They actually incorporate the constant C from the definition above, which in this case represents some maximum possible price per item. Furthermore, prices are rarely found to vary between purchasing one apple and the next; we'll assume that the environment this organism finds itself in has a rather volatile produce market.

thing to try would be:

$$\text{Heur} = -\text{CostWgt}*\text{DollarValue} + \text{CalWgt}*\text{CalorieValue} + -\text{PieWgt}*\text{AppleNeed} \quad (1)$$
$$\text{AppleNeed} = \textit{if}\ \text{AppleCount} < 10\ \textit{then}\ 1\ \textit{else}\ 0$$
$$\text{DollarValue} = \text{PricePerOrange}*\text{OrangeCount} + \text{PricePerApple}*\text{AppleCount}$$
$$\text{CalorieValue} = 100*\text{OrangeCount} + 150*\text{AppleCount}$$

Here we have added a negatively valued term derived from **AppleNeed** that won't go away until the system has bought ten apples. This, however, fails to work properly, because it provides no incentive to buy the first nine apples, and so if oranges are a better price/calorie tradeoff, the system will do better locally by buying only oranges and putting up with the "pain" of the **AppleNeed** as inescapable. A workable solution in this case would be as above but defining:

$$\text{AppleNeed} = \textit{max}(0,10\text{-AppleCount}) \quad (2)$$

Now, buying another apple under ten provides the benefit of decreasing the **AppleNeed** as well as increasing the **CalorieValue**. The only subtlety in this formulation is that one must ensure that **PieWgt** is large enough to dominate the decision even when **AppleCount** reaches nine, to keep the system's attention focused on apples until enough have been obtained.

3.3.1. Evaluation surfaces and hill climbing

From the perspective of the organism's overall behavior patterns, the use of application coefficients has an interesting effect. A given evaluation function describes a surface in the space defined by the inputs and the utility metric. The behavior of the system is then simple hill-climbing on this surface, and the system can easily be trapped on local maxima. If the evaluation function is linear in the inputs, the "geography" of the evaluation surface is fixed and unchanging, but using application coefficients introduces a kind of relativity: if the application coefficients are assumed to be constant when projecting the view from any given position on the surface (local linearity), then the *shape of the landscape* changes depending on where the system is.

To make this clearer, assume that the market for oranges is such that the price rises with every purchase, and that the price was such that initially the system bought oranges, but that when the system has bought eight oranges the price will be such that apples are the preferred choice. The system is unaware of this dependency and projects the landscape as though the prices would be constant. Figure 3-3 shows the perceived view after three oranges have been purchased.[17] The "lay of the land" is sharply in favor of purchasing oranges.

[17] In these figures, **AppleCount** increases toward the left and **OrangeCount** increases toward the right.

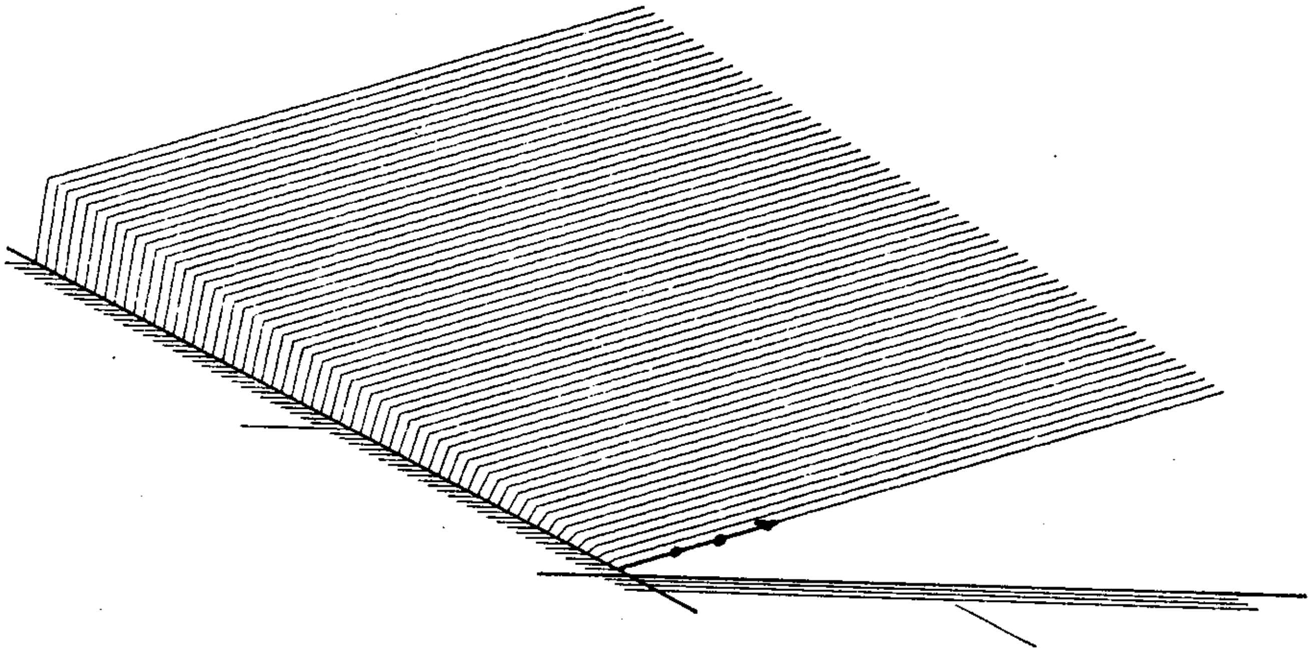

Figure 3-3: Projected view after purchasing three oranges

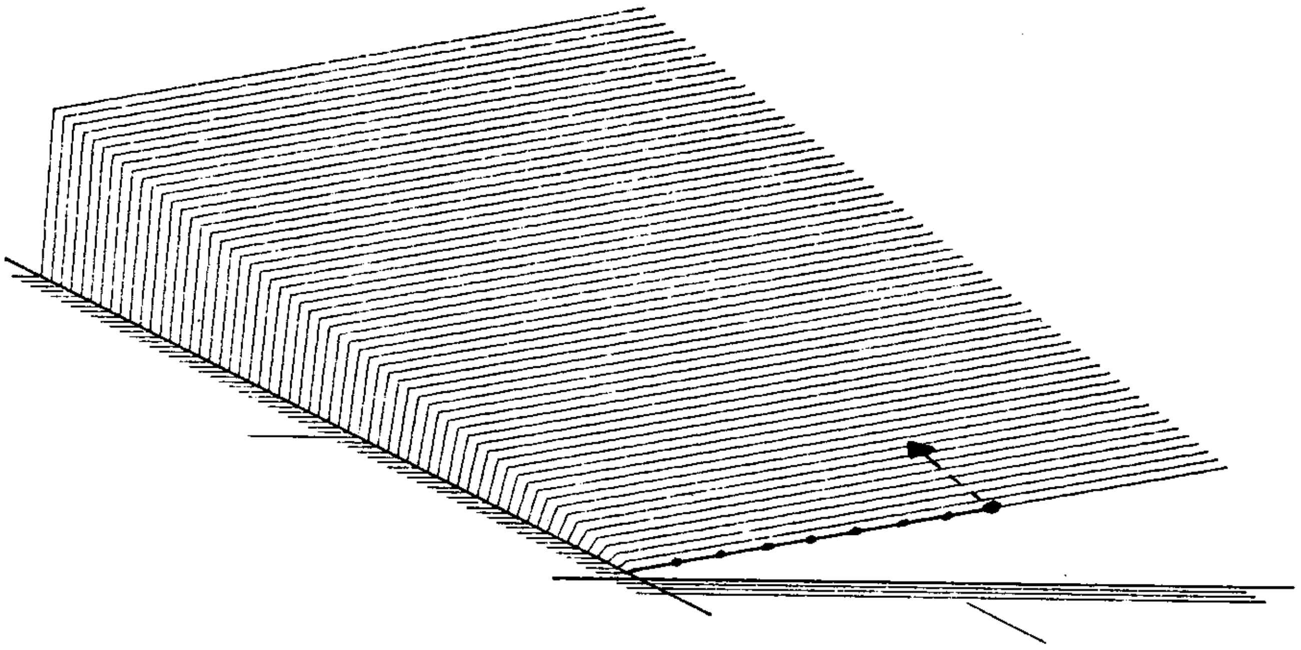

Figure 3-4: Projected view after purchasing eight oranges

Contrast Figure 3-3 with Figure 3-4, when the system has purchased eight oranges. Now the slope in the orange direction has flattened and the slope in the apple direction is steeper, and based on this view, the system "changes its mind" and begins buying apples. Figures 3-3 and 3-4 represent tangent planes to the "true" landscape (generated by an evaluation function which models the dependency in orange prices) shown in Figure 3-5.

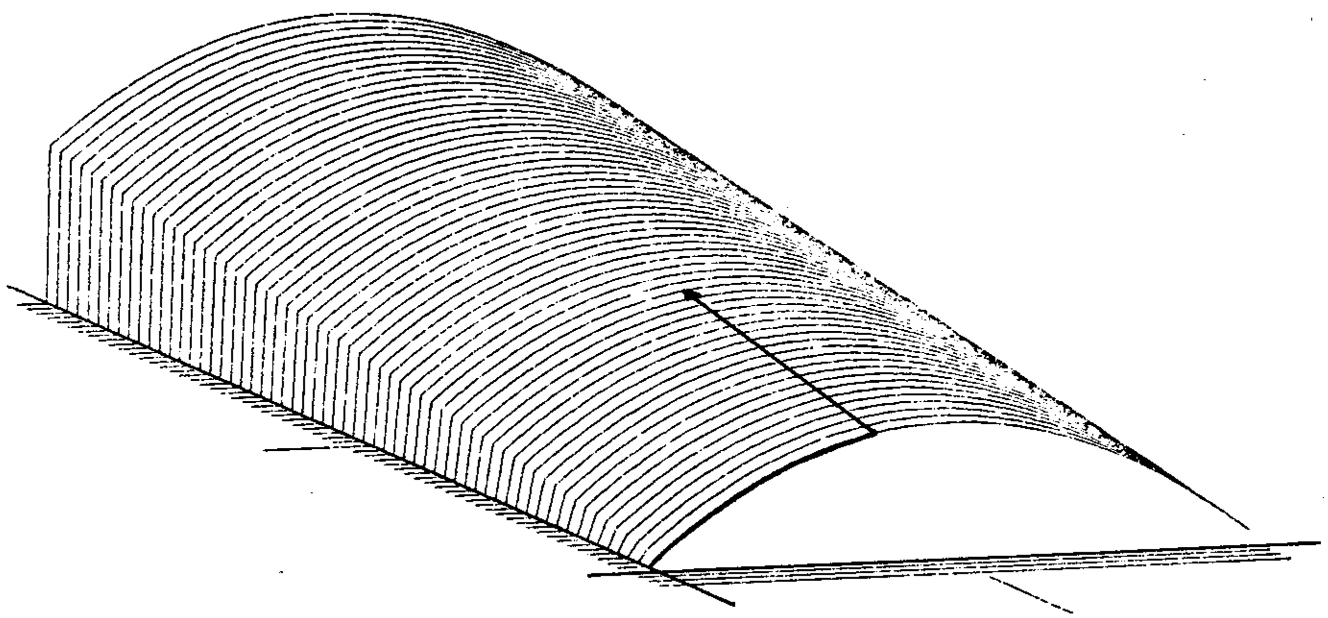

Figure 3-5: Landscape modelling the non-linearity in orange prices

It is worth taking a moment to compare the evaluation function using Equations 2 with a somewhat similar one meant to accomplish the same task:

$$\begin{aligned}
\textbf{Heur} &= \textit{if}\,\textbf{AppleCount} < 10\ \textit{then}\ \textbf{PieWgt}*\textbf{AppleCount}\ \textit{else}\ \textbf{Factors} \\
\textbf{Factors} &= -\textbf{CostWgt}*\textbf{DollarValue} + \textbf{CalWgt}*\textbf{CalorieValue} \\
\textbf{DollarValue} &= \textbf{PricePerOrange}*\textbf{OrangeCount} + \textbf{PricePerApple}*\textbf{AppleCount} \\
\textbf{CalorieValue} &= 100*\textbf{OrangeCount} + 150*\textbf{AppleCount}
\end{aligned} \tag{3}$$

In this case, the buying of ten apples has been effectively separated into its own task that must be disposed of before other factors are even considered. For any positive **PieWgt**, the system performs well initially, purchasing nine apples. Upon hypothesizing the purchase of a tenth apple, however, the condition of the *if* statement becomes false, and the system suddenly becomes aware of the price and calorie issues which, let us assume, result in a much lower value for **Heur**. The hypothesized purchase of an orange, however, causes no such uncomfortable revelations and will be selected from then on. Note that this can happen even if apples are the better buy, based on price and calorie issues alone. This is an instance of the system becoming trapped in a local maximum, and is a potential hazard whenever there is a significant discrete boundary between adjacent states in the evaluation function. The previous evaluation function avoids the problem by having the importance of buying apples for the pie smoothly decrease relative to the other issues, and at the hypothesized purchase of the tenth apple, the other factors are already being weighed in the decision and the system is not "surprised" by the price it has been paying for apples. Figures 3-6, 3-7, and 3-8 show the landscapes generated by Equations 1, 2, and 3 respectively.

