Explaining Quantitative Scheduling Systems to Uninitiated Users

Susan A. Slotnick, *Cleveland State University*
Johanna D. Moore

Pergamon

Expert Systems With Applications, Vol. 8, No. 4, pp. 475–490, 1995
Copyright © 1995 Elsevier Science Ltd
Printed in the USA. All rights reserved
0957-4174/95 $9.50 + .00

0957-4174(94)E0037-U

Explaining Quantitative Systems to Uninitiated Users

SUSAN A. SLOTNICK

W. Averell Harriman School for Management and Policy, State University of New York at Stony Brook, Stony Brook, NY

JOHANNA D. MOORE

Department of Computer Science and Learning Research and Development Center, University of Pittsburgh, Pittsburgh, PA

Abstract—*The importance of explanation in expert systems has been documented from the early days of their development; there is an equally pressing need for explanation in systems that employ a decision-making process based on quantitative reasoning. This is particularly necessary for users who do not have a sophisticated understanding of the formal apparatus that the system employs to reach its decisions. In order to generate meaningful answers to questions asked by such unsophisticated users, an explanation facility must translate the formal structures of the problem solving system into the concepts with which the user understands the problem domain. Previous work on the explanation of quantitative systems is based on the assumption that the user has at least a basic grasp of the formal approach of the problem solving system. However, in realistic application situations, it is more likely the case that in order for the human user to understand why a mathematically-based advice-giving system makes the suggestions that it does, the problem solving rationale of the system must be explained in the user's own terms, which are typically different from those of the mathematical system. To develop an explanation methodology that is capable of justifying the results of a system based on quantitative reasoning to an uninitiated user, we employ a representation that enables our explanation facility to translate the abstract mathematical relationships that make up a quantitative system into the domain-specific concepts with which a typical user approaches the problem solving task. In our system, the process of generating explanations, therefore, involves translating one set of concepts into another. An added feature of this system is that it is capable of providing explanations from two perspectives: that of the quantitative problem solving system, and that of the human user who is familiar with the domain problem but not with the mathematical approach. We have implemented this approach to explaining quantitative systems by creating an explanation facility for a problem in the manufacturing domain. This facility responds to user queries about a scheduling system that uses a mathematically-based heuristic to choose jobs for an annealing furnace.*

1. INTRODUCTION

IN ADDITION TO its primary function of providing decision support, an advisory expert system (such as an automated factory-floor scheduler) must be able to explain its reasoning and justify its conclusions. The importance of explanation in expert systems has been documented from the early days of their development (Buchanan & Shortliffe, 1985). There is an equally vital need for explanation in systems that employ a decision-making process based on quantitative reasoning. By "quantitative reasoning" we mean reasoning that is based on mathematical models; this includes mathematical heuristics as well as provably optimal methods. This is essential in particular for users who do not have a sophisticated understanding of the mathematical apparatus that the system employs to reach its decisions. In order to generate meaningful answers to questions asked by such "unsophisticated" users, an explanation facility must translate the formal structures of the quantitative system into the conceptual framework with which those users understand the problem domain.

Requests for reprints should be addressed to Susan A. Slotnick, W. Averell Harriman School for Management and Policy, SUNY Stony Brook, Stony Brook, NY 11794. E-mail: slotnick@fac.har.sunysb.edu

Quantitative problem solving systems are a fact of life in the arena of real-world applications, in fields such as business forecasting, inventory control, planning, and scheduling. While expert systems are growing in number, the majority of applied decision support systems in business and industry are still based on quantitative problem solvers. In addition to the acceptability that comes from the fact that the quantitative approach has been used in these fields for a few decades, these systems also have certain advantages over expert systems, which typically rely on heuristic problem solving knowledge obtained from human experts. Some formal models give provably optimal solutions. In addition, quantitative models are usually more robust, and are often easier to implement, because they do not require the extensive knowledge engineering that is typical of the development of knowledge-based systems. One distinct disadvantage, however, is the common complaint that the uninitiated user, who is familiar with the domain but not with the mathematical subtleties of the quantitative system, will be reluctant to use a decision-support aid that he or she cannot understand. In fact, there are academic systems sitting on university shelves that might be of practical use if they could be reliably explained to their potential users. Moreover, many real-world problems require a combination of knowledge-based problem solving and quantitative reasoning (e.g., to deal with tradeoffs and uncertainty) (Buchanan & Shortliffe, 1985; Cooper, 1984; Langlotz, Fagan, Tu, Sikic, & Shortliffe, 1987), and therefore, techniques for explaining quantitative reasoning are crucial for a wide range of real-world application systems.

Previous work on the explanation of quantitative systems ranges from simple approaches that arrange numerical output in lists or tables (e.g., Ben-Bassat, Carlson, & Puri, 1980; Spiegelhalter & Knill-Jones, 1984) to more ambitious attempts to mediate between the quantitative and qualitative worlds (e.g., Ackley & Berliner, 1983; Cooper, 1984; Langlotz, 1989). However, these approaches have two basic limitations that make them unsuitable for explaining the results of quantitative systems based on sophisticated mathematical models to users not familiar with the mathematical formalism. First, common to almost all of these systems is the assumption that the user has at least a basic grasp of the formal approach that is employed by the problem solving system being explained. Thus, even the systems that attempt to provide qualitative explanations of numerical results do so in terms that are not likely to be understood by users who are not familiar with the formal model. Second, as is the case for early rule-based expert systems (Swartout & Moore, 1993), these facilities generally cannot provide meaningful justifications of their actions to users, because they do not have a representation of the abstract problem solving model that the user understands.

It is these two problems that are the focus of our research. We develop a representation that enables an explanation facility to translate the abstract mathematical relationships that make up a quantitative system into the domain-specific concepts and strategies with which a typical user approaches the problem solving task. We begin with the assumption that, in order for the human user to understand why a mathematically-based advice-giving system makes the suggestions it does, the problem solving rationale of the system must be explained in the user's conceptual framework, which is typically different from that of the mathematical system. Since our explanation facility uses a representation of problem solving (rather than, say, textbook knowledge) for two models (one quantitative, one based on human expertise) that have different structures, the process of generating explanation involves translating one set of concepts into another. The kind of system that we develop will ultimately be able to provide explanations from two perspectives: that of the quantitative problem solving system, and that of the human user who is familiar with the domain problem but not with the mathematical approach.

Our approach differs from previous work in two important ways. First, we do not assume that the user has a sophisticated understanding of the workings of the quantitative problem solving system. This distinguishes our approach from that of QBKG (Ackley & Berliner, 1983), which explains the strategies of an automated backgammon player. While QBKG converts numerical values into qualitative statements about the moves it chooses to make, it need not explain the heuristics that are embodied in the quantitative system, since the assumption is made that these are known to backgammon players (for example, the idea of keeping one's men relatively close together in order to impede the opponent's progress and enhance one's own movements). In contrast, we assume only that the user understands the domain problem. We do *not* assume that the user is familiar with the strategies used by the mathematical model to solve that problem. The user may have his or her own set of problem solving strategies, not necessarily isomorphic to those of the system. Thus, the decisions of the mathematical system that we are explaining must be translated into the concepts familiar to the user.

Our explanation facility also differs from earlier work that deals with the issue of explanation from different "perspectives," in that we are representing two different problem solving approaches, not just different levels or views of the same body of knowledge. This is why we cannot use an approach based on "marking" different parts of a knowledge base to indicate different perspectives, and then selecting those entities or relations that are marked in order to produce explanations for different types of users or to achieve different communicative goals. We are not dealing with an encyclopedic body of facts that needs such a strategy to constrain search (Souther, Acker, Lester, & Porter, 1991), nor with a single knowledge base that incorporates different viewpoints (Cohen, Jones, Sanmugasunderam, Spencer, & Dent, 1989; McKeown, 1988), nor with two models of the domain whose differences point to misconceptions that need to be corrected (McCoy, 1989). Our two problem solving models consist of different concepts and have different structures, and so we must represent them separately, and provide a way of translating between them in order to generate meaningful explanations.

We apply this approach to explaining the decisions of an automated scheduling system that uses a mathematically-based heuristic to schedule a factory floor. The innovation of this work lies in the way that it moderates between the conceptual worlds of the mathematical system and that of the human scheduler. The mathematical scheduling system considers some of the same factors used by the human scheduler, but also employs mathematical concepts and procedures that have no counterpart in the conceptual world of the human user. The explanation facility that we have developed, called Quantitative EXplainer (QEX), is intended to facilitate the use of this scheduling system by a human being on the factory floor who welcomes the assistance of an automated scheduler, but is more likely to use such a system if it is capable of explaining its suggestions in nonmathematical concepts and terminology.

