EXPLAINING AND ARGUING WITH EXAMPLES [1]

Edwina L. Rissland
Eduardo M. Valcarce
Kevin D. Ashley

Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Abstract

In this paper, we discuss two tasks – on-line help and legal argument – that involve use of examples. In the case of on-line HELP, we discuss how to make it more intelligent by embedding custom-tailored examples in the explanations it gives its user. In the case of legal argumentation, we discuss how hypotheticals serve a central role in analyzing the strengths and weakness of a case and describe the generation of hypotheticals, stronger or weaker for one of the parties with respect to a doctrinal aspect, through modification of already existing cases or hypotheticals.

1. Introduction

Explaining and arguing are two tasks which both often involve example-based reasoning. In explaining, one tries to elucidate certain knowledge, educe and correct misconceptions, answer questions and otherwise satisfy the question asker. Argumentation involves all that and more, but in a much more adversarial context; the emphasis is on convincing another that one's position is correct or showing that the other's is not. Admittedly there are major differences in explaining and arguing – for instance, the goals of the explainer and arguer – but nonetheless, there are striking commonalities. In particular, both rely heavily on the use of "for instances" to accomplish their tasks. In this paper, we shall focus on this shared theme.

Examples are critical to learning and to the structure of knowledge and memory [Dietterich & Michalski, 1983; Kolodner, 1983; Rissland, 1978]. Recently, Schank has suggested that explaining is perhaps even more critical than "reminding" in the structure of dynamic memory [Schank, 1984]. Examples play a central role in explaining since it is with examples that one finds the limits of generalizations and explanations. Anomalies and counter-examples, in particular, help bound concepts and rules.

In legal argument, one is constantly trying to test the limits of "rule-like" propositions and show why certain precedents should or should not control the decision of another case. In the law, it is cases, both "real" and hypothetical (i.e., cases which have not actually been litigated), which serve as examples. Hypotheticals serve many roles; they create, remake, re-focus, and organize experience and are used to explore concepts and rules and to tease out hidden assumptions [Rissland, 1984].

These observations apply to other domains as well, like mathematics and computer programming. In mathematics, where concepts and truth are more clearly defined than in the law, one is constantly engaged in the "dialectic of proofs and refutations" [Lakatos, 1976].[2] In programming, there is an "inevitable intertwining" [Swartout & Balzer, 1982] of the examples with the evolution of programs and specifications. This intertwining of examples and experience with proposing, refining and refuting can be seen "almost everywhere"; it is inherent to the basic life cycle of science [Kuhn, 1970].

In this paper, we will discuss the use of examples in two types of explanation and argumentation: the first is the case of on-line HELP and the second is legal argument with hypotheticals.

2. Explanation: On-Line HELP

By on-line *help*, we mean command assistance and assistance about certain concepts and standard tasks and, although we have not included it in our work, error and prompting assistance as well.

One important component of knowledge that is missing in most on-line (and off-line) explanation, especially help and manuals, concerns examples. Examples offer a concrete illustration of what is being explained and a memorable hook into more general information. They are especially important for the beginner.

[1] This work supported in part by Grant IST-8212238 of the National Science Foundation.

[2] Certainly in mathematical research. One could argue that learning mathematics should also follow along this line. Regardless, examples are an important component of mathematical knowledge and are vital to teaching, learning and understanding [Rissland, 1978].

Examples can provide easily understood and remembered usages. For instance,

PRINT VITA.MEM

is clearly more perspicuous than

"**PRINT** [[*d:*][*filename*[*.ext*]][/T][/C][/P]...]" (from [IBM 1983])

A novice uses simple cases: to figure out how to instantiate the general syntactic description, to use as "recipes" for standard tasks, as a basis for generalizing, and as a basis for a "retrieval+modification" [Rissland, 1981] approach to generating other examples. For the expert, examples can serve as a reminder of syntax and things previously done, much like an icon.