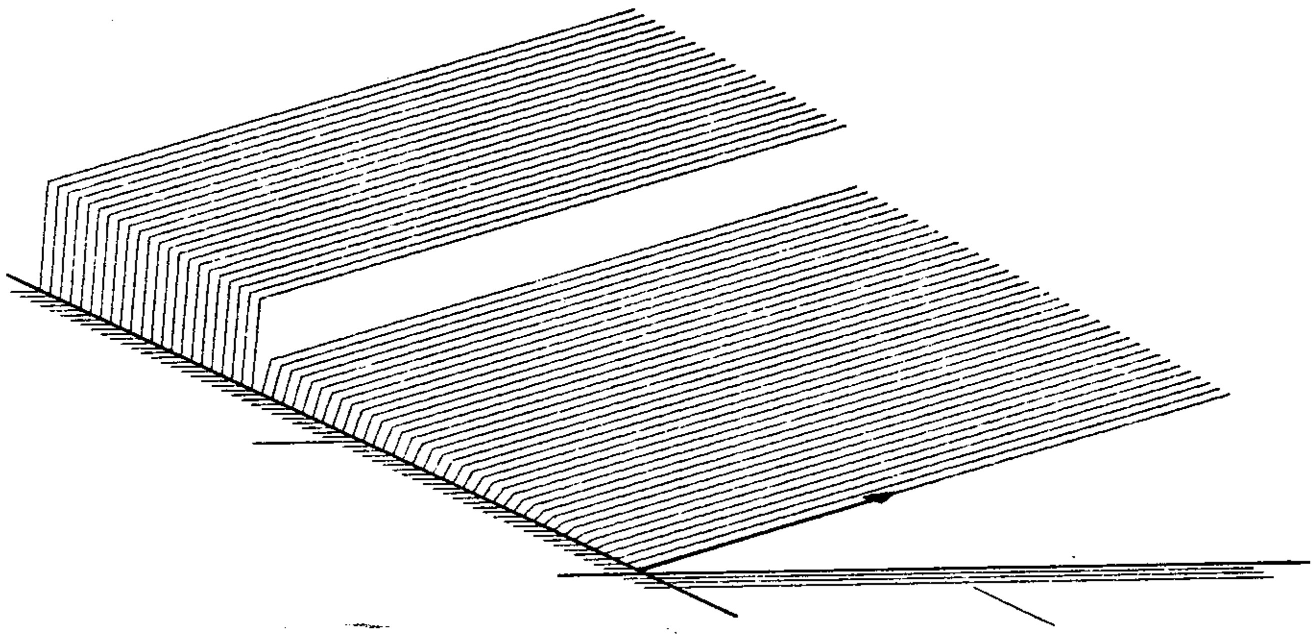

Figure 3-6: Landscape corresponding to Equations 1

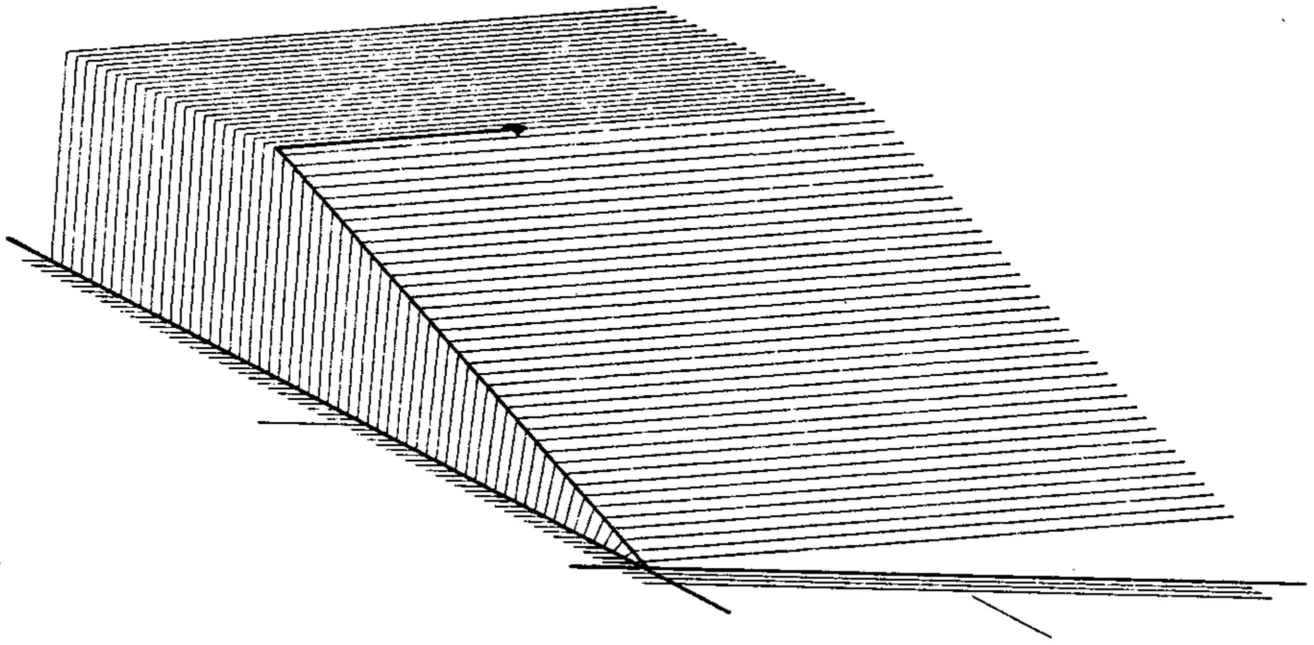

Figure 3-7: Landscape corresponding to Equations 2

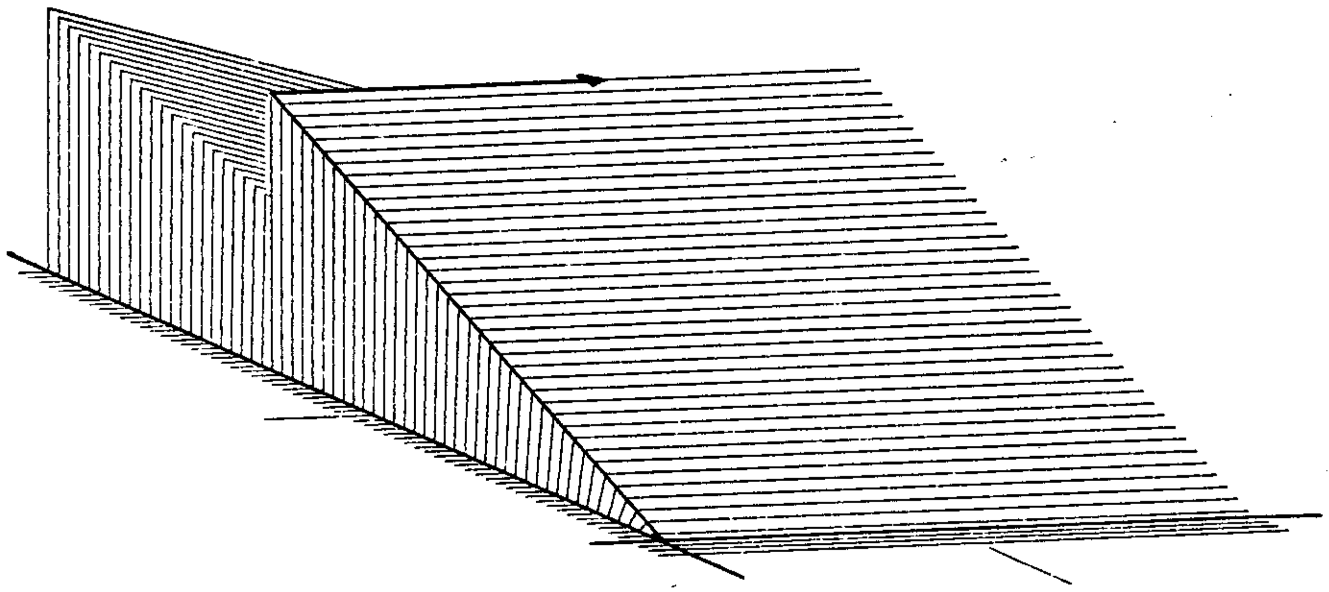

Figure 3-8: Landscape corresponding to Equations 3

3.4. Boundary cases and the blemish effect

The evaluation function represented by Equations 3 in the previous section contains an instance of a hazard that has been termed the *blemish effect* in [2]. A blemish is a spot of roughness or a discontinuity caused by the joining of two otherwise smooth surfaces formed by the values of the evaluation function in the state space. The key aspect of a blemish is that it is *artificial*, that it is an artifact of inadequate representation of knowledge, rather than an irregularity of the "true" landscape. If, for example, the system was purchasing not apples and oranges but apples and plutonium, there could be a *genuine* cliff in the evaluation surface at the amount of plutonium which is near its critical mass. In such a case, assuming plutonium had greater incremental utility than apples, the proper behavior of the system would be to climb to the top of the ridge by buying plutonium, and then run along the ridge by buying apples.

Probably the most pervasive source of blemishes is *excessively discrete representation of heuristics*, as in the example in Equations 1 and 3. The typical feeling in creating a discrete form of a heuristic is that it recognizes all the common positive instances, rejects all the common negative instances, and the boundary cases will occur infrequently enough that the possibility of errors due to them can be ignored. The fallacy is in the last clause: in any system which uses heuristic knowledge to decide between actions, the system will actively *seek out* the boundary cases that correspond to ridges in the evaluation surface.

What makes blemishes such a problem for the designer of an evaluation function is that, once introduced, they can be very hard to find. Severe blemishes generally arise through the interaction of multiple heuristics, each of which may look quite reasonable by itself. Faced with something like the powerset of the heuristics as the set of possible sites for blemishes, and the possibility of introducing new blemishes in the course of eliminating old ones, the designer who hopes to eliminate blemishes by case-testing is engaged in a very tedious task, the end of which may be nowhere in sight.

The familiar notion of "obeying the letter of the law while violating its spirit" is exactly this sort of blemish-finding behavior. The difference between the letter and the spirit of the law is the difference between the evaluation surface induced by the rules of law and the abstract evaluation surface of "right behavior" which the rules of law attempt to approximate. Anybody who takes only the letter of the law as an evaluation function will naturally head in the direction of those boundary cases where the utility of the letter of the law differs most from the utility of its spirit.[18] The (almost) monotonically increasing corpus of common law, defined by judges in response to specific incidents of this sort of behavior, represents the current status of the ongoing process of incrementally patching the rule of law to more closely approximate its spirit.[19]

Another case in which the blemish effect can be seen clearly, this one involving recent AI research, is the behavior of the Eurisko system [11, 12] in designing squadrons of space ships for a simulation game.[20] In this instance, the creators of the Traveller: Trillion Credit Squadron (TCS) game had produced a set of rules that they hoped would model some aspects of space-warship design in an interesting and "realistic" way (whatever that might mean, in such circumstances.) Eurisko was provided with the rules of the game as written (but not, of course, with the abstract rules which the creators of the game were attempting to model) and a simulator allowing it to determine which of two given fleets was superior, and it proceeded to design fleets and gather data about their performances. Inexorably, the blemish effect made its presence known:

> After a while, Eurisko noticed another kind of regularity. For each parameter, the optimal value seemed to be almost, but not quite, an extreme value. This was formed into a heuristic that enabled Eurisko to very rapidly settle in on a winning design. That new heuristic said
>
>> IF designing either an individual ship or a fleet for Traveller TCS, and a certain parameter is having its value changed, THEN change it to a nearly — but not quite

[18] "Tax loopholes" are excellent examples of this sort of behavior. In those cases, there is a continuous quantity to be maximized and all sorts of various magnitude discrete boundaries to seek out. The boundary between December 31st and January 1st is one obvious example. Exploiting it in combination with other tax laws can lead to truly outrageous, but technically legal, results.

[19] This analysis, of course, ignores the fact that the evaluation surface of right behavior is not unchanging. Much of the effort of the legal system in general, and the Supreme Court in particular, is spent on updating those rules of law which once approximated right behavior adequately, but no longer seem to.

[20] Such squadrons are, according to the scenario of the simulation, part of the "Navy of the Imperium"; the usage of "naval" in [13] led some readers to believe that the game involved ocean-going vessels.

> — extremal value.
>
> The final fleet Eurisko designed had a large number of ships — each was fairly small, each had nearly as many types of weapons as allowed, each was nearly as heavily armored as possible, each was nearly as slow as possible, etc. [12, pp 20-21]

Eurisko had discovered the blemish effect, and formulated it into a heuristic allowing it to leap across the evaluation surface directly to potentially useful boundary cases. The sort of "strategic" fleet design that the TCS creators might have had in mind probably involved different classes of ships (fighters, battleships, etc.) loosely analogous in function to what contemporary military fleet design principles might suggest, but that was unnecessary given the letter of the rules they had created. In their effort to make the rules of the game relatively simple (when directly modelling something as complex as space warfare, the set of rules for TCS must be counted as simple, even though they occupy some two hundred pages), they created an evaluation surface full of blemishes, where the optimal solutions lie near the extremes of parameters, at the edges of cliffs, rather than in positions which balance the values of the various parameters, as they may have hoped.