It is important to note that the feasibility of an explanation system that uses this kind of translation depends on the characteristics of and relationship between the two conceptual worlds. If the two worlds have no concepts at all in common, then this kind of explanation is not possible; at the other end of the continuum, two isomorphic worlds render this approach unnecessary. It is on the middle ground, where there are two different structures with some mutual components, that systems like QEX can make a contribution. Even though there will always be some mathematical concepts that cannot be imparted to the user, because they do not have close equivalents in that user's conceptual world, we claim that it is both possible and worthwhile to do this kind of explanation.

For the QEX approach to be feasible and useful, the nature of the relationship between the two conceptual worlds must have the following properties. First of all, the two problem solving approaches must share the same top-level goals. If this is not the case, then it does not make sense to translate between lower-level goals because the two systems are in essence solving different problems. Second, it follows that most if not all of the primitive concepts (e.g., processing time, length of queues, or number of machines set up for a certain product in our manufacturing domain) should exist in both worlds, because they are characteristics of the problem itself. Third, in the ideal case, every concept in the quantitative system that is not directly translatable into the human's conceptual world can be broken down (at some level) into components that are translatable. If this is not the case, then there are some justifications of the decisions of the quantitative system that cannot be imparted to the nontechnical user. In contrast, the existence of user's concepts that are not shared by the mathematical system has a less deleterious effect on the explanations generated; though the user will never have the satisfaction of seeing these particular reasons in an explanation, there is no loss of information about what the mathematical system actually did.

Our system is similar to the REX system developed by Wick and Thompson (1992). REX decouples the "line of reasoning" used by the expert system to solve the problem from the "line of explanation" used to justify the system's conclusion. Given the result produced by the expert system and a set of "reasoning cues" that represent key data and inferences used by the expert system during its problem solving, the explainer uses a separate body of "explanatory knowledge" to "reconstruct" an explanation that leads from the initial data to the final conclusion. That is, the explainer re-solves the problem using its own knowledge base. By varying the "reasoning cues" provided, the degree of coupling between the expert system's reasoning and the justification of the conclusion can be varied. An advantage of this approach is that during reconstruction the explainer may add additional evidence for the conclusion that was not used in the original problem solving.

However, we see three aspects of the REX approach that make it unsuitable for our current purposes. First, it requires that two complete problem solving systems be built. In our case, the scheduler's model represents, in successive levels of detail, the factors that contribute to a conclusion and relationships between them. The model is not a complete problem solver. Second, depending on the number of reasoning cues provided, REX's explanations vary between being a paraphrase of the expert system's line of reasoning (completely coupled) to a ratification of the expert system's conclusion (decoupled). In QEX, we are interested in *translating* the quantitative system's reasoning into a form the human scheduler understands. Finally, the explanation knowledge base in REX differs from the expert system knowledge base in that it includes idealized strategies for solving problems, deeper knowledge of the domain, and alternative strategies that are not necessary for the expert system's problem solving. Thus, it is a variant of the same conceptual model used by the expert system, that is, REX makes the assumption that the user's view of the domain has much in common with that of the expert system. In QEX, the scheduler's model represents a different conceptual framework, and therefore, a translation process is necessary. We believe that a REX-style system could be used for our task, but that such a system would still require the type of mapping that we provide in QEX in order to translate expert system concepts and relations into concepts and relations in the explanation knowledge base. Thus, the approaches are complementary. In future work, we wish to investigate extending the REX approach for our task.

2. THE DOMAIN PROBLEM AND THE QUANTITATIVE SOLUTION

2.1. The Factory Floor Scheduling Problem

QEX was developed in order to explain a system that produces factory floor schedules for the Westinghouse Specialty Metals Plant in Blairsville, PA, which produces fuel and instrument tubes for fuel assemblies of nuclear reactors. These tubes are processed in three steps. The first is a cold-extrusion step, in which a tube is reduced in diameter and lengthened by pushing it through a metal die. In the second step, the tube is sent to an acid bath to cleanse it of grease and other impurities. And in the third step, it is annealed in a furnace in order to reduce brittleness, restore molecular stability, and prepare it for the next stage of extrusion. These three steps are repeated a number of times, until the tube meets its final specifications. Each combination of the three steps is referred to as a *pass*.

Scheduling tasks in this factory include decisions about machine setups, release of raw material, and scheduling of jobs on the various types of machines. Because of limited capacity, the scheduling of the annealing furnace is an especially challenging task. The furnace is one of the major bottlenecks in this factory, running at about 90% of capacity most of the time. Getting material through the furnace expeditiously ensures that products flow smoothly through the shop, expensive extrusion machines are not kept idle, and ultimately, the plant meets its projected demand figures in a timely manner.

A research team from Carnegie Mellon and the University of Pittsburgh visited this factory twice a week for 3 months, in order to gather information about the machines and processes, as well as knowledge about how the human scheduler performs the above-mentioned tasks. The goal of these visits was to lay the basis for an automated scheduling system, which would include the mathematical system as well as an expert system, to help the schedulers in their daily work. The team learned about manufacturing processes and scheduling expertise through interviews with factory personnel, as well as through a kind of apprenticeship for scheduling tasks. Interactions with the head scheduler—an expert with over 20 years of experience—included listening to his explications of what he did and why, and then attempting to apply this knowledge by solving his daily tasks ourselves. Our transcripts of how he explained his work, as well as of the critiquing sessions in which he let us know how our attempts measured up, provides the basis for the conceptual model representing the human's view of the scheduling problem in QEX, as well as for the types and phrasing of the questions and answers generated by the explanation system.

2.2. The Mathematical Scheduling System

The mathematical scheduling system that QEX explains is based on Morton's (1993) bottleneck dynamics. As implemented for this particular application, the bottleneck dynamics system chooses jobs (called "lots") for the furnace by taking, from the material available for annealing, those lots with the highest priority as determined by a benefit–cost ratio. This ratio balances the importance of the job with the cost of the resources that it needs to finish processing. Job importance is determined by a weighting factor, which in the real world may reflect customer importance or managerial priorities, times a slack factor, which reflects the relationship between the time that the lot is due and the time that it has left to process. Thus, the priority of the job is determined by:

$$\frac{(\text{weighting factor}) \times (\text{slack factor})}{\sum_{\text{machines left}} (\text{machine price} \times \text{process time})} \qquad (1)$$

The price of any machine is the sum of the delay costs that would be incurred by the jobs in process and on queue (presently and expected) for that machine, if it were to be idle for a given period of time. The formula for machine prices is:

$$\pi_k = \left(\sum_{j=1}^{L_k} W_j\right) + \overline{W}\frac{L_k \rho_k}{(1 - \rho_k)} \qquad (2)$$

where L_k is the current queue length on machine k; W_j is the current weighting factor for job j; $\overline{W}$ is the average of the weights of all jobs in the shop; and ρ_k is the long-term utilization factor on machine k. The first term of Equation 2 takes into account the jobs currently on queue for the machine, and the second term is an estimate of jobs that will be arriving in the near future (i.e., before the machine next becomes idle) using a known result from queuing theory. The utilization factor reflects how close to capacity a given machine usually runs; this is adjusted by the rest of the formula in Equation 2 to reflect the fact that the "bottleneck," or machine running closest to capacity, may shift from one process to another depending on the situation on the factory floor. A longer queue in front of a machine will raise its price, so depending on the trade-off between historical utilization and actual queue length, different machines may be bottlenecks at different times.

Intuitively, the benefit–cost ratio (Equation 1 above) allows "important" jobs (those from critical customers, or those very close to their due date) to be given higher priority for processing, as long as their processing does not cost the shop too much in delay costs of other jobs, as represented by the machine prices in the denominator. If all the lots have the same importance, then later-pass lots (i.e., lots that are further along in the manufacturing process) have higher benefit–cost ratios, since, all things being equal, the denominator will be smaller because of less processing time to go. If they are in the same pass, those lots with higher job importance would be scheduled first. Depending on the amount of resources to go (i.e., the pass number), more important material from an earlier pass may or may not be scheduled before later-pass material of a less important product.

3. THE QEX EXPLANATION SYSTEM

QEX produces justifications of the scheduling decisions made by the bottleneck dynamics system. Users can initiate a dialogue by asking two types of questions (chosen from a fixed set of questions on a menu screen). First is a "why" question, such as "Why was *lot-X* scheduled?" The second question is a "why not," querying the reasons that a particular job was not chosen for scheduling. After a response is generated, the user may ask follow-up questions to obtain more details about the specific attributes of the job in question that led to the decision to place it on the schedule (or not).