In most current help facilities, like that of VAX/VMS, the user asks for information about a particular command, like "HELP PRINT", and is then presented with information on PRINT, including relevant parameter options, but almost never including examples of standard, potentially dangerous, or clever uses. The explanations often include system jargon, like "queue", "world" or "file-spec", which the user ought to be able to ask about but usually can't.

Research on intelligent on-line explanation is still not very far advanced (see [Houghton, 1984]) although some interesting starts have been made. For instance, Wilensky, in his UC system, allows the user to ask for assistance in natural language; his work has concentrated on request understanding [Wilensky, 1982a, 1982b]. Finin, in his WIZARD system, has focussed on the problem of recognizing when the user needs help, particularly, because he is using inefficient means to accomplish a task (e.g., using repeated DELETEs instead of PURGE) and then volunteering advice [Shrager & Finin, 1982; Finin, 1983].

Our approach to improving on-line HELP is two-fold: (1) include more ingredients of expert knowledge like examples of various categories, heuristics, and pragmatic knowledge in the information provided to the user; (2) embed user-tailored examples in the explanations. In the rest of this discussion, we shall assume that the on-line help facility has already been invoked (by the user or the system) and that the facility has already 'parsed' the user's request (i.e., knows with what the user requires assistance). Our emphasis is on the generation of the response, in particular, from a model of the user's expertise, contextual information, and exemplar knowledge. We don't use any text generation, although using a language generation program like McDonald's MUMBLE [McDonald, 1982] is an obvious thing to do. Clearly, this work should eventually be tied in with work on invocation and parsing like that of Wilensky or Finin.

2.1 (More) Intelligent On-line HELP

In our HELP facility, we too offer explanations on commands like PRINT but we embed relevant, user-tailored examples in the explanation and allow the user to ask about system jargon as well as certain tasks. Further, we "ripple" contextual information in our explanations as well as use some (very) crude user modelling. For instance, if the user has just asked about PRINT and then asks what a "queue" is, the system would give as examples of queues those used for print jobs. We also allow the user to ask for assistance in a more task-oriented way, through keywords and phrases[3] like "clean-up" and "unlock". (The first leads into assistance about PURGE and saving files; the second, about setting the protection – which in most help systems is only accessible by explicitly asking about a seemingly unrelated command like SET – and possibly related system jargon like "world".) Task-oriented requests usually feed into explanations about specific commands, which are then focussed by the context defined by this type of access.

2.2 Embedding Customized Examples

We use taxonomic knowledge of examples[4] to help select and order the presentation of examples. For instance, we provide the absolute neophyte user with "start-up" examples and the more experienced user with "references". Where a sequence of examples is called for, the taxonomic knowledge can be used to order the examples, for instance, with references presented before models which are presented before counter-examples and anomalies. Taxonomic knowledge can also enable the explanation facility to allow the user to ask specifically for examples in a certain class (e.g., "easy", "anomalous", "clever").

One way to customize examples is to modify them to reflect specifics of the user and his context, for instance, using information about the user's own directory in explanations about directory commands. We use techniques of *retrieval+modification* and *instantiation* in our on-line HELP by linking the explanation program with an **example generator**, which uses an **Examples-Knowledge-Base (EKB)** of already existing examples together with procedures for modification and instantiation. The EKB consists of examples, represented as frames, harvested and organized by an expert. Procedurally attached to examples are instantiation and modification procedures like those to generate extreme variations or to personalize an example.

The ability to generate examples on the fly allows the explanation facility to respond dynamically to the user, his tasks, goals, context, domain, etc. User-specific "constraints" on the examples are provided by user-modelling capabilities, for instance, keeping track of how many examples have already been presented in the current invocation of HELP, and roughly gauging the user's expertise on the basis of certain system data like the block size of the directory, the number of sub-directories, the types of files owned, etc.

---

[3] Already known to the system and available for perusal by the user, this list easily can be augmented on the basis of users' needs.