In the devastating blemish that Eurisko located, several boundary cases were involved. The major relevant rule was that as long as *any* ships in the fleet were still functioning, other damaged ships could be returned to "port" for repairs. Eurisko exploited this, in combination with the rules governing the conditions under which a ship is hit by enemy fire, by designing a very small, very high-speed ship with so much evasive and defensive capability that it was *impossible* to hit, thus allowing all the "big guns" as much time for repair after battles as was needed. When looking over the results of Eurisko's efforts, Lenat noticed the anomaly and ensured that one such ship was included in the final fleet design. After the fleet designed jointly by Lenat and Eurisko won a national tournament, the rules were changed to decrease the utility of the particular blemish that Eurisko had found, by eliminating the process of making repairs, as well as other changes. Not surprisingly, Eurisko's near-extremal heuristic still applied, and Lenat and it designed another boundary case fleet that was subsequently victorious in a regional tournament.

Incorrect behavior because of blemishes is not limited to particular search strategies such as QBKG's 1-ply search; it is endemic to the heuristic search technique generally. As an example, the horizon effect [1] can be viewed as a special case of the blemish effect. In the example in the previous section (Equations 3), since the system is searching just one ply, by buying oranges the system is able to push the perceived unpleasantness of the price of apples over the horizon indefinitely. From this perspective, it can be seen that the point at which any heuristic search system stops searching (at whatever depth) and performs a terminal evaluation may be an instance of a discrete boundary. The technique of searching until *quiescence* is an attempt to smooth out this discrete boundary by continuing the search to a point where the evaluation function will hopefully be well-behaved in the region just beyond where the search stopped.

3.5. Language description

The language in which the QBKG Judgement Structure is written is a real-valued algebraic LISP-based language. Concepts are represented by atoms with several associated properties specifying how and when to compute the value of the concept (FORM, AGE, LAGE) and supplying information for use by the explanation system (NAME, UNIT, COL, LO, HI). Two typical concept definitions extracted from the QBKG database appear in Figure 3-9. Each definition supplies an atom by which the concept is referred to internally and a list of property-value pairs. The FORM property specifies how to compute the value of the node in terms of an operator and a series of concepts whose values become the arguments to the operator. The AGE property retains the time (in terms of a once-per-evaluation counter) of the last computation of the value of the concept. Table 3-1 summarizes the properties that are defined.

```
(DBDEF  BLOCKING-GAME (
 FORM   (SUM APPLIED-O-CONT APPLIED-M-CONT RUN-DESIRE)
 UNIT   BWC
 NAME   "in the blocking game"
 AGE    31035
 COL    FEAT
 LO     -248.73600
 HI     575.14665
 LAGE   1972
))

(DBDEF  APPLIED-O-CONT (
 FORM   (TERM ((END-AND-RACE 25.0) O-CONTFACTOR))
 UNIT   LS
 NAME   "effective containment of $o"
 AGE    31035
 LO     0.0
 HI     600.0
 LAGE   1972
 AC     END-AND-RACE
))
```

Figure 3-9: Sample concept definitions

The value of the FORM property looks somewhat like a Lisp functional form, but it is not. The operator is restricted to being one of the QBKG language operators as described in Table 3-2, and the arguments must be either other concepts or constants.[21]

A priori, it would seem necessary to include in some manner all possible mathematical functions, as it is easy to imagine that the utility of some sensory input might vary in any possible way with respect to the amount of the input. While it is possible to provide only a minimal set of language operators and use

[21] The TERM operator has a more structured argument format in order to support the common cases where either the application coefficient or the scaling constant equals 1. The PRIM operator, of course, is the exception to the rule for argument types.

Property	*Usage*
NAME	The body of the english phrase to output when discussing this concept. Note that adding qualifiers is controlled by the UNIT property.
FORM	The definition of this concept in terms of a mathematical operator and a set of lower-level concepts.
UNIT	A "qualifier type" code, which specifies what scale to use when qualifying differences in this concepts, such as larger/smaller, better chances/worse chances, further ahead/further behind, etc.
USED-IN	A list of all immediate parents of this concept. (The USED-IN property is computed during the loading of the database and thus does not appear explicitly in the definitions shown in Figure 3-9.)
COL	Specifies whether this concept (which must be a SUM) is to be regarded as a *collection* by the explanation system.
DEFINE	A canned statement attached to PRIMitive concepts explaining the meaning of the concept, to be output if the user requests a definition of a PRIM.
AGE	The value of the once-per-evaluation clock as of the last time this concept's value was computed and stored.
VAL0,VAL1	Two slots for storing the value of this concept, one slot for the move actually made and one slot for a suggested comparison move.
LO,HI	Global minimum and maximum values for this concept over the lifetime of the database.
LAGE	The value of the once-per-move clock as of the last time this concept's value was stored. (This slot is used to determine when to reset LLO and LHI.)
LLO,LHI	Minimum and maximum values for this concept over all possible moves from a given position.
AC	(Not currently used.) A vestige of an experiment with producing counterfactual explanations (see Section 6.1.1).

Table 3-1: Properties of QBKG concepts

functional composition to create the functions needed, this leads to difficulties in automating the analysis of the Judgement Structure for explanation purposes.[22] Instead, the system directly provides operators for each common function encountered in creating the evaluation function. The system currently provides a squaring function, for example, but not a logarithm function, since logarithms have not been necessary so far. It is

[22]In effect, leading to a large number of nodes with a low information content and forcing the explanation system to be able to examine groups of connected nodes to derive the overall effect. The alternative taken here, on the other hand, of providing a separate operator for each common function, could cause the explanation system not to recognize similarities between related functions that have different operators. So far, this has not been a problem.

relatively easy to add a new operator to the system, and the lack of an unused operator is of little consequence unless the system is extended to do learning.

Arithmetic Operators	*Interpretation*
(SUM f_1 f_2 ...)	$f_1 + f_2 + \ldots$
(TERM (f_a f_b) f_c)	$(f_a / f_b) * f_c$
(SQR f)	f^2
(CUBE f)	f^3
(MAX f_1 f_2 ...)	maximum f_i
(MIN f_1 f_2 ...)	minimum f_i
(PRIM s-expression)	primitive operation = (EVAL s-expression)

Boolean Operators	*Interpretation*
(NOT f)	1 if f = 0, else 0
(AND f_1 f_2 ...)	$f_1 * f_2 * \ldots$
(OR f_1 f_2 ...)	$\|f_1\| + \|f_2\| + \ldots$
(GTR f_1 f_2)	1 if $f_1 > f_2$, otherwise 0
	similarly for EQL,NEQ,LSS,LEQ,GEQ
(IF f_c f_t f_e)	f_t if $f_c \neq 0$, f_e otherwise.

Table 3-2: Language Operators

At the leaves of the tree induced by the chaining through the FORM properties are concepts that are denoted *primitive* (labelled *Prim* in Figure 3-1). These are values that are either directly part of the World Model or derived from the World Model by the support routines. At the top of the tree is the distinguished concept **Heur**, which is the value returned as the result of the evaluation.

A single evaluation is produced in the following way. First, the once-per-evaluation counter is incremented, which invalidates all values stored on the concepts. The recursive evaluation procedure EVA is called on **Heur**. EVA first checks if the AGE of its argument is greater than or equal to the counter. If so, it returns the stored value. If not, EVA calls itself to obtain the values of all the concepts referenced in the FORM property of its argument. The recursion ends at PRIM nodes which either access the World Model directly or run procedures which access the World Model and produce a value (see Section 2.2). EVA then computes the value of the FORM using the returned values. The value is checked against LO and HI (and LLO and LHI) and the bounds are expanded if necessary. Finally, EVA stores the value and the current time, and returns the value.

Since there is significant sharing among the concepts, this caching procedure effects a speed up. More importantly, though, it makes all the intermediate values used in computing the heuristic value of a position available for inspection by the analysis system. There are in fact two properties reserved for storing values, so two sets of intermediate values may be cached at one time, facilitating comparisons of values to generate explanations.

After having seen the position we take on continuous and discrete representations (both in this report, especially in Sections 3.3 and 5.3, and in [4]), the reader may be surprised to find boolean operators in Table 3-2. There are several reasons for the inclusion, foremost of which is that there are occasions when the environment of the system genuinely contains a discrete distinction. In backgammon, for example, once the opposing player has successfully gotten one man off the board, it is then impossible to win a gammon or backgammon against him. The evaluation function should completely ignore its knowledge relating to winning a gammon once that has happened. Similarly, but viewed from the outside, if you observed someone run madly to some apparently unremarkable place on a sidewalk and then stand around doing nothing, you might wonder why there was such an abrupt change in the person's behavior, until you discovered that spot was a poorly marked busstop. Whether a discontinuity in the evaluation function should be considered a blemish depends entirely on whether or not it reflects a discontinuity in the actual utilities of states of the environment.

Boolean operators are needed in QBKG, therefore, because the true evaluation surface of backgammon contains significant discontinuities. Because of that, boolean operations do not *necessarily* cause blemishes if they are used carefully, and they add to the expressiveness of the language, making the explanation task easier. Using the IF construction is guaranteed not to cause an *unintended* discontinuity if it can be shown that the then-clause and the else-clause will have similar values whenever the if-clause is near changing truth value. For example, the following two definitions are equivalent:

AppleNeed = *max*(0,10 − **AppleCount**)
AppleNeed = *if* **AppleCount** < 10 *then* 10 − **AppleCount** *else* 0

Conversely, it is possible in any case to cause blemishes using just the arithmetic operations. For example, consider the following doomed attempt at representing the desire for ten apples:

AppleNeed = **PieWgt**∗**AppleCount**∗(1 − **HaveTen**)
HaveTen = **AppleCount**∗(**AppleCount** − 1)∗(**AppleCount** − 2)∗ . . . ∗(**AppleCount** − 10)

This version of **AppleNeed** rises steadily in **PieWgt** increments, rewarding the purchase of the first ten apples. When the system hypothesizes the purchase of an eleventh apple, however, it drops to about −**AppleCount!**∗**PieWgt**. It can be expressed in QBKG operators by alternating TERMs and SUMs nested twenty deep. Boolean operators neither necessarily engender blemishes nor does their lack automatically protect from blemishes.

4. Generating explanations

The main related issues in explaining judgements are presented: determining what is relevant, and determining when a quantitative difference should be explained as a qualitative difference. The heuristics employed in QBKG for making such decisions are presented. Examples of QBKG's commentary are presented and discussed.

The explanation mechanism of QBKG must handle two main issues. First, it must isolate the backgammon knowledge relevant to any particular query from the large amount of knowledge that does not bear on a given situation. The second issue is to provide some mechanism for deciding when quantitative changes should be viewed as qualitative change; in essence, to provide the ability to make distinctions that discrete systems enjoy free by virtue of their discrete representation. (There was relatively little effort expended in the generation of natural language output; in the examples below, the output has been left "in the rough" as the system generated it. Most "language issues" have been ignored.) The explanations that QBKG generates cover only a portion of the possibly useful comments that could be made, depending on the particular situations and on the level of backgammon knowledge of the listener. Some of the explanation topics which QBKG does not address are discussed in Section 6.1.

4.1. Finding relevant issues

QBKG is oriented around answering the question "Why did you make *that* move, as opposed to *this* move?" This reduces the explanation task to one of accounting for the *differences* between a pair of moves. The fundamental assumption of the explanation process is that important differences between a pair of moves will be reflected by "large" changes in the values of the highest level concepts that are related to the differences. Letting *Move1* denote the move with the larger **Heur** and *Move2* the one with the smaller, define $\delta concept$ = value of *concept* for *Move1* - value of *concept* for *Move2*. Referring to Figure 3-1, this assumption implies that if δ**Blocking** is "small", then backgammon knowledge related to blocking is not relevant to this comparison and should not be mentioned. If only one subconcept of **Heur**, say, **Tactical**, is not small, then all interesting differences are with respect to **Tactical** concepts, so the level of discourse for comparison is narrowed to just tactical knowledge, and the process repeats on the subconcepts of **Tactical**.

That is the method by which the relevant backgammon knowledge is isolated. Beginning at **Heur**, the system searches down the tree until a level is reached at which more than one significant difference is found.[23]

[23] There are two other ways in which this procedure can terminate. In the first case, the search proceeds all the way to a primitive feature without finding more than one significant difference. When that happens, the system states that that particular feature accounts for the difference between the moves. In the second case, the system arrives at a level where no significant differences are found at all. When that happens, the system states that a combination of small differences in the parent concept accounts for the difference between the moves. Both cases are infrequent.

As desired, if the two moves are radically different in their effects, the commentary will begin at a relatively abstract level (e.g., tactical and positional issues) and if the two moves are quite similar, the commentary will focus on the crucial differences at whatever level they are found.

If the discussion is at a fairly narrow level, it is usually sufficient to just describe the differences and their magnitudes; at higher levels, this leads to unsatisfying, "hand waving" commentaries. The level at which an explanation feels satisfying varies from person to person and topic to topic[24], so we have adopted a simple heuristic. The broad concepts at the top of the tree are denoted *collections* (*Coll* in Figure 3-1), and the system is built to automatically "look inside" any collections that are mentioned; in effect, supplying for free the question "Why is there a difference in that collection?"