To produce explanations, QEX makes use of two problem solving models: that of the human scheduler (or other potential user of the automated scheduling system) and that of the bottleneck dynamics system. The scheduler views the problem in terms of satisfying the requirements of jobs (i.e., priorities, due dates, and customer demand) and of machines (balancing the line so as to avoid either starving machines or piling up too much material). The goals of the bottleneck dynamics system are similar, although achieved in a different manner: it schedules each job by balancing its importance (including its priority and relative closeness of its due date) with the costs the plant would incur by running the job now (the cost of resources, as measured by delay costs of other jobs waiting to run). Both approaches take into account job priorities, due dates, and the length of queues for various machines. The explanation facility exploits these correspondences to answer the queries in the user's own terms.

Since the bottleneck dynamics system makes decisions based on numerical values, questions about why a certain job was (or was not) scheduled (and follow-up questions asking for more details) must be answered by determining which numerical values are significant, and then translating this significance into terms familiar to the human scheduler.

3.1. Overview of QEX Explanation Process

An overview of the QEX system is shown in Figure 1, where data structures are depicted in ovals and pro-

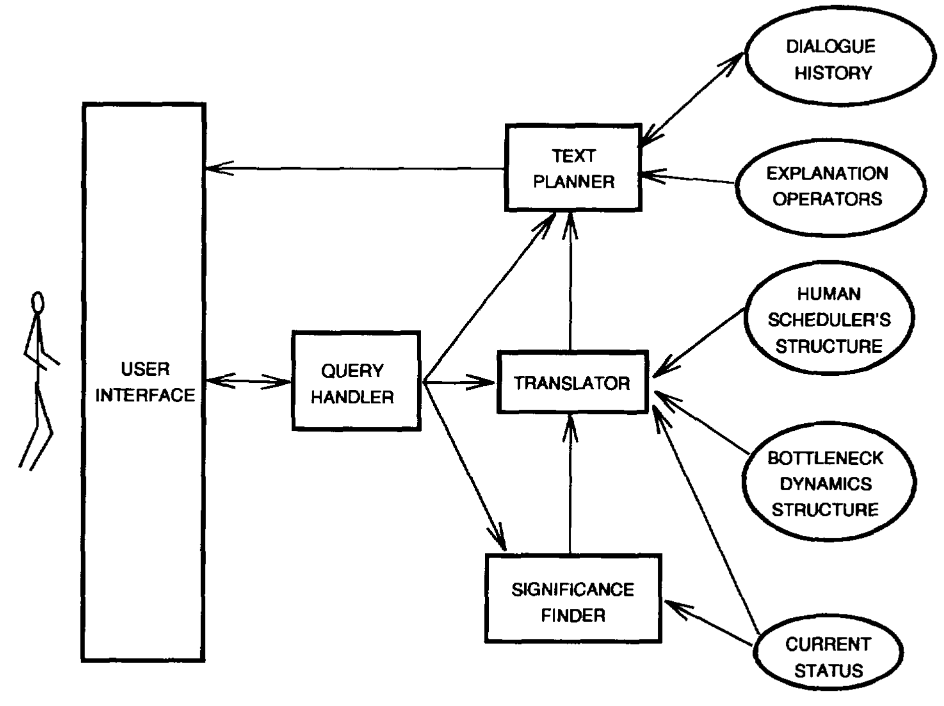

FIGURE 1. Diagram of QEX processing modules.

cessing modules are depicted in rectangles. As shown, QEX is made up of a text planner and associated modules that process user queries and generate explanations. It takes as input three static structures and one dynamic structure. Two of the static structures represent the two conceptual approaches to scheduling, that of the human scheduler and that of the bottleneck dynamics system. The third static structure contains a set of explanation operators, which comprise the system's repertoire of strategies for producing responses to user's queries. The fourth structure represents knowledge about the current status of the factory floor, and contains two kinds of related information. The first is static information about the "world" of scheduling in this domain (products and their attributes, etc.). The second is a dynamic structure consisting of the results of the FORTRAN program that implements the bottleneck dynamics model, and includes information about the current status of the factory floor (machine setups, status of lots, etc.), the current furnace queue and schedule, and information about each lot on the queue (including its status, size, due date, benefit–cost ratio, and information about the processes that it has completed so far and the resources on which it has yet to process). These structures are encoded as objects of the Common LISP Object System (CLOS).

Using these data structures, QEX generates explanations as follows. Before QEX begins, the bottleneck dynamics system has been given a factory floor configuration, has produced a schedule, and information about the results of this run have been translated into CLOS objects. The schedule is displayed on the screen by the user interface, and the user may then begin to ask questions about the schedule by choosing questions from the interaction menu. To ask a question such as "Why was lot N88-99 scheduled?," the user selects the option "Why was lot_scheduled?" and is then prompted to fill in the lot number. An internal form of the complete query is then passed to the query handler for further processing.

The query handler performs several functions. First, it checks to see whether the question is appropriate. That is, when the question is of the form "Why *fact*?," the query handler first checks to make sure that *fact* is true in its current model of the world state. For example, if the user asks "Why was lot NN-MM scheduled?," the system checks the current schedule to make sure that this lot is indeed on the furnace schedule. If it is not, the system points out the user's mistake and asks for further input, if desired. Similarly, when the question is of the form "Why not *fact*?," the query handler checks to see that the fact in question is indeed not true, and provides the appropriate feedback to the user if this is not the case.

If the user's question is appropriate, the query handler forms a communicative goal and passes this to the text planner and the significance finder. Communicative goals represent the speaker's intentions to affect the knowledge or goals of the hearer. They are denoted as states, such as the state in which the hearer knows why a particular lot has been scheduled. For example, if the user asked the question "Why is lot N88-99 scheduled?," the query handler would form the goal (KNOW-WHY HEARER (IN-SCHEDULE N88-99)).

Before the text planner begins the task of forming an explanation plan to achieve the communicative goal, several preprocessing steps must be done. First, the significance finder must determine why the bottleneck dynamics program has (has not) chosen the lot in question, which amounts to determining which factors of the benefit–cost ratio make this lot more (less) "important" or less (more) costly than an appropriate "peer group" of lots on the furnace queue. We discuss the significance finder in more detail below. The output of the significance finder is a list of nodes of the bottleneck dynamics structure that correspond to significant variables.

Next, the translator attempts to find the human scheduler's equivalent of the bottleneck dynamics concepts corresponding to each "significant" variable, by matching across the hierarchies. Because the two structures are not isomorphic, not every "significant" concept from the mathematical system will be mapped into the human scheduler's world. This is an inevitable result of the premise of QEX that the scheduler does not understand the theory or mechanics of the bottleneck dynamics system, and so not everything in one world can be explained in the other.[1] The output of this phase of processing is a list of nodes from the human scheduler's structure, which form a knowledge pool that is used by the text planner in constructing the explanation.[2]

Once "significant" variables have been determined and a relevant knowledge pool has been created, the text planner can begin synthesizing the text of the explanation. It does this via a simple decomposition planning mechanism based on the planner of Moore and Paris (1993). When a goal is posted, the planner searches its library of explanation operators looking for candidates capable of achieving the goal. In general, several candidate operators may be applicable, and the planner employs a set of selection heuristics to determine which strategy is most appropriate in the current situation. These selection heuristics take into account information about the user type and the conversation that has occurred so far (as recorded in the dialogue

[1] In the present implementation, significant variables that cannot be mapped into the scheduler's world are ignored; if they can be decomposed into various factors, and some or all of those factors are translatable, then they are communicated, at some point, to the user.

[2] We could also generate explanations for a user fully conversant with bottleneck dynamics, by simply skipping the translation step and handing the text planner the original bottleneck dynamics nodes. We plan to add different types of users in future versions of QEX.

history). The selection heuristics also consider the specificity of the explanation strategy employed by each operation, the number of constraints on the operator, and the number of actions in its decomposition. Once a strategy is selected, it may in turn post subgoals for the planner to refine. Planning continues in this fashion until all goals are refined into speech acts, such as ASSERT and RECOMMEND.

As the system plans explanations, it records the goal structure of the response being produced. The result is a text plan that explicitly captures the intentional and rhetorical elements of the text it produces. Text plans are recorded in the dialogue history, and then realized to produce English text. To produce the actual text, the system realizes each speech act, adding discourse connectives based on the rhetorical relations that connect subtrees in the text plan. In the current system, speech acts are mapped directly into templates. Examples of the operators used by the text planner are given in Section 7, and we work through a sample dialogue with QEX in Section 8. The text is presented by the user interface, and the user is again presented with the menu of questions that may be asked.