[4] [Rissland, 1978] described a taxonomy of examples: "start-up" (easy, perspicuous cases); "reference" (standard, textbook cases); "model" (paradigmatic, template-like cases); "counter-examples" (limiting, bad cases); "anomalous" (ill-understood, strange cases).

The point is to work examples into the explanation given the user, and better still to make the examples meaningful in the sense of addressing the particular needs and knowledge of the user.

2.3 TEXPLATES: Separating Control from Content

In our HELP system, we separate the control of the help session from its content. We use a script-like control structure of text and examples organized in a template, a "TEXPLATE", which is then used to generate HELP's response. The explanations are assembled by retrieving the needed text and examples which are pointed to in the relevant texplate (texplates are indexed by commands, jargon and task keywords).

A TEXPLATE is a set of related nodes, each of which points to chunks of text or examples, and contains the selection criteria for that node. To maintain consistency with other system documentation, wherever possible we use text used elsewhere, for instance, in the manual. Calls to examples are either requests for explicitly named examples in the EKB or constraints describing modifications to be made to an example. For instance, example calls could be for: a named, known extreme case (e.g., DEL *.*.*); an example generated to fulfill prescribed constraints (like using the name of the user's most recently edited file); using a previously used example perhaps with an embellishment to make it extreme, anomalous, or clever.

A **Texplate-Interpreter** controls the flow through the texplates, including which user-options to present and what to do in response (e.g., MORE to go on with the explanation, EXAMPLE for an example, QUIT, etc.) as well as directing the generation of sequences of examples, should the user repeatedly ask to see an example. Each TEXPLATE defines a context within which the possible meanings of a particular request are narrowed. For instance, asking about the PRINT command will constrain certain specifications in subsequently selected examples such as those for related jargon (e.g., "queue", "job"), commands (e.g., SHOW/QUEUE), and tasks like those that *undo* the command just explained, (e.g., deleting a job from a print queue). Thus, if the user then asked about the meaning of "queue", the examples would be tuned to reflect the fact that the system just explained PRINT. Certain control options are always available for a texplate: these include asking for repeated examples and asking about commands that *verify* or *undo* the command just explained.

2.4 Two Examples: IA-LADYBUG & VMS

In our on-going work, we are working within two systems. The first is IA_LADYBUG, a system designed specifically for novice programming students with the aim of introducing them to notions useful in the Pascal programming language (like subprocedures) by having them work with a graphics icon, the LADYBUG, which can be commanded by LOGO-like commands such as CRAWL, RIGHT-TURN, etc. [Levine & Woolf, 1984]. The second is a subset of VAX/VMS command language [DEC, 1978] dealing with directory and protection commands like PURGE, DELETE, and SET PROTECTION.

Figure 1. Example scenario of user interactions with HELP

$ help

Enter a subject or command you need help with, or type a ? for a list of choices. Enter Q to quit.

HELP >**print**

PRINT
Queues one or more files for printing, either on a default system printer or on a specified device.
For example:
print login.com
will cause your file login.com to be printed on the default system printer.

HELP (Type M for More, E for Example, Q for Quit, or enter another request)> **queue**

A print queue is a list of files to be printed on a specific device.
For example:
print login.com /queue=narrow
requests that file login.com be printed on the printer corresponding to the queue named "narrow".

The /QUEUE qualifier is synonymous with the /DEVICE qualifier. For instance,
print login.com /device=narrow
will do the same thing.

HELP (Type M for More, E for example, Q to quit, or enter another request)>**E**

Another example:
print login.com/queue=sys$laser
will cause the file login.com to be printed on the laser printer

HELP (M, E, Q,...)> **m**

PRINT expects the file type to be "LIS" unless otherwise specified.
For example:
print login
will not print your file login.com and it will cause an error because you do not have a file named login.lis.