4.2. Qualifying differences in magnitude

The above discussion is predicated upon having the ability to recognize "large" or "significant" differences in the values of concepts. In a two-valued system, any difference is a large one (on the order of *True* versus *False*), and the process of recognizing significant differences is done *outside* of the system, during the generation of discrete primitive observations of a continuous world. Unfortunately, it is impossible to determine what should be considered significant without considering the *context* within which the judgement is to be made. A difference of ten feet, for example, is much larger in the context of "Distance I am from the ground" than in the context of "Distance I am from the moon." A context provides a means of classifying differences into fuzzy classes or "buckets" such as "about the same", "somewhat larger", and so on. The number of classes will vary depending on personal taste as well as the degree of refinement of the knowledge base. The heuristics in QBKG employ different numbers of buckets, from three ("insignificant", "significant", "major") to six ("not significantly", "slightly", "somewhat", "much", "very much", and "vastly").

In terms of the QBKG structure, the context of a concept is the set of more general concepts of which it is a part. Some of the more primitive concepts, such as **MyPipcount**, appear in several places in the Judgement Structure and can therefore be judged in several different contexts. To judge a difference in context, it is necessary to determine how that difference affects the value at the top of the Judgement Structure. Given the sign and magnitude of a difference that is to be considered a significant improvement for **Heur**, this *goodness metric* can be propagated down through the tree to determine how much better or worse one move is than another with respect to a given concept in a given position. In Figure 3-1, for example, if the goodness metric for Heur is assumed to be +10, and in a given position TactWgt equaled 3, then the goodness metric for

[24]As evidenced by the child who responds to every explanation with "Why?"

EdgePrime would be $+10/3$, and a δ**EdgePrime** of less than $10/3$ would be judged "about the same", a δ**EdgePrime** between $10/3$ and $20/3$ would be "somewhat better", and so on.

Using this procedure would require an *a priori* goodness metric for **Heur**. In QBKG, probably the most satisfying overall context would be "What is the expected value of the game?" with a goodness metric of perhaps a hundredth of a point. Such an evaluation function could in theory be built, and in fact the system has an independent computation used to approximate the expected value, which is used in making doubling decisions.[25] The original BKG evaluation function only needed to order the possible moves with respect to a given initial position, and this "relative" nature remained through the translation to the QBKG-style evaluation function, so **Heur** values resulting from different initial positions are not directly comparable. With respect to a given position, however, various heuristics have been devised that empirically give satisfactory results in the determination of significant differences.

While an absolute goodness metric would cleanly solve the problem of determining the overall significance of a difference in a concept, there is reason to suspect that such a metric might be so difficult to come by for complex domains that systems in general must make do without. Clearly, an error-free, absolute goodness metric for a problem space, of sufficient resolution to distinguish between individual actions, would constitute an algorithmic solution to the problem. If such a metric were available for a domain and a goal, few would consider a system using it to be performing "problem-solving" in any satisfying way. On the contrary, the very spirit of heuristic knowledge is that it is not truly global or absolute, but instead specifies some set of circumstances in which this much more of something is that much better, *relative* to those circumstances. When actions are determined by using heuristic knowledge to compare results, the justifications for those actions, ultimately, can have no higher appeal than "It seemed like the best thing to do under the circumstances."

There are several different heuristics employed for qualifying differences in the commentary generated by QBKG. Three of them are used to generate preliminary remarks about the general game situation and the overall relative merit of the two moves, the others are used to judge differences in concepts within the Judgement Structure, depending on the particular operation involved in the concept. In all cases, the task is basically the same: to produce a set of "buckets" labelled with appropriate words and to fit a difference into a bucket that would seem appropriate to the competent backgammon player reading the commentary. The tasks and methods as of the writing of this paper are as follows.

[25] The approximation of the expected value is too crude to provide adequate discrimination between individual moves, however, thus motivating the "two function" scheme used in BKG.

4.2.1. Judging the absolute status of the game

QBKG's commentaries always begin with a statement of its opinion as to who's ahead in the game and by how much. This sets the stage for the explanation derived from the comparison of the moves, since which high level goals are relevant typically depend on who has the advantage and by how much. Two statements are made, one about how far ahead or behind the player is in the race, and one about whether the player is at an advantage or disadvantage.[26] The first statement is obtained by taking the difference in the pipcounts and bucketing the value according to the common backgammon dogma.

The second statement derives from QBKG's computation of the expected value of the game. The expectation computation is an admittedly crude piecewise approximation producing a number from -300 to +300 with 100 representing an almost certainty of winning the game, 200 for winning a gammon, 300 for winning a backgammon, and symmetrically for losing and the negative values. There are some seven or eight functions combined in the approximation, which does not use application coefficients to combine the heuristics. The expectation is not used in the Judgement Structure, so the blemishes in the computation do not impact the quality of the move selection. Outside of the explanation system, the expectation is used only for the Double/Don't double and Accept double/Reject double decisions, which do not involve the hypothesization machinery of the World Model and thus escape the clutches of the blemish effect.[27] Within the explanation system, while a human expert might occasionally quibble about the precise wording used (thinking "strong advantage", say, would be more appropriate than "very strong advantage" in a given position), the computation is accurate enough that the error is very rarely seen to be more than "off by a bucket."

4.2.2. Judging the relative overall quality of the two moves

After setting the stage, the commentary makes a statement about the relative worth of the two moves. QBKG, of course, played the best move it could find, so naturally it finds the suggested move to be, at best, not significantly worse than the move it played. This computation (Figure 4-1) is based on the heuristic values of the two moves in the context of the range of heuristic values possible for all legal moves. The

[26]These usually track each other, but not always; in a "back game" a player can be far behind in the race but have a respectable position.

[27]By comparison, if the program were designed to make such choices by hypothesizing both outcomes of a doubling decision and comparing the resulting expectations, the blemish effect would be a significant problem. As it is, QBKG's doubling decision performance is probably the weakest aspect of its otherwise strong play. With more work the expectation computation could be significantly improved, although a perfectly accurate function would constitute a solution to the game of backgammon and is probably not to be expected in the near term.

difficulty which the Goodness(c)[28] computation is addressing is that, while a given difference in heuristic value may represent a vast difference in quality in one set of circumstances, in another set of circumstances the given difference may represent a relatively small difference in quality. In some situations, when there are only a few relevant heuristics, the best move may be distinguished from a significantly inferior one by only a few points, while in a more involved situation a difference of dozens of points may be relatively unimportant.

Let $H_1 \ldots H_n$ be the heuristic values of all n legal moves sorted into decreasing order. Then H_1 is the heuristic value of the move that was actually made. Let H_c be the heuristic value of the move c suggested for comparison. For move i, define:

$$\text{WRank}(i) = i/\sqrt{n}$$
$$\text{Goodness}(i) = \text{WRank}(i)|H_1 - H_i|/\sqrt{|H_1 - H_{n/2}|}$$

Then for move c: Goodness(c) $< 0.2 \rightarrow$ "Actual move about as good as suggested move"
$< 0.8 \rightarrow$ "Actual move better than suggested move"
$\geq 0.8 \rightarrow$ "Actual move much better than suggested move"

Figure 4-1: Determining relative goodness of a comparison move

The computation uses the heuristic value of the median move as an estimate of the volatility of the position to obtain a notion of how valuable a point of heuristic value is in the current position. The WRank (weighted rank) function adds a measure of sensitivity to the number of moves that are better than the move suggested for comparison. The statistical rank of a move is modulated by a function of the number of possible moves to produce this measure; the intuition behind this is basically that it is better to be the second best of one hundred moves than second best of ten. In many cases, however, the number of possible moves is deceptive. Frequently, there will be several moves all of which address the major issues in the position and only differ in subtle ways, and a whole string of other moves of much lower heuristic value which do not. Taking the square root of the number of moves is a rough attempt to compensate for this long tail of terrible moves, so, for example, being second best of ten is considerably better than being 20th best of 100.

4.2.3. Judging the magnitude of a difference in a component of a SUM node

With the exception of the high level summations that are denoted collections and are discussed below, differences in concepts that are part of SUM nodes are handled as shown in Figure 4-2. The computation uses the LocalRange of a concept to estimate how much gain is possible in this concept in this situation, and

[28] Which is actually more of a "badness" function, since increasing values of Goodness(c) imply that the suggested move is increasingly poor with respect to the actual move.

Let: s be an element of SUM node that is not denoted a collection,

δs = value of s for the actual move − value of s for the suggested move,

LocalRange(s) = the difference between the highest and lowest values obtainable for s in the current position,

GlobalRange(s) = the difference between the highest and lowest values of s observed in the history of the Judgement Structure, and

SumDiff(s) = $2\delta s/(\text{LocalRange}(s)+\max(\text{GlobalRange}(s),50))$.

Then any SumDiff(s) < 0.15 is not worth mentioning, and
$> 0.5 \rightarrow$ "very much better",
$> 0.25 \rightarrow$ "much better",
$\geq 0.15 \rightarrow$ "somewhat better".

Figure 4-2: Determining significant differences in components of SUMs

the GlobalRange as an estimation of the general "value of points" for this concept.[29]

Let c be a collection. Define:

SigColl(c) = if $|\delta c| < 10$ then 0.0 else $|\delta c/\delta\textbf{Heur}|$

Then: SigColl(c) $< 0.25 \rightarrow$ insignificant; not worth mentioning
$< 0.50 \rightarrow$ "a significant factor"
$\geq 0.50 \rightarrow$ "a major factor"

Figure 4-3: Determining significant differences in collections

4.2.4. Judging the magnitude of the difference in a collection

The high level SUMs that are denoted collections are bucketed into just three groups: those which make an insignificant contribution to the overall difference in the value of the SUM and should not be mentioned, those which make a significant contribution, and those which make a major contribution. The collections in QBKG's Judgement Structure are such that they all contribute equally to the overall heuristic value, so the heuristic for qualifying differences in collections is based on computing the percentage of the δHeur which

[29]Bounding the GlobalRange below with a constant is clearly a hack, but it works surprisingly well for QBKG's particular Judgement Structure. The intuition behind it is simply that there are concepts that have a very small range of values. In most situations the contribution of those small factors is outweighed by other issues, but in close situations these "small things" can sway the decision. Bounding the GlobalRange in this way keeps the explanation system from being overly effusive about differences in such concepts.

δcollection accounts for.[30] The details of the computation are shown in Figure 4-3. Insisting that a δcollection be at least ten points of heuristic value in order to be mentioned is a generally conservative guideline, so QBKG tends to mention topics which a human backgammon expert would consider relatively unimportant much more frequently than it skips one an expert would consider critical (see for example Figures 4-6, 4-7, and the accompanying text.) Furthermore, in many positions some of the δcollections will have opposite signs, representing compensating aspects of the two moves (e.g., Figure 4-7), so that **δHeur** is significantly smaller than the sum of the |δcollections|, making it easier for a δcollection to make the significance cutoff.

The last two heuristics above, for handling SUMs and collections, perform most of the commentary generated by the explanation system, but occasionally differences in nodes using other operators crop up. (Note that due to the nature of the procedure for finding relevant issues, which terminates when *more than one* significant difference is found at a given node (Section 4.1), unary operators such as square and cube are never candidates for discussion.[31]) Each non-unary operator has a routine attached to it to perform explanation should multiple differences occur in its arguments; these are straightforward given the semantics of the operator. For example, the boolean IF operator has a routine capable of explaining that in one move a certain condition was true while in the other move it was false, and the differences in the values of the then-clause and else-clause account for the overall difference in the value of the concept.

4.3. Some examples

The QBKG system is now able to produce cogent commentary on about 70% of positions randomly presented to it. Its principal flaws at this point are idiosyncrasies in the knowledge base due largely to historical reasons. For example, the system is unable to comment on the relative merits of two moves where one move separates the opposing armies (creating a non-interfering race to the finish of the game) and the other move does not.

An example of its ability is shown in Figure 4-4, taken from a set of problems by Holland [9]. QBKG has chosen 17-24 as its move, and the user has asked for a comparison with 12-18,17-18. Holland comments on this position, "The correct play is to move one man from [the 17 to the 24 point], hitting Black's blot. You

[30]Thus, in Figure 3-1, **TactWgt**, **PosWgt**, and **PrimeWgt** all equal 1. The reason this is so is that when the evaluation function was originally constructed for the BKG program, the higher level structure expressed by the collections was not represented and the form of the **Heur** computation was the summation of some fifty terms, all of which had been scaled independently to relate directly to utility. When the needs of the explanation system for QBKG dictated that subdividing that set of terms into groups of related concepts was necessary for coherence, each term retained its direct relation to utility so that the application coefficients and scaling constants for the collections were unneeded.

[31]Although they may be mentioned in the course of an interaction if the user requests the definition of a concept represented by a unary operator.