If the user asks another question about the same lot, QEX does not repeat its earlier explanation. Instead, by comparing the current communicative goal to the goals that appear in the dialogue history, it recognizes that the user is asking about a lot that has already been discussed. In this case, a new knowledge pool is formed by progressing down through the nodes of the human scheduler's structure. This corresponds to further decomposing the knowledge structures in the model, thus placing more detailed knowledge in the pool, and thus resulting in more detailed explanations. The user can continue asking about a particular lot, and the system will produce more detailed answers until it reaches the leaf nodes of the structure being traversed and has nothing more to say. Of course, there are other ways to answer follow-up questions besides providing more detail about what has already been said; a more sophisticated type of follow-up strategy is planned for a future version of QEX.

In addition, if the user asks about a lot that is similar (i.e., one with some of the same attributes) as a lot that has previously been discussed, QEX points out the similarity to the previous lot, and provides more information at the user's discretion. Again, QEX probes the dialogue history to determine which lots it has previously discussed and in what context (i.e., explaining why a lot was or was not scheduled).

In the sections that follow, we describe the knowledge structures and processing modules of QEX in more detail.

4. QEX KNOWLEDGE STRUCTURES

4.1. Human Scheduler's Structure

Figure 2 is a simplified picture of the hierarchical representation of the model of the human scheduler's approach to choosing lots for the furnace. It is based on the knowledge acquisition described above. Ideally, this model might be constructed from a planner or rule-based system that constructs the model from a set of operators/rules encoding the human scheduler's approach to solving the scheduling problem. However, for the purposes of this research, it has been hand coded.

As shown in Figure 2, the human scheduler thinks in terms of satisfying job and machine requirements by manipulating queues. The scheduler has three main objectives: making sure that jobs are finished on time (minimize lateness costs), trying to keep the factory floor balanced and prevent excessive idleness on machines (minimize processing costs), and keeping the work-in-process within acceptable limits (minimize inventory costs). Each of these objectives can be decomposed, in turn, into a number of subgoals, and so on, down to leaf nodes that represent the scheduler's concerns with individual attributes of lots and machines (such as due date, processing time, number of lots on queue, etc.).

The nodes are represented as CLOS objects. As shown in Figure 3, each node consists of a name, a goal (representing the scheduler's objective), information about its level in the model, pointers to the knowledge structures that contain information about the current shop status (processing times, due dates, queue lengths, etc.), and pointers to its parent, sibling and children nodes. The goal is a statement of the scheduler's objective, which is used by the translation module (discussed in more detail below). The associated variable is an optional slot, which points to a CLOS object containing a numerical value associated with this node. For instance, the objective of the node shown in Figure 3, "consider remaining time," is associated with the notion of "lead-time," that is, the time that this lot will take to finish its processing in the shop. As we will see in the examples below, this pointer enables QEX to include specific numbers in its explanations when they are part of the scheduler's conceptual world. Not all variables used by the bottleneck dynamics program will be associated with a node on the scheduler's model; for the most part, they are decomposed into variables that can be translated.

4.2. Bottleneck Dynamics Structure

Figure 4 is a simplified picture of the hierarchical representation of the bottleneck dynamics scheduling system, as implemented in the FORTRAN program. This structure was created using three sources: the benefit–cost ratio itself (and its decomposition into primitive values), its implementation in the FORTRAN program, and interviews with Thomas E. Morton and David W. Pentico, the developer of the theory and the specific implementation, respectively. The three highest-level

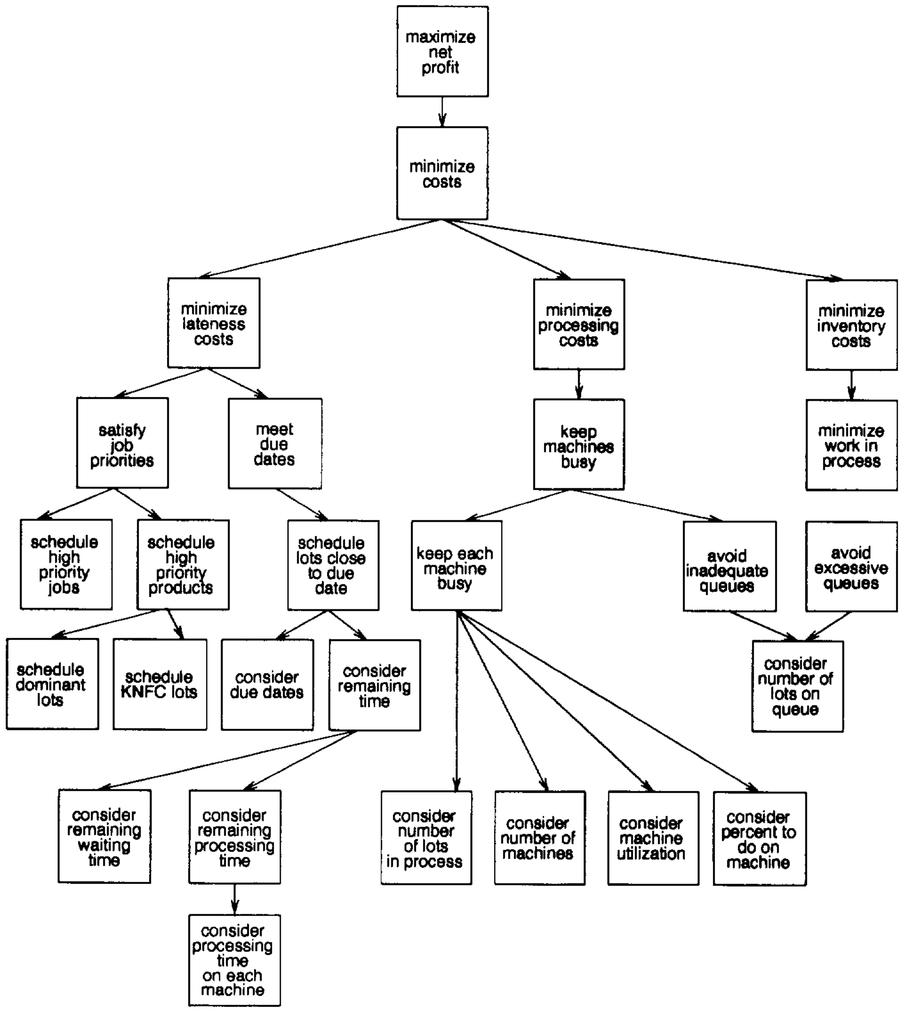
maximize net profit
minimize costs
minimize lateness costs
minimize processing costs
minimize inventory costs
satisfy job priorities
meet due dates
keep machines busy
minimize work in process
schedule high priority jobs
schedule high priority products
schedule lots close to due date
keep each machine busy
avoid inadequate queues
avoid excessive queues
schedule dominant lots
schedule KNFC lots
consider due dates
consider remaining time
consider number of lots on queue
consider remaining waiting time
consider remaining processing time
consider number of lots in process
consider number of machines
consider machine utilization
consider percent to do on machine
consider processing time on each machine

FIGURE 2. Hierarchical structure of the human expert's approach to scheduling.

nodes represent the overall objective of helping the firm to maximize its profit; the general objective of the scheduling system, which will minimize costs by scheduling efficiently; and the specific objective of this particular implementation (the FORTRAN program), which minimizes a combination of weighted flowtime and weighted tardiness.

The next node represents the benefit–cost ratio itself. Each subsequent level of nodes represents a stage in the decomposition of this ratio. It is first decomposed into the numerator (representing the benefit of scheduling a particular job at a particular point in time) and the denominator (representing the cost of scheduling this job at this particular time), which are in turn decomposed into their factors, which become the next generation of nodes. For instance, the numerator is the product of job weight (that is, the importance of the job) and a slack factor. This decomposition is necessary because the human scheduler does not share the concept of "benefit–cost ratio" (in the sense that it is being used by the bottleneck dynamics system), and so it must be broken down into components that this user can understand.

```
name:  hs-node-6.4
goal:  consider-remaining-time
level: 6
associated variable:  lot-lead-time
parent-node:  hs-node-5.3
sibling-nodes:  (hs-node-6.3)
children-nodes:  (hs-node-7.1 hs-node-7.2)
```

FIGURE 3. Sample node from the human scheduler's model.