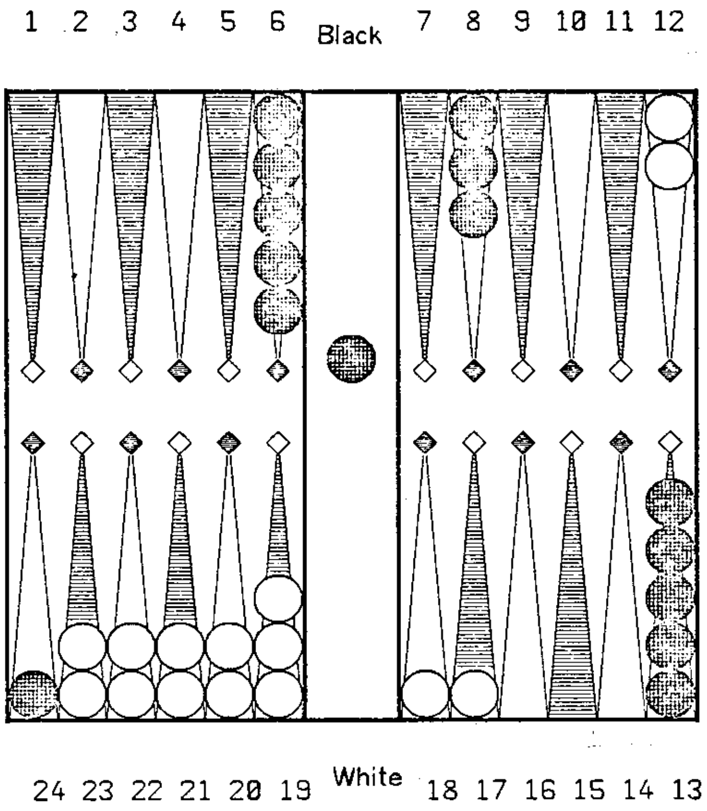

Figure 4-4: Sample position, White to play 6,1

```
(1)In the given position, White is far ahead in the race, and has
  a winning advantage, with substantial gammon chances.
(2)The actual move, 17-23,23-24(Move 1), is much better than the
  suggested move, 12-18,17-18(Move 2).
(3)There is nothing to recommend Move 2.The advantages of Move 1 are:
    o vastly better chances of keeping Black from making an
  advanced point [1].
    o very much better attack by White [2].
```

Figure 4-5: QBKG's commentary on two moves in Figure 4-4.

must try to prevent Black from establishing [the 24 point]. If you were to make [the 18 point] in lieu of hitting, Black would have 11 chances out of 36 to roll a 1, giving him a position from which he will still be able to win the game." (p. 66, paraphrased into QBKG's notation.) Figure 4-5 shows QBKG's commentary on this choice. Part (1) is some general comments about the situation, based on **PipDifference** and the independent expected value computation. Part (2) is QBKG's opinion on the worth of the two moves, based on δ**Heur** and some knowledge about the range of possible **Heurs** in this position. Part (3) is the result of the focusing mechanism discussed above and shows the extreme importance of hitting the lone Black man. The crucial issue of stopping Black from making an advanced point (**AdvPoint** in Figure 3-1) is discovered and reported, while irrelevant differences between the moves, such as the added risk that White may be hit (11 chances for Move 1 vs 1 for Move 2), are ignored. The bracketed numbers in part (3) are reference numbers by which the user may request further commentary on the specified topics. The system responds to such requests by recursively entering the focusing system using the selected topic as the root of the search, in the same manner as it handles topics that are denoted collections.

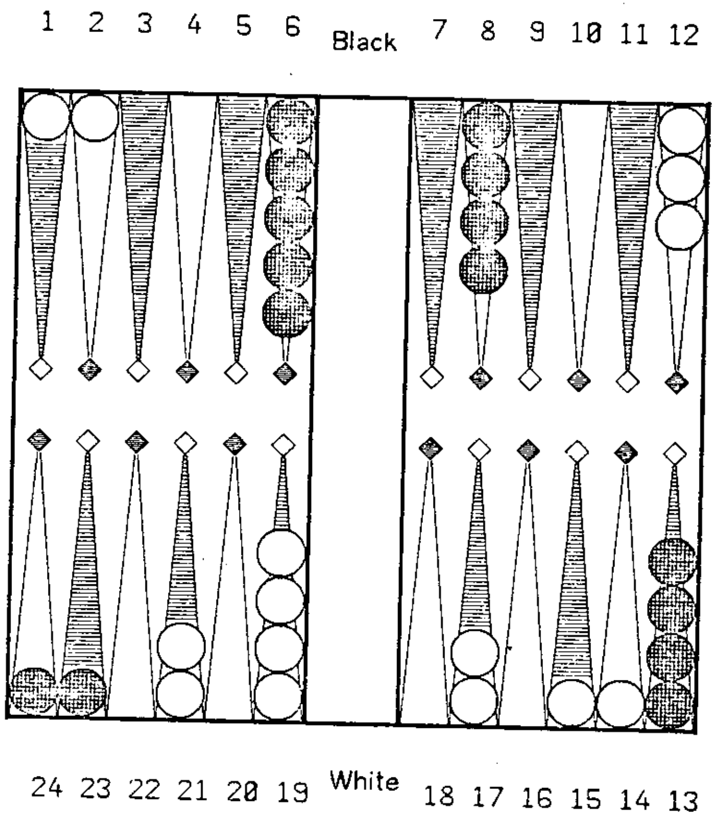

Figure 4-6: White to play 6,5

```
White is slightly ahead in the race, and had a strong advantage.
The actual move, 14-20,15-20(Move 1), is much better than the suggested
move, 1-7,2-7(Move 2).
Move 1 has the major advantage of chances in the blocking game [1]
Move 2 has the major advantage of positional prospects [2]

With respect to the chances in the blocking game :
 Move 1 has much larger actual containment of Black [3]
With respect to the positional prospects :
 Move 2 has these major advantages:
  o flexible position [4].
  o chances in various long range concerns [5].

With respect to the flexible position :
 Move 1 has much smaller White's mobility [6]
With respect to the chances in various long range concerns :
 Move 1 has much larger second moment of inertia [7]
```

Figure 4-7: QBKG's commentary on two moves in Figure 4-6

Another position from Holland's book is shown in Figure 4-6. In this situation, QBKG's commentary is not quite so crisp. On this position, Holland comments,

> The correct play is to make [the 20 point]. The tremendous value of having your [20 point] at this juncture of the game far outweighs any other move.
>
> You are putting pressure on Black's blots in your home board in two ways: first, with the start of a formidable blockade which restricts Black's forward movement and escape, and, second, with the threat of making additional points in your board which could put Black on the rim.
>
> Since there is no threat to your men in Black's home board at this moment, you should not consider either the play of making [the 7 point] or the move of one man from [the 1 point to the 12 point]. [9, p 5]

QBKG's comments, shown in Figure 4-7, correctly identify the key issue of improving the blockade and containing Black,[32] but they also spend a lot of time detailing the advantages of the suggested move, which Holland dismisses briefly. Since most of that discussion was produced as a consequence of "positional prospects", "flexible position", and "long range concerns" all being denoted *collections*, the error is basically in considering the "positional prospects" to be a major advantage for the suggested move rather than a relatively insignificant one.

In this situation δ**Heur** = 26, δ**BlockingGame** = 54, and δ**PositionalGame** = -24 (rounding to integers). Since δ**Heur** is so small with respect to the differences in those two components of the **Heur** collection, the collection qualifying heuristic (Section 4.2.4) judged them both to be major contributors even though one is twice as good as the other is bad. Once that misjudgement has been allowed for, the rest of the commentary due to it is more or less reasonable. The suggested move does provide a more flexible position since the "builders" on the 14 and 15 points remain in place and it brings the back men up to where they cannot be blockaded effectively.

The final statement of the commentary is accurate too, although it is expressed in somewhat an odd fashion. One general goal in backgammon, other things being equal, is to keep one's men relatively close together, to obstruct the opponent's progress effectively and allow one's men to advance from point to point without leaving many blots. In humans this "clumpiness" factor seems to be primarily a visual phenomenon and quite difficult to quantize. QBKG estimates the clumpiness of a position by considering the men to be

[32]QBKG has two distinct notions of containment. *Actual containment* refers to the containment of the enemy man, if one exists, that is most trapped behind a blockade. *Potential containment* refers to the containing power of the particular point on the board from which a hypothetical enemy man would have the hardest time escaping, whether there is a man on or behind that point or not. This distinction is necessary for good program performance but is not found in the literature.

point masses and the path of the men as a straight line, and then computing the second moment of inertia around the centroid. The suggested move brings up the two back men and decreases the second moment, leading to an improvement in the status of the long-range concerns.

5. A wider view

The distinction between the feedback-based paradigm of Cybernetics and the logic-based paradigm of symbol manipulation systems is interpreted as the distinction between focusing on judgement and focusing on reasoning. The centrality of the notion of **time to react** *is emphasized as the driving force for using judgemental methods in concert with reasoning methods. The difficulty of performing the signal-to-symbol transformation in a problem-independent manner is used as an argument for non-discrete knowledge representation.*

The QBKG system can be viewed as a game-playing program that has a fairly simple explanation system attached; that is what it is. It resembles in many ways all game-playing programs. What we intend to do here is to view the system in a more general way, as an example of a system which embraces both ends of a particular duality in the way that knowledge is represented and exploited. The duality of representation is the World Model against the Judgement Structure, and the duality of exploitation is search against evaluation. Our claim is that this duality is that of *reasoning* against *judgement*, and, for researchers in artificial intelligence, that understanding the nature of judgement is as important as understanding the nature of reasoning. We feel that the investigation of this paradigm, emphasizing the balancing of the two processes and the flow of knowledge between them, can be fruitful both as far as illuminating the nature of human intelligence and suggesting avenues by which, ultimately, much of the flexibility and power of the symbol-manipulation approach may be attained without the intolerable computation delays of combinatoric search.

The approach of this section is first to examine a bit of the history of artificial intelligence work in light of the reasoning-judgement paradigm, suggesting that the paradigm offers the hope of uniting two divergent avenues of research. Following that, we examine some of the consequences of this view in general and as they are manifest in QBKG.

5.1. Reasoning and Judgement

In the late fifties and early sixties, two avenues of research diverged. One direction was typified by Cybernetics [20], with the emphasis on feedback control mechanisms, homeostatic behavior, and information-transmission theory. The prototypical mechanisms were devices like thermostats, and the concerns were with obtaining proper real-time behavior, avoiding oscillations, and the like. For relatively simple machines operating in fairly simple environments, the Cybernetic approach did very well. Unfortunately, for more complicated situations, no such successes were forthcoming. It became clearer that mechanisms which the Cybernetic approach could analyze are essentially special-purpose devices, each designed expressly for its intended environment, and no one was able to develop a mechanism that could demonstrate anything like the generality of behavior which humans are capable of.

Around the same time, the notion of a *symbol manipulation* system was developed, typified by GPS [15]. A central concern of this enterprise was providing the generality that was lacking from the homeostatic, feedback-based designs. This desire for *universality* [16], meaning the ability for a single mechanism to provide any computable function of its inputs to its outputs, was satisfied by recasting the task of generating intelligent behavior in terms of *problem solving*, and applying the relatively well-understood principles of mathematics and logic.[33] With the symbol manipulation approach came two critical ideas: the notion of *searching a problem space* to determine behavior, and the need to *discretize* inputs and actions in order to produce well-defined symbols amenable to logic-based reasoning. It is precisely that discretization, which seems so intuitively safe and reasonable, that we are arguing the danger of in this report.

At the outset, concerns for real-time behavior were largely held in abeyance; to be replaced, effectively, with concerns that the symbol manipulation system be able to perform on satisfying problems when constrained only by the amount of time the experimenter is willing to wait. The method of demonstrating universality by proving Turing equivalence implicitly supports the notion that time is irrelevant, since it matters not how fast a simulation runs, so long as it runs properly. The argument, perhaps, was that we should discover what can and cannot be done in theory first, given no or few restrictions on resources, before we complicate matters by adding such limitations.

The success of the symbol manipulation paradigm was so great that it largely obliterated its competitors in most AI work. As the capabilities of symbol manipulation systems grew, so grew the desire to build systems that solve harder and harder problems requiring more and more knowledge. At the same time, it became clearer and clearer that researchers in AI could no longer assume indefinite resources of space and time, and nobody had much interest in a system unless it had the ability to produce results in some human time span (weeks or months, possibly, but not centuries). Increasing recognition of the fact that time constraints in particular are of tremendous practical significance has led some researchers to argue "Well, we know what we want do in *theory* now, and it tends to take almost forever, so let's figure out how to do it *fast*." Symbol manipulation systems have demonstrated themselves to be lovely, powerful castles in the sky, and people are beginning to work on building the foundations beneath them that will allow them to be useful *in real time*.

[33] It is of course understood that no physically realizable, and therefore finite, system can be truly universal in this sense. The force of the argument for universality is that time and space limits can be assumed to be large relative to the resource demands for significant problem solving, or, if they cannot, then at least the limitations can be dealt with as a separate problem.

This order of development, from mathematical grounds where time is of no consequence, to the empirical time constraints of implementations on a given architecture, to designing architectures expressly to meet time constraints, is interesting because it is exactly backwards to the order of evolutionary development that lead to human beings. In a way, it is an understandable reversal, since it emphasizes the universal, general purpose problem solving capabilities which humans alone are presumed to possess. It is only reasonable in studying the nature of intelligence, it would seem, to begin with that characteristic, call it *general reasoning ability*, that most decisively separates what we might consider intelligent from what we would not.

We wish to argue somewhat differently. While it is extremely important to investigate the abstract nature of reasoning ability, what it is, what it can and cannot in theory do, it is equally important to investigate the less well-specifiable questions of how systems can be made to function effectively in complex environments and how the *most generality can be bought for the least amount of time.*

In the state of nature, survival is the game, and the ability to act in real time is *the* paramount issue. The universality of a symbol manipulation system can usually be avoided by designing a special-purpose organism, via natural selection, to fit the environment. Thus, designing architectures expressly to meet time constraints is accomplished first, with no notion of *universality* being considered, in fact, just the opposite: in a stable environment, the form that is better adapted, i.e., that is *more specialized*, will perform more successfully.

The salient feature of the control of most of these special-purpose organisms is that they exploit *judgemental knowledge*. Provided genetically, and in many cases also fine-tuned by experience, judgemental knowledge in its simplest form is a set of associations between states of the world and actions of the organism. It is knowledge of *correlation* as opposed to *causation*. The advantages of correlative, judgemental knowledge are that it can be acquired easily, since it requires only a very limited internal model of the environment, and that it can be exploited quickly, in parallel, since it does not require the sequential chaining of cause to effect, and thus can provide guidance in time-critical situations. The disadvantage of judgemental knowledge, of course, is that it lacks flexibility in dealing with unfamiliar situations, whether time is of the essence or not: it has *limited* generality.

Judgemental knowledge is thus, conceptually, a set of stimulus-response associations, but to independently store such a set of associations between states of the world and actions would require far too much space. Instead of storing all the states of the environment which are to cause any given action, commonalities among those states can be extracted and used as evidence that the particular action is called for. Those commonalities which are most stable, which most effectively *discriminate* the states which call for a given action from the states that do not, will be maintained and improved. Over the course of generations (or system design iterations, depending on who is doing the designs), the judgemental apparatus becomes more and more effective at this task, until the discriminations become so sharp that, for a given environment, they amount to discrete distinctions.

5.2. A hybrid organization

The main point in all of this is that the universal symbol manipulation system, the reasoning system, possessed by humans is built directly upon a massive foundation of judgemental knowledge and apparatus for gaining and exploiting judgemental knowledge. This fact offers an answer to the too-rarely posed question "Where do symbols come from?" The answer offered, of course, is that symbols come from the discrimination-making ability of the judgemental apparatus. It is all well and good to build a system that can reason from inputs like "All men are mortal" and "Socrates is a man" to the conclusion that "Socrates is mortal", but until that system can itself look at Socrates and decide that he's a man, the system is still relying on the judgemental apparatus of the people providing the input.[34] This may seem like a small point; it may seem that it is "easy" or "bells and whistles" to add the ability to recognize, say, a man. We suggest that in general it is not, for reasons that are discussed in Section 5.3 and [4].

Humans can thus be viewed as possessing a *hybrid* structure, poised somewhere between a maximum speed strictly judgemental stimulus-response organization and a mathematically universal reasoning system where time is of no consequence. Most importantly, this hybridization is greater than the sum of its parts. The parts taken individually allow for quick reactions by the judgemental apparatus when time is critical and allow for success in unfamiliar circumstances when time is available.

Taken together, however, the judgement-reasoning hybrid has the potential for great improvements in capabilities through the interaction of the two modes. The critical step is the ability to move knowledge gained by the reasoning system into the judgemental system. This corresponds closely to the familiar notion of *chunking* [15] whereby common sequences of inputs (such as phonemes) or outputs (such as keystrokes) are eventually recognized or executed as single aggregate events (such as words). Although the chunking process speeds reactions by short-circuiting the slower, more general reasoning process when it isn't needed, it

[34]One common answer to the question "Where do symbols come from?" is that they arise fundamentally from the sensors and effectors of the system, so that, for example, something like "PIXEL#387542" would be a symbol that could be bound to the values ON and OFF. Then these primitive symbols are manipulated and combined via causal reasoning into more abstract symbols until ultimately statements such as "(OBJECT at MY-LEFT is a MAN)" are reached, and *then* the reasoning proceeds as before. This formulation is certainly elegant and is hard to argue with on time-independent grounds. However, on the grounds of efficiency, there is growing reason to doubt that such a uniform, causal mechanism as implied by this formulation could run with any speed. Our introduction of the notion of "judgemental apparatus" is aimed at limiting slow symbolic reasoning to high-level issues and allowing the "bookkeeping" to proceed at high speed. While it is still possible to view the combined judgement-reasoning system only as a symbol manipulation system, top to bottom, this obscures the important notion of judgement and the speed of the judgemental process.

has a far more important effect. The process of moving knowledge from the reasoning component to the judgemental component has the effect of *resymbolizing* the input to the reasoning system in different, more accurate, or higher-level terms. This helps to keep the size of the problems faced by the reasoning system down to a size where the familiar combinatoric explosion due to causal analysis is not an insurmountable obstacle.

Viewing a system of this type as a problem solver, this organization can be seen as an instance of the *generate and test* paradigm, but in a way that is strangely symmetric. The obvious form of the paradigm is that the judgemental system does the generation and the reasoning system does the testing. Thus if the problem to be solved is the interpretation of an image, the judgemental system provides an interpretation and the reasoning system checks for consistency. If the image is a familiar one, the job of the reasoning process may be reduced to rubber-stamp approval. In a more difficult case, perhaps when the objects are familiar but seen in an unusual configuration, the reasoning system may reject several candidates before reaching a satisfactory interpretation. In more difficult cases still, such as in the case of a mirage of a desert oasis, the system may end up "not believing its eyes", and only be able to produce the interpretation "Optical Illusion".

But the generate-and-test paradigm can be seen in the opposite direction as well, with the reasoning system being the generator and the judgemental system the tester. As an example, consider the syllogism in Figure 5-1.

Given	(1) If an elephant weighing x is big, then an elephant weighing x - 1 gram is big.
	(2) A ten ton elephant is big.
Therefore	A 1 gram elephant is big.

Figure 5-1: The Big Elephant Argument

What has gone wrong in that argument? More importantly, how can you immediately tell that something *has* gone wrong? In this case, the reasoning system has generated a conclusion that is tested by the judgemental system and rejected out of hand as being absurd. The problem arises because the quality of being *big* is not a discrete and well-defined quality. Thus, even though we feel that 1 gram is an almost insignificant amount relative to elephants, we'd want to claim that the elephant weighing x - 1 gram is not *quite* as big as the elephant weighing x, and therefore we would deny the validity of premise (1). But if (1) is invalid, there should be some value of x for which the antecedent is true and the consequent is false. Since none of us could distinguish between two elephants that differ only by a gram in weight, we would judge them both the same size, and we must conclude that there is no such value of x.

There are routes out of this dilemma, such as via various non-standard formulations of logic,[35] but the point to be made here is that the route that humans seem to follow is to *presume* the discreteness and well-definedness of symbols, use some form of discrete causal reasoning on those symbols to reach conclusions and make interpretations, and rely on the judgemental system to raise a clamor if the reasoning system goes astray because of that presumption.

5.3. The signal-to-symbol transformation

Systems that have been built for research into knowledge representation and problem solving have customarily lived in a world that is composed entirely of discrete, well-defined, symbolic entities. This is usually recognized as the simplifying assumption that it is. The further assumption, that the process of deriving symbols from non-discrete signals is (at least largely) a separable task from the symbol manipulation itself, is usually left implicit. The hope seems to be that this signal-to-symbol transformation can be performed adequately *outside* of the purview of the representation language itself. We wish to argue that this implicit assumption is not valid, and that discharging this assumption leads to a far more pervasive need for non-discrete representation than is generally acknowledged.

The main point to make about this assumption is that it only holds in two cases, both of which are in some sense degenerate. First, it holds in "toy domains", which can be interpreted as environments in which the signal-to-symbol transformation is a function, so there is never a question of one signal being transformed into more than one different symbol depending on circumstances. In the blocks world, for example, there is never any problem identifying an object as a cube or a cone;[36] there is never any need to use knowledge of the blocks world to form an interpretation of an object within the blocks world. In short, a system operating in a toy domain does not require judgement, since all judgements have been predetermined by the *creator* of the system. This is precisely what accounts for the well-known fragility of toy systems when faced with noise or uncertainty — the judgements provided by the creator are no longer reliable, and the system has no

[35] One of which has been to eliminate the apparatus of standard logic and replace it with "fuzzy set theory" [10], which discards the notion of all-or-none membership in a set and substitutes a *membership function* with values from 0 to 1 indicating degree of membership. (The notion of a membership function corresponds in some ways to that of an application coefficient.) The fuzzy set of big elephants, then, would assign a high degree of membership to a ten ton elephant and a slightly less but still high degree of membership to a ten ton less a gram elephant. The truth of the conclusion (i.e., the degree of membership of the conclusion in the fuzzy set of true statements) would be derived via the fuzzy inference procedures, and, for the example given, would hopefully be very small.

[36] We are referring here to the blocks world of, say, SHRDLU [21], rather than the line drawing blocks world of Waltz filtering [19]. In the latter case, this phenomenon appears in terms of edges and vertices.

capability for the performing the judgements itself.

The other case in which the assumption holds is when a symbol manipulation system is designed to operate on "real world" problems, but the signal-to-symbol transformation is performed by the *users* of the system rather than the system itself. As an extreme example, consider the game "Twenty Questions," wherein one player attempts to identify a secret object by asking yes-or-no questions. The questioner (analogous to a very discrete reasoning system) is constrained to two-valued inputs, and the answerer (the user of the system) must perform the transformation. Anybody who has played the game much will be familiar with the circumstance when the answer given is so marginally correct as to be misleading, as in the answer "No" to the question "Is it bigger than a breadbox?" when the object *is* a breadbox. If the secret to be discovered was the identity of an arbitrary disease in a patient, an overly discrete diagnosis system would have very few customers.

In such cases, the user and the program together form a hybrid system as discussed in Section 5.2, and the program is apt to run into difficulties of which the Big Elephant Argument is one simple example. A reasoning system which assumes the discreteness and well-definedness of symbols cannot escape this sort of trap, and it is behavior such as this which can lead the user to the feeling that "this program doesn't know what it's doing, or it wouldn't generate such absurd conclusions." The topic of human-machine hybrid systems is an interesting one, but it is a research arc somewhat different from the investigation of machine intelligence.

For systems which are to operate in non-toy domains, without the benefit of a human judgemental system to provide the symbolization and to judge and possibly reject the conclusions, the assumption that the signal-to-symbol transformation can be performed outside the scope of the representation language does not hold. Systems which have been designed not as general-purpose knowledge representation languages but as performance systems in complex domains have not been able to enjoy the luxury of such a simplifying assumption. In particular, the ability to use symbolic-level processes to guide the signal-to-symbol transformation is critical. For example, in the Hearsay [14] speech understanding system, syntactic and semantic information (symbol level) extracted from one portion of an utterance can guide the process of segmentation and phoneme extraction (signal level) in another portion of the utterance. Unless a representation language provides access to signal-level information, this cannot be done. If a language does provide such access, it must be able to express continuous quantities and continuous relations among those quantities.

Two possible objections should be noted. The first is the feeling that there might be a "natural" method of deriving symbols from signals that is (at least largely) independent of the particular problem, so that a representation could in theory provide this natural symbolization automatically with (little or) no knowledge of the task domain. This is somewhat of a straw man, since it rests so baldly on an appeal to some abstract notion of a distinguished "correct" interpretation of an environment *independent* of a system's goals, but it is worth mentioning just because it frequently lurks behind arguments that a given symbol manipulation system could be easily extended to operate in a continuous, noisy, real world domain. When the domain is sufficiently restricted there can be a simple, low-level transformation that loses no critical information, but as the domains become more complex the job of finding such a transformation rapidly becomes virtually impossible.[37]

The second objection grants that the method of extracting symbols does depend inextricably on the task domain, but argues that the signal-to-symbol transformation can be done cleanly at a relatively low level of processing, so that the bulk of the task-specific knowledge can ignore continuous quantities and speak in terms of discrete symbols only. The belief is that it is possible to provide a relatively small amount of knowledge that will "clean up" the environment for a particular task sufficiently that existing discrete symbol manipulation technology will then be able to handle the bulk of the information processing effectively. The problem in this case is that it may be arbitrarily hard to discretize a continuous value without making an error, in the sense that it may require arbitrarily high level knowledge to determine how to properly discretize a given value. Properly identifying a character in handwritten text, for example, could require extensive analysis of the meaning of the context surrounding the character.

Discretization always involves information loss, since many distinguishable values are mapped into a single discrete value. At the same time, and for the same reason, discretization simplifies the problem by reducing the size of the problem space. It cannot be determined *a priori*, nor by simple domain-dependent knowledge, when the information that is lost by discretization is irrelevant and when it is critical. The view we favor is the obvious consequence of this: *No unnecessary discretization!* Rather than discretizing the signals immediately and then composing higher and higher level symbols from the results, the signals themselves are transformed and composed mathematically into higher and higher level continuous values which are increasingly useful in determining behavior.

6. Limitations and extensions, summary and conclusion

The major weaknesses of QBKG both as a program and as an instance of a representation scheme are discussed with respect to the issues of proper behavior, making explanations, and questions of implementation. Directions in which the work could be extended are considered. The main points of the report are recapitulated.

[37] If there truly was a general-purpose "natural" signal-to-symbol transformation, that would imply, for example, that there would be some object *x* such that asking "Is it bigger than an *x*?" in Twenty Questions would never yield a misleading, boundary-case answer.