Consider the node from the bottleneck dynamics model with the goal "consider remaining time" in Figure 5. As for the scheduler's node discussed above, the variable associated with this node is also "lot-lead-time". The goal of this node corresponds to the goal of the scheduler's node in Figure 3. Thus, when the associated variable "lot-lead-time" is found to be significant, bds-node-7.2 will be placed on a list, and eventually "translated" by matching it with hs-node-6.4, which has a corresponding goal. Note that, although the goals (and in this case, the associated vari-

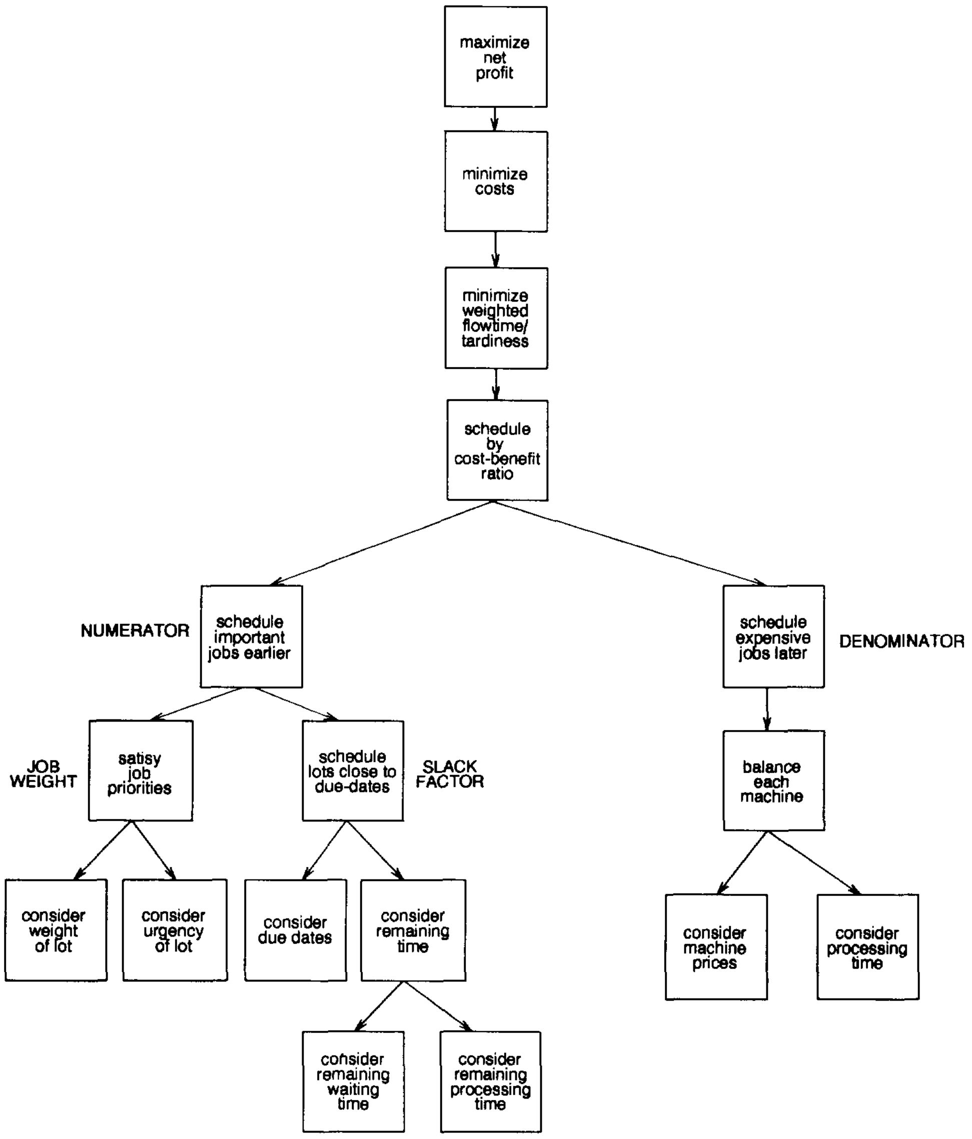

FIGURE 4. Hierarchical structure of the bottleneck dynamics approach to scheduling.

ables) correspond, these two nodes are in different places in their respective models; both are children of the parent node with the goal "schedule lots [that are] close [to their] due dates" and both have as their children nodes with the goals "consider remaining waiting time" and "consider remaining processing time." However, for the bottleneck dynamics system, these

```
name:  bds-node-7.2
goal:  consider-remaining-time
level:  7
associated variable:  lot-lead-time
parent-node:  bds-node-6.2
sibling-nodes:  (bds-node-7.1)
children-nodes:  (bds-node-8.1, bds-node-8.2)
```

FIGURE 5. Sample node from the bottleneck dynamics model.

children are leaf nodes. The human scheduler's structure continues to branch, with an additional child node with the goal "consider processing time on each machine" (this is actually a number of children, one for each machine, represented as one node in this graphical depiction). The leaf nodes that consider machine processing time for bottleneck dynamics, on the other hand, are in a completely different place in that structure. It is this lack of isomorphism that requires the creation of two separate models, and necessitates translation between them.

4.3. Factory Floor Domain Knowledge

This part of the knowledge base consists of CLOS objects representing lots and machine resources. These data structures are produced by the FORTRAN program that implements the bottleneck dynamics solution to the scheduling problem. Each lot on the furnace queue is set to an object that contains attributes like the following: its name (as known to the scheduler); pass; product; urgency (the numerical coefficient corresponding to the "importance" of the lot); due date; routing through the shop (the setup[3] numbers of resources and groups of resources); the benefit–cost ratio calculated at the time that the furnace schedule was assembled; its lead-time (the time it still has to process in the shop); and the price of each resource on which it still must process. QEX uses these attributes for three purposes: to determine significant variables associated with the lot, to ascertain whether it is similar (i.e., shares some attributes) with other lots that the scheduler may be interested in, and to identify it to the scheduler in terms of its product, pass, current size, and so on.

The representation of the factory floor also contains information that does not change, such as attributes of particular products. For instance, the attributes of the product "N88" include the size of its final diameter, its size in each pass, and whether or not it belongs to any specialized product line. This information may be included in an explanation in order to identify a particular "N88" lot, as well as to justify its place on the furnace schedule.

5. DETERMINATION OF SIGNIFICANT VARIABLES

In order to determine why a lot was scheduled by the bottleneck dynamics program, we need to isolate the factor or factors that made its benefit–cost ratio significantly higher than those of its peers. This might be one factor, for instance, a due date that is very close, or a combination of factors such as due date, remaining processing time, and estimated queue length on a future resource. It is important that we determine this significance in context, that is, in the context of the benefit–cost ratio itself, since it is precisely by such a combination of factors that the bottleneck dynamics system schedules jobs. QEX creates two lists of significant factors, which are actually lists of the nodes of the structure associated with each of these values.

In order to do this, we use an adaptation of the method developed by Kosy and Wise (1984) and used by Roth, Mattis, and Mesnard (1991). Our method of computing significance is as follows. Consider the set of values $V = v_1, v_2, \cdots v_n$, where each v_i is one of n factors in the decomposition of the benefit–cost ratio. In order to determine the significance of each individual factor, we measure the effect of the value of that factor for the lot in question in the context of the corresponding average value of its peer group.[4] So, if the set of average values (that is, the n factors representing the average values of the peer group) is represented by $A = \{a_1, a_2, \cdots a_n\}$ and the set of values for the lot in question is represented by $B = \{b_1, b_2, \cdots b_n\}$, we calculate the "effect" of factor b_j by computing a benefit–cost ratio R_j using the factors in the set $A' = \{a_1, a_2, \ldots, a_{j-1}, b_j, a_{j+1}, \cdots a_n\}$, and comparing it to the benefit–cost ratio R_a computed using just the average factors in A. That is, we substitute, one by one, the values of the "lot-factors" into the calculation, to see which characteristics of an individual lot make a "significant" difference. To determine which differences are significant in this sense, we compare the absolute value of the normalized difference $\Delta_j = (R_a - R_j)/(R_a - R_b)$ (where R_b is the benefit–cost ratio computed using the original set of values B described above) to a parameter which is determined empirically (for the example given below, we used 0.04). For each R_j for which the absolute value of the normalized difference is larger than the parameter, the corresponding value b_j is recorded as significant for inclusion in explanation.

Both Kosy and Wise (1984) and Roth, Mattis, and Mesnard (1991) go one step further, and try different ways of determining which combinations of factors are significant. For example, the Δ_i by itself might not be large enough to call factor b_i significant, but adding to it Δ_k might push it past the threshold, and so we could then say that b_i and b_k are significant in combination. Because we do not consider all of the possible factors b_j in our explanation (some might be untranslatable into the scheduler's world), it is not meaningful to apply this procedure in a straightforward way to our problem. Developing a way to account for the effects of combinations of factors in a system like QEX is an area for future research.

6. TRANSLATING BETWEEN TWO WORLDS

Once it has determined which variables are significant for explanation, QEX must find the corresponding concepts in the human scheduler's model. To do this, it processes the lists of significant nodes (that is, nodes that are associated with significant variables) that were produced by the significance finder, and tries to match the goal of each such node with the goal of a node on

[3] "Machine setup" is the physical status of the machine when it has been prepared to process a certain product; for example, a pilger machine would be set up by preparing it to produce tubes of specified inner and outer diameter.