6.1. Weaknesses and limitations

It can often be quite difficult to separate implementation-dependent weaknesses from failings in a general method. For the QBKG system it is particularly troublesome since we have at present only the outlines of a general method — principles, guidelines, and intuitions — rather than a full-blown knowledge representation language and judgement/reasoning system. Accurate evaluation of the general method will have to await more research into making judgemental systems, but it is possible now to make some rough assessments of where some of the trouble spots are likely to be. This section first discusses problems with the QBKG system viewed mostly as a special-purpose expert system, and then considers difficulties for the more general model.

6.1.1. The QBKG system

As a backgammon player, the QBKG system is more than adequate. The weakest aspect of its play is in making doubling decisions, which is a direct consequence of the lack of a sufficiently high resolution and accurate computation of the expected value of the game. In addition, there are incompletenesses in the Judgement Structure. As the performance of the system improved, the knowledge needed to improve it further became more and more specialized, so the incremental improvement began to fall off. From a point of view of effective use of research time, improving the evaluation function further eventually reached the point of diminishing returns. As a consequence of this, QBKG will occasionally make the wrong move in a position, or make the right move for the wrong reason, and the commentary in those situations reflects that. (The explanation system, in fact, turned out to be a useful debugging aid, for just that reason.)

As a backgammon commentator, the QBKG system works reasonably well in many cases, displays a tendency towards verbosity a significant amount of the time, and very occasionally omits mention of an aspect of a position that most competent backgammon players, and ourselves, would have expected to be at least mentioned, if not prominently featured.

Overall, the main failings of the QBKG system as an expert system can be summarized as follows:

1. *Failures in qualification heuristics.* A major limitation which prevents the QBKG organization from being extended to a more general-purpose judgement making and explaining system is the lack of an adequate general-purpose statistics gathering mechanism. The current qualification heuristics are specifically designed for QBKG's particular Judgement Structure, and they succeed as often as they do, based on the very limited sort of minimum and maximum value information that is collected for each concept, only because of that. As they stand, there are occasionally circumstances in which the heuristics become quite unstable and disregard an important feature or become rather more excited about the importance of a feature than a human commentator would.

2. *Failures in the Judgement Structure.* There are things about backgammon that people know that

QBKG does not know. For example, the program has no explicit notion of contact among one's own forces. Having men and points close together allows for safer movement by increasing the chance of being able to move from point to point rather than leaving blots. A computation of the average distance from any man to the next point is an important measure of the contact of the men which is not in the current system. In QBKG, the only sensitivity to this issue is provided somewhat indirectly by the second moment of inertia computation.

3. *Failures in the Analysis System.* The commentary that QBKG generates covers only a small portion of the possible things that a human might find worthy of mention. Some of the sorts of things that QBKG cannot explain include:

   - Counterfactuals. When comparing moves, people will frequently make statements such as "Well, that move *would have been* better if I hadn't already gotten a man off, so you could try to gammon me. But since I have, ..." We considered some aspects of generating this sort of comment, and we implemented a function that would try to compute what values a set of application coefficients would have to have in order to make the suggested move better than the actual one. The results were inconclusive, but suggested that an absolute goodness metric would make such statements much easier to make.

   - Upward branching dependencies. In rare circumstances, the critical difference between two moves will be due to a change in a relatively low-level feature which is part of several ascendancy chains. In such a case, the method of finding relevant issues, which starts from the top of the tree, finds many genuine differences to report but never reaches the single difference which was responsible for them all. Given the organization of the Judgement Structure, the general task of finding such a "man behind the scenes" presents enough difficulties that, since the situation seemed unusual, we did not attempt an implementation.

   - Cancelling differences. The fundamental assumption of the explanation generating process, as discussed in Section 4, is that important differences between moves will be reflected by significant changes in the highest level concepts related to the difference. In one sense, this is obvious: how could a concept which has not changed significantly be relevant to a comparison? There are cases, however, where two moves will have cancelling differences so that a higher level concept shows no significant change. It could be argued that a comparison of moves, especially one meant for a non-expert listener, should mention such cases, because it may not be obvious how some perhaps obvious differences between the positions actually cancel out. This circumstance seemed rare, and so dependent on what seems obvious to the listener and what does not, that we decided not to worry about it.

4. *Failures in the reasoning system.* There are situations where the QBKG system could significantly improve its play if it could perform a little more searching. In situations where both sides have many ways to hit each other's blots, for example, exhaustively searching through one or perhaps two subsequent dice rolls and moves would provide valuable information that is difficult to capture in a static evaluation. The determination of whether any particular time is the right time

to double could also benefit from additional searching capability.

Another situation in which QBKG could benefit from a more powerful reasoning system involves the computation of the application coefficients. Most of the application coefficients are computed once per move, based on the current position, and then used as constants during the evaluations of the hypothetical subsequent positions.[38] They are computed under the assumption that at least one of the possible subsequent positions will be near the current position, in terms of the values of the application coefficients. This is the local linearity assumption discussed in Section 3.2.1. There are circumstances, however, in which the assumption fails. For example, consider a situation in which the application coefficients based on the current position indicate that the game is basically a flat-out race, and a major goal is to avoid being hit. For a given roll, however, it might be impossible to avoid leaving one or even two blots, resulting in a situation where the system is almost certain to be hit when the opposing player moves. Given that additional knowledge gained from an initial round of evaluations, a more powerful reasoning system could realize that running game considerations were becoming less relevant, recompute the application coefficients under the assumption that a blot hitting contest may ensue, and re-evaluate its options.

6.1.2. The general model

The difficulty of producing a computation for an absolute goodness metric for a complex domain stems from the conflicting needs of resolution and accuracy. In order to gain resolution, there must be significant heuristic knowledge available which is relevant in every possible position, and in order to gain accuracy, the strength of each heuristic (i.e., the values of its scaling constant) must be precisely tuned globally with respect to all the other heuristics, even those which are very far removed from the region of applicability of a particular heuristic. In complex problem spaces, this can be extremely difficult. An absolute goodness metric must have sufficient bandwidth to represent minute differences when the true utility surface is nearly flat and still have room to accurately represent the height of the highest mountains and deepest trenches. The route taken for the evaluation function in BKG and QBKG was to sacrifice overall comparability to gain sufficient resolution to usually pick the right move. The expectation computation used to make doubling decisions sacrifices resolution to gain overall comparability.

There are hopes that one may have one's cake and eat it too in this situation. For example, recent work [8] has shown that it is possible to derive an overall measure of the degree to which a set of constraints are being violated which retains comparability over the entire problem space. If a method can be demonstrated for

[38] In some cases, the features that would be used as application coefficients can vary significantly during the evaluations of the hypothetical subsequent positions. For example, the opponent's pipcount will vary when the moving player has a choice of whether or not to hit the opponent's blot, or which blot to hit if there are ways to hit more than one. This is handled by using the opponent's pipcount as of before the move as the application coefficient, and creating a separate concept, **GainInHitting**, which is *not* used as an application coefficient.

causing such a set of constraints to converge on an expected utility computation for a complex domain, in a reasonable amount of time, then the problem will be solved. Such a learning algorithm is not yet in hand, but work is progressing towards that end.

Another route around this difficulty, a familiar one in fact, from the proper vantage point, is to make do with less resolution in the evaluation function, and compensate for it with a more powerful reasoning system. Eurisko's extremal value heuristic discussed in Section 3.4 is an example of this. When the organism has complete control over which part of the problem space should next be explored, as Eurisko does when it designs spaceships, it can avoid a lot of relatively flat regions of the problem space entirely. In domains such as backgammon, when such flexibility is not available because the accessible regions of the problem space depend on where the system currently is and how the Real World decides to react, such knowledge can still supply likely landmarks to work towards. The various search and planning techniques that have been studied, such as means-ends analysis, can then be brought to bear on the task of getting there. Reasoning methods can compensate for defects in judgement, but only to a point, and only at a price. Unless the judgemental abilities are fairly sophisticated, and knowledge gained from reasoning can be converted into judgement, we feel the price of reasoning is so high as to be intolerable.

6.2. For further research

As is usual, the amount of work remaining to be done is much greater than that which has already been accomplished. Settling the question of whether or not absolute goodness metrics are the right things to look for, as discussed above, is one major issue bearing on the construction of any sort of "empty-QBKG" knowledge representation scheme. Some other issues which we have thought about in varying amounts are discussed in this section.

6.2.1. The quality of a judgement

The QBKG system has no notion of whether or not a position which it is evaluating is under the aegis of its known heuristics; for all possible positions in the problem space, the judgement structure returns a single value to the reasoning system: how good the position looks. For a system such as QBKG, with such severely limited reasoning capabilities, this is adequate, but a more sophisticated system would also want to know how much confidence the judgement structure has in the judgement that it has produced. If the position is an instance that has been seen many times before, the judgement structure should be able to evaluate it with confidence and say so, and if the position is particularly strange or unusual in some way, that should be reported as well.

One way to begin to provide such as confidence measure would be to examine the values of the application coefficients in the position. If they are mostly small or zero, the system has moved into a region of the problem space where its judgement will be of limited help.[39] When some of the application coefficients are near their maximum values, the system is on familiar ground and the confidence should be good.

There are difficulties with this good-sounding idea. For example, QBKG's judgement structure is complete enough that some application coefficients will always be significantly non-zero, yet there are situations when the quality of the judgements are suspect. This may, of course, be due to inadequate heuristic knowledge relative to human standards, rather than because of poor-quality judgement in spite of adequate knowledge. More likely, this derives from the fact that some application coefficients are weaker than others, in the sense that regions of applicability can be widely varying sizes, and that the largest, most general application coefficients, by themselves, do not suffice to make consistently good judgements.

Another factor which should be considered when determining the quality of a judgement is the shape of the local evaluation surface. When there is a strong slope in the landscape the judgements should be better than when the system is in, say, a very shallow bowl. Assuming an absolute goodness metric, the range of possible heuristic values in the states adjacent to the current state would provide a measure of the slope of the land, and thus provide information about the current quality of judgements.

6.2.2. Learning to make judgements

The process of learning to make judgements involves changing the evaluation surface to more accurately model the domain. There are, basically, two kinds of learning that must take place. The more immediate kind of learning is a "tuning" process, whereby an existing heuristic is improved by adjusting any of its components: adjusting the shape of the region of applicability by altering the computation of the application coefficient, adjusting the constant of proportionality, or (recursively) adjusting the computation of the lower-level heuristic from which the given heuristic is derived.

Although that is simple to state, it is not so easy to see through to an algorithm for performing these adjustments, since it requires solving the credit assignment problem for a complex, non-linear system. Furthermore, even given an instance where a heuristic is somehow determined to be faulty, it is possible to fix it up, for that particular instance, by tinkering with any of the components. However, unless the right changes are made, the surface as a whole may move away from what it is attempting to model rather than towards it. There are rules of thumb to help determine which component should be adjusted; for example, changing the constant of proportionality is likely to be the safest change to make, since that is the only quantity which is not potentially shared by other heuristics. Similarly, changing the shape of an application coefficient is likely to

[39] When building special-purpose systems, such as QBKG, one of the jobs of the designer is to ensure that there will always be some non-zero application coefficients everywhere in the problem space.

cause the most widespread effects.

The more open-ended sort of learning that must go on is the creation of new concepts — adding new heuristics, creating new goals to be accomplished, and defining new application coefficients. One strategy for accomplishing this could be a bottom-up approach relying on a generalization facility. In the most extreme case of a completely empty Judgement Structure, a move which leads directly to a won or lost position can be stored as a concept with a high positive or negative utility. Such concepts would proliferate and generalizations would be attempted to produce a more compact representation. The QBKG system could be made to perform an analogous, but more powerful and risky, sort of learning. Rather than waiting until a won or lost position is reached, the system could watch for large changes in the value of the expectation computation, and use that as a weaker indicator of good or bad behavior. Over the course of many learning trials, one could hope that the vagaries of good and bad dice would cancel out, leaving a set of positions from which both tuning adjustments and generalizations could be computed. If the expectation computation is systematically biased, of course, this could lead the system away from proper behavior rather than converging. The final court of appeal is always and only won and lost positions.

Such a learning scheme involves relatively little reasoning, and is roughly analogous to the judgement-improving scheme discussed at the end of Section 5.1. A system with a powerful reasoning system could learn much faster than a system without, because the adjustments to be made when the system misjudges a situation can be computed more specifically, even given only a small number of instances of the problem, using cause-and-effect hypotheses to "diagnose" the failure.[40] A reasoning-weak system must exploit the statistical properties of large numbers of instances to separate the culprits from the innocent bystanders, using correlative, rather than causative, methods.

6.2.3. Network organization

There are circumstances in which it would be useful to have downward and lateral flow through the judgement structure as well as the upward flow of evaluation. For example, one of the bugs that had to be ironed out of earlier versions of the evaluation function involved simultaneously using an input in two inconsistent ways, such as counting a given man as part of an attacking group and as part of a blockade formation, even though the man couldn't possibly be both. By careful construction of the evaluation function, such anomalies have been eliminated, but sometimes only by paying the price of fixing a given man's function using local heuristic rules.

A better solution might incorporate relaxation methods, allowing alternate interpretations of inputs and

[40] Assuming, of course, that the general structure of the assumptions already matches the environment fairly well.

middle level heuristics to compete with each other and settle down to a harmonious global interpretation of the inputs. Such an organization would be more in the spirit of our "No unnecessary discretization" motto because it would allow relevant knowledge at all levels to be brought to bear on the apparently "low-level" task of interpreting the function of a given man.

6.2.4. Architectures for hybrid systems

We have suggested that systems which are competent at *both* judgement and reasoning, rather than just one or the other, are likely to be the most successful problem solvers, but we have said little about how such a fruitful union might be created. Since we are committed to recognizing computation time as a primary design constraint, we need an architecture of some sort which will support both reasoning and judgement; one that provides flexibility for the reasoning system, speed for the judgemental system, and a mechanism for moving knowledge gained in the reasoning system into the judgemental system. The question of how to implement a fairly general-purpose World Model mechanism, which was side-stepped in the QBKG system, must be faced directly. In this section we speculate on what such a hybrid architecture might look like.

One possible implementation for a World Model would be a constraint-satisfaction parallel network such as the kind proposed in [8]. Nodes in the network represent hypotheses about the Real World, and links between the nodes represent knowledge about the likelihood of the two hypotheses being true simultaneously. A subset of those nodes would correspond to the sensors of the organism, and thus would be true, initially, if and only if the measurements of the Real World by the sensors indicates that they should be.[41] The rest of the network, corresponding to the interpretation of the portion of the Real World which is not directly observable, would settle into states which are (hopefully) maximally consistent with the observed data.

An action of the system could be defined by a set of hypotheses and truth-values which those hypotheses are to take as a result of the action. The determination of whether an action is possible at any given time, and thus the computation of the Action Set, could be performed by checking whether any of the hypotheses associated with the action are inconsistent with the current state of the hypotheses in the World Model, in the sense that they would cause a "practically impossible" link to be asserted or a "practically mandatory" link to be denied. In such cases, the corresponding action would be not be considered as part of the Action Set. (This is not totally satisfactory, since the inconsistency might arise as a consequence of a number of "fairly unlikely" links rather than a single "impossible" one. Actually setting the values of the action-hypotheses and

[41] We are systemically suppressing the probabilistic aspect of these networks here, for simplicity's sake, leading to a very discrete-sounding architecture. In fact, the probabilistic nature of this kind of network is critical to our enterprise, since it provides the non-discrete "degrees of confidence" which we require for judgement.

measuring the change in the overall consistency of the network would be more reliable, but there are difficulties there as well.)

The Judgement Structure, then, uses the states of the hypotheses in the World Model as input and produces a notion of utility for the situation. For purposes of exposition, it suffices to imagine the Judgement Structure as being implemented in the obvious manner, with a processor for each node computing the value of the node and collecting statistics on the computed values. A single judgement would be performed, assuming no pipelining, in time proportional to the height of the structure: approximately the log of the number of heuristics in the knowledge base.

A reasoning system at the level of the one in QBKG could then be implemented. It would hypothesize possible actions, allow the World Model to settle, apply the Judgement Structure, and perform the most favorably judged action. The Real World would respond to the action, leading to a new set of observable data, and the process would repeat. If the Judgement Structure produced utility on an absolute scale, the architecture would also support searching more than just one ply away from the current position.

To move knowledge into the judgement system, it is necessary to extract it somehow from a tree search or other reasoning process. Some of the techniques explored in belief revision research (for a selected survey, see [6]) could be of use to hypothesize cause-and-effect relationships, leading to heuristic rules for incorporation into the Judgement Structure.

Open issues with respect to such an architecture include the following:

The issue of meta-knowledge: in order to be effective, the system must have the "reflective" ability suggested in our model by the Analysis System. For best learning performance, a system needs to be able to judge not only the quality of a state in the problem space, but also the quality of its own reasoning processes. Some mechanism for representing the system's reasoning processes in the World Model might allow existing judgemental machinery to be applied to this task.

The issue of plausible move generation: how to use judgement to *propose* avenues to explore rather than only *selecting* among alternatives produced by a knowledge-poor mechanism like exhaustive search. This impacts the problem of how to produce the Action Set which was mentioned above. In a sense, this suggests the notion of a stimulus-response organization as a component of an entire system — using judgemental apparatus to produce codings for actions rather than codings for utility.

6.3. Summary of main points

- Few-valued knowledge representation is useful because it provides a great reduction in the size of a problem space by treating many states of the space as though they were equivalent, and it allows for the creation and testing of discrete cause-and-effect hypotheses. Few-valued knowledge representation is susceptible to serious errors because the binary or few-valued assumption is usually only an approximation, and with boundary cases the "round off" error can accumulate and lead to absurd conclusions. One way to gain the benefit of few-valued reasoning without the risk of catastrophic error is to add judgemental knowledge. (Sections 1.3, 1.4, 3.4, 5.2, 5.3)

- Many-valued, non-discrete representation is useful because it provides sensitivity to subtle differences which would be lost in a few-valued representation. In many cases the subtleties are unimportant, but in certain inevitable circumstances they are critical. By maintaining a fine-grained representation the smaller factors are considered in proportion to their importance. (Sections 1.3, 1.4, 3)

- Reasoning is considered as the process of perturbing an internal world model. A world model is composed of abstract symbols, probably in a fairly discrete representation, which approximates effects from causes. The major job of the world model is maintaining the consistency of the mapping between the internal representation and the observed behavior of the environment. Perturbing the world model allows for all sorts of reasoning, mathematics, and the simulation which demonstrates Turing equivalence. (Sections 1.4, 2.1, 5.1)

- Judgement is considered as the process of interpreting a state of the world model with respect to a goal. A judgement structure is a hierarchy of heuristic rules which approximate the goodness or utility of a problem state. Good judgement is fundamentally non-discrete in that it is sensitive to the small differences between problem states which a few-valued representation would ignore. (Sections 1.4, 2.1, 3, 5.1)

- Reasoning alone is powerless since judgement is required to differentiate the utilities of possible alternate paths of action. Reasoning with too little judgement is fragile in the face of error or uncertainty. With good judgement, however, reasoning is capable of great flexibility of behavior, by virtue of the assumption of cause and effect. (Sections 3.4, 5.2, 6.4)

- Judgement alone has limited generality since it is tailored to a particular environment and goal hierarchy. Without a world model and a reasoning system, judgement degenerates into a (possibly highly sophisticated) stimulus-response organization. With a very basic reasoning system, and in a domain that allows for a simple world model, the judgement structure of the QBKG system displays behavior rivalling human performance. With a powerful reasoning system capable of retailoring a judgement structure as needs change and knowledge is refined and extended, a system may be able to show high levels of performance on diverse problems without intolerable computation delays. (Sections 5.2, 6.4)

- Heuristic knowledge can be viewed as a three-way relationship between a context and two values which can be approximated as having a linear relationship within that context. It is critical in judgement structures to avoid hard boundaries between regions of applicability of heuristic knowledge. Application coefficients provide a method of combining heuristic knowledge so as to effect smooth transitions between regions in which differing heuristics are relevant. (Section 3)

- The process of explaining a judgement involves comparing the values that the nodes of the Judgement Structure have in one problem state with the values that they have in another, and isolating a small set of nodes whose differences largely account for the difference in the overall utilities. By searching the structure from the heuristic value node at the root, continuing down until more than one significant difference is found in the descendants of a node, the explanation begins at the lowest level of discourse that is high enough to encompass the key differences between the two problem states. The differences deemed relevant are quantized into a small number of magnitude classes. Since the importance of a given size difference depends upon the circumstances in which it appears, the bucketing process is performed by classification-specific heuristic rules. An absolute goodness metric for an entire problem space would simplify such classifications greatly, but may be too much to hope for. (Section 4)

6.4. QBKG and the tradeoff of reasoning and judgement

The basic tradeoff between search and knowledge in heuristic problem solving is well understood. Additional knowledge allows for less searching because fewer blind alleys are followed. Additional search allows for less knowledge since eventually something other than a blind alley will be found. This distinction between knowledge and search is equivalent to that between judgement and reasoning.

The straightforward stimulus-response organization represents one extremal point in the tradeoff, where no reasoning is performed and the behavior of the system is entirely governed by judgemental knowledge. In such systems, the judgement structure produces a coding for an action, rather than an estimation of the utility of a state of affairs. The price of such an organization is inflexibility.

At the opposite end of the spectrum, at the "pure reasoning" end, one might find systems such as brute force theorem provers. Containing only the knowledge of well-formedness of formulae and proofs, such a system can only describe states of the problem space as "goal" or "non-goal". The price such a system must pay, of course, is time.

Apparently, the true evaluation surfaces of the environments we wish our systems to function in, and indeed the environment we all exist in, are sufficiently smooth and balanced with respect to time and complexity that we will not often find optimal solutions at the extremal points.

The investigations into machine game-playing, particularly chess, have made significant incursions in the much-search little-knowledge end of the spectrum. The effectiveness of alpha-beta search, for example, is one significant datapoint extracted from such investigations, one that is useful regardless of how much judgement the system possesses. When a system is knowledge-poor, however, alpha-beta search is critical.

QBKG represents a point in the much-knowledge little-search end of the spectrum. The danger of the blemish effect, for example, is one significant datapoint extracted from this line of investigation, one that is useful regardless of how much reasoning the system performs. When a system is reasoning-weak, however, a smooth evaluation surface is critical.

The parallel structure of the two previous paragraphs is not gratuitous; we feel the analogy is deep. Explorations of the near-extremal points of a dimension often provide valuable insights into the nature of the evaluation surface, and allow an understanding of the problems that every system has to face to some degree or another.[42] The search vs knowledge dimension has long been recognized, but the knowledge end has until recently remained largely unexplored.

References

[1] Berliner, H. J.
Some Necessary Conditions for a Master Chess Program.
In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, pages 77-85. 1973.

[2] Berliner, H. J.
On the Construction of Evaluation Functions for Large Domains.
In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 53-55. 1979.

[3] Berliner, H. J.
Backgammon Computer Program Beats World Champion.
Artificial Intelligence 14(2):205-220, September, 1980.

[4] Berliner, H. J., & Ackley, D. H.
The QBKG System: Generating Explanations from a Non-Discrete Knowledge Representation.
In *Proceedings of the National Conference on Artificial Intelligence.* AAAI, 1982.

[5] Davis, R., Buchanan, B., & Shortliffe, E.
Production Rules as a Representation for a Knowledge-Based Consultation Program.
Artificial Intelligence 8, 1977.

[6] Doyle, J., & London, P.
A Selected Descriptor-Indexed Bibliography to the Literature on Belief Revision.
SIGART Newsletter (71):7-23, April, 1980.

[7] Haley, P., Kowalski, J., McDermott, J., & McWhorter, R.
PTRANS: A rule-based management assistant.
Technical Report, Carnegie-Mellon University Department of Computer Science, 1983.
In preparation.

[8] Hinton, G. E., & Sejnowski, T. J.
Optimal Perceptual Inferences.
In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* Washington, D.C., June, 1983.
To appear.

[9] Holland, T.
Better Backgammon.
Reiss Games, Inc., New York, 1974.

[10] Kandel, A., & Lee, S. C.
Fuzzy switching and automata: Theory and applications.
Crane, Russak, New York, 1979.

[11] Lenat, D. B.
The nature of heuristics.
Artificial Intelligence , 1982.

[12] Lenat, D. B., Sutherland, W. R., & Gibbons, J.
Heuristic Search for New Microcircuit Structures: An Application of Artificial Intelligence.
The AI Magazine 3(3), 1982.

[13] Lenat, D. B.
Learning Program Helps Win National Fleet Wargame Tournament.
SIGART Newsletter (79), January, 1982.

[14] Lesser, V., Fennell, R., Erman, L., & Reddy, D. R.
Organization of the Hearsay-II speech understanding system.
IEEE Transactions on Acoustics, Speech and Signal Processing ASSP-23:11-23, 1975.

[15] Newell, A. & Simon, H. A.
Human Problem Solving.
Prentice-Hall, Englewood Cliffs, N. J., 1972.

[16] Newell, A.
Physical Symbol Systems.
Technical Report, Carnegie-Mellon University Department of Computer Science, March, 1980.
Paper given at the La Jolla Conference on Cognitive Science, 15 Aug 1979.

[17] Rosenbloom, P. S.
A World-Championship-Level Othello Program.
Technical Report, Carnegie-Mellon University Department of Computer Science, August, 1981.

[18] Samuel, A. L.
Some Studies In Machine Learning Using the Game of Checkers.
IBM Journal of Research and Development 3(3), 1959.
Also an update in *IBM Journal of Research and Development* 11(6), 1967.

[19] Waltz, D.
Understanding Line Drawings of Scenes with Shadows.
In Winston, P. H. (editor), *The Psychology of Computer Vision*, . McGraw-Hill, New York, 1975.

[20] Wiener, N.
Cybernetics, or Control and Communication in the Animal and the Machine.
Wiley, New York, 1949.

[21] Winograd, T.
Understanding Natural Language.
Academic Press, 1972.

[22] Winston, P. H.
Learning Structural Descriptions from Examples.
In Winston, P. H. (editor), *The Psychology of Computer Vision*, . McGraw-Hill, New York, 1975.