[4] By "peer group" we mean here the set of lots that was *below* the lot in question in the furnace queue, if the scheduler is asking *why* the lot was on the schedule, and the set of lots that was *above* this lot on the queue, if the scheduler is asking, conversely, why a lot was *not* scheduled.

```
name:  justify-scheduling-regular-lot
effect:  (KNOW-WHY hearer (in-schedule ?lot))
constraints:  (AND (in-schedule ?lot)
                   (priority ?lot ?lot-priority)
                   (due date ?lot ?lot-due-date)
                   (KNFC ?lot nil)
                   (thimble ?lot nil)
                   (user type scheduler))
preconditions:  (AND (KNOW hearer (in-schedule ?lot))
                     (KNOW hearer (identification ?lot)))
decomposition:  (KNOW hearer (significant-factors ?lot))
```

FIGURE 6. Sample Operator 1.

the human scheduler's structure. There are two lists of significant bottleneck dynamics nodes, corresponding to variables that indicate significance for and against scheduling (for instance, a due date that is, on average, far away is a reason *against* scheduling; an anticipated queue of less than average length is a reason *for* scheduling). These are converted into two new lists, containing the corresponding human scheduler's nodes. This is done by taking the goal of each bottleneck dynamics node, and scanning the objects representing human scheduler's nodes for a corresponding goal. If no correspondence is found, then the concept represented by this node cannot be directly explained to the scheduler. However, chances are that it will be represented by some combination of children nodes, because most leaf nodes (which represent "primitives" such as processing time, queue length, etc.) represent concepts that are shared by the scheduler. The children nodes will then be the basis for the explanation of this concept that is produced for the human scheduler.

These two lists are used to generate an explanation in the following manner. Each of these nodes has an attribute indicating what level it is in the hierarchical structure. When the user first asks about a particular lot, the selection heuristics prefer an operator that accesses "primary nodes" (that is, the node or group of nodes that are highest in the hierarchy). These are used to generate the first explanatory text. Subsequent queries about the same lot motivate the choice of an operator that accesses "children nodes," and so on, until leaf nodes have been reached, and QEX has no more information on this particular subject.

7. THE TEXT PLANNER

QEX's text planner is based on prior work in text planning (e.g., Cawsey, 1993; Moore, 1994; Moore & Paris, 1993). Like Moore's (1994) system, QEX keeps track of the context of the interaction with the user, so that it can answer follow-up questions in a coherent and nonrepetitive manner. It does this by using a dialogue history, that is, by keeping track of what has already been asked and answered. The planner works by posting communicative goals (e.g., "achieve the state in which the user knows why a lot was scheduled"), and refining them until it reaches the level of speech acts (e.g., "tell the user the primary reasons for scheduling this lot"). A library of explanation operators used by the planner embodies the mapping between communicative goals and speech acts. When there are multiple operators that are capable of achieving a particular goal, they are ranked by a set of selection heuristics, and the highest ranking operator is chosen.

Figure 6 is an example of a plan operator that might be chosen when the user asks why a certain lot was scheduled. A goal is posted to "make the hearer know why a certain lot was scheduled," i.e., (KNOW-WHY hearer (in-schedule ?lot)). This matches the effect of this operator, so the text planner next tries to satisfy the constraints by checking the facts about the lot itself (its priority, due date, etc.) and the user type. Constraints specify when an operator is applicable, but they also specify the type of knowledge to be included in an explanation, that is, the process of satisfying constraints causes the planner to find information that will be included in the text. For example, to satisfy the constraint (priority ?lot ?lot-priority) in the above operator, the planner must find a binding for the variable ?lot-priority. If there is a set of variable bindings satisfying the constraints, then this operator is put on the candidate list. If it is chosen by the selection heuristics, the operator is added to the plan tree, and the subgoals in its precondition and decomposition fields are placed on the agenda of goal to be satisfied.

Figure 7 shows an operator that can be used to satisfy the goal (KNOW hearer (identification ?lot)), which is one of the preconditions of the operator in Figure 6. The operator in Figure 7 has no preconditions, and the decomposition is a speech act that will be used, in conjunction with a template, to generate the explanatory text.

In QEX, follow-up questions consist of queries for more detailed information about why a lot was (or was not) scheduled. The system keeps track of what has already been explained, by keeping a stack of pre-

```
name:  identify-lot
effect:  (KNOW hearer (identification ?lot))
constraints:  (AND (name ?lot ?lot-name)
                   (pass ?lot ?lot-pass)
                   (product ?lot ?lot-product)
                   (final-OD ?lot ?lot-final-OD)
                   (size ?lot ?lot-size)
                   (user type scheduler))
preconditions:  nil
decomposition:  (Assert speaker hearer (LOT-IDENT
                                        ?lot-name
                                        ?lot-pass
                                        ?lot-product
                                        ?lot-final-OD
                                        ?lot-size))
```

FIGURE 7. Sample Operator 2.

```
FURNACE SCHEDULE:
    S18-75
    S18-74
    N88-99
    S18-82
    S18-97
    S68-49
    S68-54
```

FIGURE 8. Sample furnace schedule.

viously refined communicative goals and checking it whenever the user asks a question. In general, QEX will not choose the same operator to achieve a communicative goal twice; in effect the list of candidate operators is progressively shortened each time the same question is asked. The exception to this is those operators that decompose "reasons" by looking at successively lower levels of nodes of the human scheduler's structure; these operators may be reused since the explanation will differ because the knowledge pool that is used to satisfy constraints will contain a different set of nodes as the system moves down in the structure it is explaining. In this way, QEX may follow up an explanation referring to "remaining time," for example, with one that refers to the components of that variable, "remaining waiting time" and "remaining processing time" (see Figure 2).

In order to make use of context in the case of similarities among lots that are the subject of queries, QEX uses an adaptation of the approach of Moore (1993). It does this by keeping track of which individual lots have been asked about, and comparing certain attributes (product name, stage of processing, and due date) of each of those with the attributes of the lot that is the current focus of explanation. "Similar" lots are those of the same product that are in the same stage of processing; "equivalent" lots are similar lots that have the same due date. Operators that refer to a prior explanation of a similar or equivalent lot are preferred by the selection heuristics. The sample interaction with QEX shown in the Section 8 will illustrate this functionality.

8. SAMPLE EXPLANATION

The following example is taken from an actual run of the bottleneck dynamics system, and the explanations generated by queries to QEX. The scheduling program has been run to suggest to the scheduler what should be loaded into the next furnace run. The scheduler is presented with the list of recommended lots (Figure 8), as well as a list of all the lots available on the furnace queue (Figure 9). Now the interactive session begins. In the figures showing sample dialogues, the system's utterances are depicted in typewriter font, while the user's utterances appear in *italics*.

As shown in Figure 10, QEX begins the session by asking the user to specify a user type. This is because the system is potentially capable of tailoring answers to different user types (e.g., scheduler, theorist, manager). Tailoring is accomplished by including the user type as a constraint on certain operators used by the planner. One of the heuristics that chooses among possible alternative operators gives preference to operators that cater to the current user type. This will also prevent the system from generating inappropriate explanations for a specific user (for instance, schedulers will not be confronted with an explanation that refers to machine prices, which are not in their conceptual world).

Next, the user is prompted for a query by presentation of a menu of questions that can be asked at this point. The scheduler first asks why one of the lots, N88-99, was placed on the furnace schedule.

Turn 5 of the sample dialogue in Figure 10 provides an example of the strategy QEX uses to explain why a lot was scheduled. It first identifies the lot by its final outer diameter (0.357), its pass (second) and its current size (1.00 inch). This is how the scheduler thinks about, and refers to, lots on the factory floor. QEX finds this information about lot N88-99 in its data structure representing the status of the factory floor. The numerical values are slots in the CLOS object that is instantiated with the attributes about N88-99, which are in turn bound to variables that the text planner uses in generating explanations.

The next two sentences present the "highest level" reasons (in terms of the conceptual hierarchy depicted in Figure 2) for and against scheduling this particular

```
FURNACE QUEUE:
    S18-75
    S18-74
    N88-99
    S18-82
    S18-97
    S68-49
    S68-54
    S18-95
    S18-96
    S68-50
    N88-15
    K21-24
    K21-25
    W73-16
    W73-17
    W73-18
```

FIGURE 9. Sample furnace queue.

```
Welcome to QEX, The Quantitative EXplainer.                     [1]
Please enter your user type.

scheduler                                                       [2]

What would you like to ask?                                     [3]

Why was lot N88-99 scheduled?                                   [4]

Lot N88-99 is a 0.357 second-pass lot of size 1.3.  In          [5]
comparison with lots on the furnace queue which were not
scheduled, this lot was chosen for the following reasons.
Although, in comparison with other lots, its due date is not
really that close (137 hours), it was scheduled because there is
a chance that it won't finish processing in time to meet its due
date.

Tell me more about lot N88-99.                                  [6]

You wanted to know more about why this lot was scheduled.  It   [7]
was placed on the schedule because, in comparison with other
lots, it will take a relatively long time to get through the
shop (104 hours).

Tell me more about lot N88-99.                                  [8]

You wanted to know more about why this lot was scheduled.  It   [9]
was placed on the schedule because, in comparison with other
lots, its total processing time on remaining machines is pretty
long (82 hours), and also because there are a large number of
third-pass pilgers set up for this product (3 pilgers).

Tell me more about lot N88-99.                                  [10]

You wanted to know more about why this lot was scheduled.       [11]
Sorry, I've already told you everything I know about this
subject.
```

FIGURE 10. Sample dialogue with QEX.

lot, as derived from an analysis of the output of the FORTRAN program, as described above.

The reason *against* scheduling, the fact that the due date of this lot is *not* that close, relatively speaking, comes from the fact that the variable ``lot due date'' turns out to be significant. That is, when it was compared to the due dates of the lots *below* it on the furnace queue, its effect on the benefit–cost ratio was significant, in the direction of not scheduling. In other words, the due date of this lot was relatively farther away, as measured by the algorithm described above. The bottleneck dynamics node with the goal ``consider due dates'' was placed on the list of significant nodes, and subsequently ``translated'' into the human scheduler's node with the same goal, since the human scheduler also thinks in terms of due dates when planning a furnace load. The due date itself (expressed in hours from the time of the furnace load) is also given; this comes from the associated variable of the human scheduler's node ``consider due dates.''

The reason for scheduling this lot is ``because there is a chance that it won't finish processing in time to meet its due date.'' This was arrived at in the following manner. The variable ``lot_slack'' was found to be significant. This corresponds to the bottleneck dynamics node with the goal ``schedule lots close to due dates,'' which is matched in turn with the human scheduler's node with the same goal. In this case, although the goals match, enabling us to translate the concept from one world to another, the number itself (which embodies a mathematical relationship between due date and expected time to completion) has no meaning to the human scheduler, so it is not presented in the explanation.

Why do we give reasons *counter* to the decision reached by the mathematical program (here, that means reasons *against* scheduling this lot)? Because if those reasons are important enough, the scheduler will certainly be aware of them, and in fact will probably have such reasons in mind when posing a question to QEX. For instance, the question in turn 4 of the sample dialogue might have been asked because the scheduler looked at the suggested furnace schedule and thought, ``But N88-99 isn't due for a while. Why schedule it now?'' It is, therefore, a good rhetorical strategy on the part of the explanation system to anticipate such objections by admitting up front that there are factors that *apparently* argue against the decision of the bottleneck dynamics system. However, QEX treats counter-arguments differently from reasons that support the program's decision, in that it prunes away all but the highest level ones. That is, when scheduler's nodes are found that correspond to reasons *against* the action taken by the scheduling program, only the top few levels of these counter-arguments will be used in generating explanations. In contrast, when presenting reasons that support a decision, all nodes are used, down to the lowest level generated. The idea is that it is only worthwhile to acknowledge the most general objections.

The explanation in turn 5 communicates the highest level reasons for and against scheduling this lot, corresponding to the highest level *scheduler's* node that was matched from ''significant'' bottleneck dynamics nodes. In this case, ''It was scheduled because there is a chance that it won't finish processing in time to meet its due date'' is the text template corresponding to the scheduler's node with the goal ''schedule lots close to due dates.'' Thus the bottleneck dynamics concept of ''slack'' has been translated into the user's language.

As we have already mentioned, the clause beginning with ''although,'' represents a reason against scheduling this lot. It turns out that its due date is ''significantly'' further away than that of its peers. However, in this case the favorable effect of another significant factor (slack factor) overrides the negative effect of due date, and so the lot was scheduled anyway. The system includes in the explanation a relevant numerical value, such as due date, when that value itself is a viable concept in the user's world (slack factor, in contrast, is not, so no number is given).

Next, the scheduler wants to know more details about why this lot was scheduled. The scheduler asks ''Tell me more about lot N88-99'' (see turn 6 in Figure 10). In effect, the same question is asked again (the same goal is posted), but the resulting explanation is different. The response of QEX is shown in turn 7 in Figure 10.

''It will take a relatively long time to get through the shop'' is the text template corresponding to the numerical value representing lead time (the anticipated amount of time that the lot will take to complete its processing). That is, this lot has a lead time that is significantly larger than that of its peers, and so this is another reason that its benefit–cost ratio is relatively larger. Lead time is a component of slack factor (as expressed by their hierarchical relationship in Figure 2 where the node with the goal ''consider remaining time'' is a child of the node with the goal ''schedule lots close to due dates''), and so we see how QEX gives additional information by decomposing variables into their component parts.

In answer to the next follow-up question, QEX gives additional information (see turn 9 in Figure 10).[5] This time QEX goes ''down'' once more, decomposing lead time into its components of waiting time and processing time, and finding that the latter is significant (the corresponding scheduler's nodes are those labeled ''consider remaining processing time'' and ''consider remaining waiting time'' in Figure 2). The second part of the explanation represents the scheduler's motivation to keep machines busy (especially the pilgers, or extrusion machines, which are relatively expensive to run and are usually quite busy in the last stages of processing). Now we are in the branch of the human scheduler's structure that descends from the node labeled ''minimize processing costs'' in Figure 2. ''A large number of third-pass pilgers set up for this product'' (corresponding to a child of the node labeled ''consider number of machines'' in the same figure) indicates to the scheduler that there had better be a steady stream of N88 lots to keep these machines busy. The bottleneck dynamics system also considers the number of machines set up, as part of the equation that computes machine prices; since machines are aggregated into groups of resources, the price for each group is divided by the number of machines in that group. A large number of machines means a lower price, that is, a lower denominator, which in turn means that the overall benefit–cost ratio of a given lot may be higher as a result. Telling the scheduler about ratios, denominators, and machine prices, however, would not be convincing, so QEX finds a corresponding concept and uses it for explanation. Turns 6–9 of Figure 10 show how QEX decomposes a concept that cannot be translated, finding the significant variables among its factors, and using them to generate explanation.

The scheduler may continue to ask for more detail, and as long as there are nodes on the list of significant nodes, the system will produce more and more detail. Eventually, of course, it will run out of things to say, as shown in turn 11 in Figure 10.

The sample dialogue continues in Figure 11, with the scheduler asking about a different lot on the schedule, S18-74. The explanation (turn 13 in Figure 11) is prompted again by the significance of the variable ''lot slack.''

The next query shows how QEX recognizes ''similar'' entities, and tailors its explanations to that context. In turn 14 of Figure 11, the user asks about lot S18-96, which is not in the schedule. The answer in turn 15 pinpoints the differences between the two lots, and it is in this context that QEX explains why S18-96 was not scheduled. They are both the same product and pass (which is the definition of ''equivalent'' lot in this domain). The variable representing slack is also significant for this lot. But the difference between these two lots is that while slack is a reason *for* scheduling the first, it is a reason *against* scheduling the second. QEX points this out explicitly saying ''Unlike lot S18-74 . . . .'' Of course this is related to the difference in their due dates, which QEX points out.

If two lots are exactly equivalent (including their due date), and one is scheduled while the other is not, the user will receive an answer like the one shown in turn 17 of Figure 11. In order to give this answer, QEX must know: (1) that the two lots are similar; (2) that one has already been asked about; and (3) that one has been scheduled and the other hasn't. This is accomplished by the use of a dialogue history, as described

[5] In the present implementation, every response to a request for more information is introduced in the same way; this will be further refined to provide more textual variety in the next version.

Why was lot S18-74 scheduled? [12]

```
Lot S18-74 is a 0.368 third-pass lot of size 0.57. It is a KNFC      [13]
lot which, in comparison with other lots on the furnace queue
which were not scheduled, was chosen for the following reasons.
There is a chance that it won't finish processing in time to
meet its due date.
```

Why was lot S18-96 not scheduled? [14]

```
Lot S18-96 is a 0.368 third-pass lot of size 0.57. It is a KNFC      [15]
lot which, in comparison with other lots on the furnace queue
which were scheduled, was not chosen for the following reasons.
Unlike lot S18-74, it will probably be able to finish its
processing in time to meet its due date.
```

Why was lot S18-95 not scheduled? [16]

```
This lot is equivalent to a lot that you have already asked          [17]
about, S18-74. You may schedule this lot instead of S18-74 if
you wish.
```

Why was lot S68-49 not scheduled? [18]

```
But that lot is on the current schedule!                             [19]
```

Why was lot W73-21 scheduled? [20]

```
But that lot is not on the current schedule!                         [21]
```

FIGURE 11. Sample dialogue with QEX continued.

above, as well as by using data structures that enable QEX to look up various facts about the factory floor.

The answers in turns 19 and 21 of Figure 11 show how QEX does not allow the user to "fool" it by asking for an explanation of something that is not true.

9. CONCLUSIONS AND FUTURE DIRECTIONS

QEX attempts to demonstrate that automated explanation can bridge the gap between quantitative systems and uninitiated users. It builds upon previous work in explanation planning and quantitative explanation, and adds the additional capability of translating between two conceptually varying worlds. Within certain constraints (with regard to the similarities and differences between those worlds), we believe that this approach to explanation can be applied generally to quantitative advice-giving systems that are meant to be used by those who are not well-versed in the formal problem solving models on which those systems are based. We plan to apply our explanation methodology to other such quantitative systems.

The development of the QEX prototype has raised many issues and avenues of future research. We plan to enhance the system itself to handle a broader range of follow-up questions. The interface will be extended to permit the user to "point" with the mouse (Moore & Swartout, 1990), highlighting words or phrases that can be further elucidated. Since the bottleneck dynamics program has been explicitly represented, another enhancement would be to enable the user to ask questions about the workings of that program (which would assume a user who is more interested in the formal model than the "uninitiated" one that we have posited so far).

Additional user types, which would require different representations and explanation strategies, will also be added. For example, a manager or floor supervisor might require a different set of concepts for translation; a user who was well-versed in the theory of bottleneck dynamics would not require that its concepts be translated at all. This enhancement will also help us to evaluate the explanation system.

The method of testing for significant variables might also be refined. We have already mentioned the fact that we do not now consider combinations of factors, and plan to investigate ways to incorporate that into our significance finding strategies. Another area of future inquiry is to consider different ways of comparing focus variables and their "peer groups." We now use simple averaging; for example, we compare the due date of the lot being asked about to the average of the due dates of the lots above or below it on the furnace queue. We plan to compare our current method with different statistical methods that consider variance in the context of determining significance.

Another extension to the present system would be to provide the capability of dynamically generating each user's problem solving approach, by reconstructing backwards from the solution provided by the mathematical system, as does REX (Wick & Thompson, 1992). This would make the explanation system more robust, because it would have a working "understanding" of how to solve the problem at hand, and would allow the scheduler's model to discover additional evidence for the bottleneck dynamics system's results. In addition, such a system might be able to resolve conflicts between differing solutions from two different types of problem solvers. These conflicting solutions might come from a mathematically-based system and one developed from studying human expertise, or even from two different mathematical systems (for example, one based on a physical model and one based on statistics).

In terms of evaluating the present implementation of QEX, a project is now under way to test the robustness of the program over different types of scheduling situations, as well as different settings of the bottleneck dynamics program. We are using actual factory data from several different historical days, and perturbing it to perform sensitivity analysis on factors such as different levels of shop loading (how busy it is), various relationships between processing times on machines (to examine the relative strength of the bottleneck at the furnace), and different spreads among the due dates of the lots to be processed. We will also change the relationship between the two parts of the objective function (flowtime and tardiness). Generating explanations from the output produced by these perturbations will show how QEX reflects differing realities on the factory floor and different parameterizations of the bottleneck dynamics program.

Acknowledgments—Susan Slotnick developed QEX as part of her thesis work while a Ph.D. candidate at the Graduate School of Industrial Administration at Carnegie Mellon University. Johanna Moore gratefully acknowledges the support of the Office of Naval Research Cognitive and Neural Sciences Division (Grant Number: N00014-91-J-1694), and the National Science Foundation Research Initiation Award (Grant Number: IRI-9113041) while co-supervising this work and writing this paper. We would like to thank Tom Morton for his time, effort and patient explanations of bottleneck dynamics; Dave Pentico for his considerable efforts in writing the FORTRAN program implementing bottleneck dynamics; Prasad Ramnath for sharing his insights about statistical methods of comparison; and Steve Roth and Joe Mattis for their very helpful discussions and guidance on the algorithm for computing significance. Any omissions, inaccuracies or other mistakes are entirely our own.

REFERENCES

Ackley, D. H., & Berliner, H. J. (1983). *The QBKG System: Knowledge Representation for Producing and Explaining Judgments* (Technical Report CMU-CS-83-116). Pittsburgh, PA: Carnegie Mellon University.

Ben-Bassat, M., Carlson, V. K., & Puri, V. K. (1980). Pattern-based interactive diagnosis of multiple disorders: The MEDAS system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **2**, 148–160.

Buchanan, B. G., & Shortliffe, E. H. (1985). Explanation as a topic of AI research. In B. G. Buchanan & E. H. Shortliffe (Eds.), *Rule-based expert systems* (pp. 331–337). Reading, MA: Addison-Wesley.

Cawsey, A. (1993). *Explanation and Interaction: The Computer Generation of Explanatory Dialogues.* Cambridge, MA: MIT Press.

Cohen, R., Jones, M., Sanmugasunderam, A., Spencer, B., & Dent, L. (1989). Providing responses specific to a user's goals and background. *International Journal of Expert Systems*, **2**(2), 135–162.

Cooper, G. F. (1984). *A Computer-Based Medical Diagnostic Aid that Integrates Causal and Probabilistic Knowledge* (Technical Report STAN-CS-84-1031) Stanford University, Ph.D. Dissertation, Stanford, CA.

Kosy, D. W., & Wise, B. P. (1984). Self-explanatory financial planning models. In *Proceedings of the Fourth National Conference on Artificial Intelligence* (pp. 176–181). Menlo Park, CA: AAAI.

Langlotz, C. P. (1989). *A Decision-Theoretic Approach to Heuristic Planning.* Ph.D. Dissertation, Stanford University, Stanford, CA.

Langlotz, C. P., Fagan, L. M., Tu, S. W., Sikic, B. I., & Shortliffe, E. H. (1987). A therapy planning architecture that combines decision theory and artificial intelligence techniques. *Computers and Biomedical Research,* **20**(3), 279–303.

McCoy, K. F. (1989). Generating context-specific responses to object-related misconceptions. *Artificial Intelligence,* **41,** 157–195.

McKeown, K. R. (1988). Generating goal-oriented explanations. *International Journal of Expert Systems,* **1**(4), 377–395.

Moore, J. D. (1993). Indexing and exploiting a discourse history to generate context-sensitive explanations. In *Proceedings of the ARPA Human Language Technology Workshop* (pp. 165–170). San Mateo, CA: Morgan Kaufmann Publishers.

Moore, J. D. (1994). *Participating in Explanatory Dialogues: Interpreting and Responding to Questions in Context.* Cambridge, MA: MIT Press.

Moore, J. D., & Paris, C. L. (1993). Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics,* **19**(4), 651–695.

Moore, J. D., & Swartout, W. R. (1990). Pointing: A way toward explanation dialogue. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 457–464). Menlo Park, CA: AAAI.

Morton, T. E. (1993). *Heuristic Scheduling Systems. With Applications to Production Engineering and Project Management.* New York: John Wiley and Sons.

Roth, S. F., Mattis, J., & Mesnard, X. (1991). Graphics and natural language as components of automatic explanation. In J. Sullivan & S. Tyler (Eds.), *Intelligent user interfaces* (pp. 207–237). Reading, MA: Addison-Wesley.

Souther, A., Acker, L., Lester, J., & Porter, B. (1991). Generating coherent explanations to answer students' questions. In H. Burns, J. W. Parlett, & R. C. Luckhardt (Eds.), *Intelligent tutoring systems: Evolutions in design.* Hillsdale, NJ: LEA.

Spiegelhalter, D. J., & Knill-Jones, R. P. (1984). Statistical and knowledge-based approaches to clinical decision-support systems, with an application in gastroenterology. *Journal of the Royal Statistical Society,* **147,** 35–77.

Swartout, W. R., & Moore, J. D. (1993). Explanation in second generation expert systems. In J.-M. David, J.-P. Krivine, & R. Simmons (Eds.), *Second generation expert systems.* Berlin: Springer-Verlag.

Wick, M. R., & Thompson, W. B. (1992). Reconstructive expert system explanation. *Artificial Intelligence,* **54,** 33–70.