HELP(M, E, Q,...)>**undo**

To remove a file from a print queue, use DELETE/ENTRY.
For example:
delete/entry=1234 narrow
will remove the job with entry number 1234 from the queue named narrow.

HELP(M, E, Q,...)>**M**

To see what job number to use, type
show queue/all narrow

Another way to stop a file being printed is to re-name or delete it; this avoids having to know the job entry number.

HELP(M, E, Q,...)>**Q**

For the IA_LADYBUG system, over which we have total control, the manual, an on-line tutorial, and on-line HELP deliberately share material, specifically examples and text. Often, the tutorial and HELP present dynamic examples only summarized in the manual, for instance, one involving the ladybug crawling off the screen, or examples too complicated or whose effect (like color) would be lost in the manual. The simpler "start-up" and "reference" examples presented in the manual are the first examples presented in the tutorial and HELP. Both, but especially HELP, go on to present more complex or difficult examples, like counter-examples to show the limits of commands (e.g., RIGHT 362 exceeds the parameter range for degrees of turning). HELP also tunes its examples based on user-information like the user's directory (e.g., in DIR examples) or procedures already completed (e.g., in SEQUENCE examples).

Figure 1 gives an example scenario of user interactions with HELP in our second domain of application, VAX/VMS command language. (What the user types is indicated in **bold face**.) A few things should be noted: (1) the first sentence explaining PRINT is that used in the existing system documentation which doesn't contain examples; (2) the explanation given for "queue" not only reflects what has just been explained (PRINT), but also offers some information on synonomous qualifiers; (3) HELP relates the "undo" explanation with what has gone before and also provides an alternative way of accomplishing the same task. (4) HELP provides pragmatic knowledge; (5) HELP provides counter-examples, i.e., instances of "bad" usage.

3. Argumentation: Dynamic Hypotheticals

In our second line of research on examples, we have built a program that will generate hypothetical cases ("hypos"). One area of our current work concerns cases involving protection of property interests in software under trade secret law. Using prior decided cases as examples and guides, the program will modify the hypos to make them stronger or weaker cases in favor of the plaintiff or defendant. Hypos and cases are contained in an EKB and are both represented using similar frames. The frames have three or four levels of subframes presenting increasingly detailed factual information.

A trade secret case frequently involves two corporations, a plaintiff and defendant, who produce competing products. The plaintiff usually alleges that the defendant gained an unfair competitive advantage in developing and marketing its product by misappropriating trade secret information developed by the plaintiff. There are at least three stereotypical scenarios by which the defendant gains access to the plaintiff's trade secrets: (1) A former employee of the plaintiff with knowledge of the trade secrets enters into the defendant's employ and brings with him trade secret information which he learned while working for the plaintiff; (2) The plaintiff may disclose the "secret" information to the defendant perhaps in connection with an attempt to enter into a sales or other agreement with the defendant; (3) The trade secret information may be stolen from the plaintiff and passed to the defendant.

Frames and subframes have been defined to represent these typical trade secret fact patterns. *Figure 2* illustrates excerpts of frame structures representing the following hypothetical trade secret case, involving the first, "employee", scenario, named *RCAVICTIM v. SWIPEINC and Leroy Soleil.*

In the hypo, plaintiff RCAVICTIM sues defendants SWIPEINC and Leroy Soleil for misappropriation of trade secrets in connection with software developed by the plaintiff over a period of two years, from 1980 to 1982, with an expenditure of $2 million. Plaintiff markets the software, known as AUTOTELL, a program to operate a system of automated teller machines, to the banking industry. In 1982, computer whiz Leroy Soleil, one of plaintiff's key personnel on the AUTOTELL project, left RCAVICTIM and began working for SWIPEINC on a competing product, TELLERMATIC, also an automated teller program, which the defendant had just begun to develop. SWIPEINC managed to perfect TELLERMATIC also in about two years after spending about $2 million. RCAVICTIM claimed that SWIPEINC used trade secret information about AUTOTELL which Soleil brought with him.

3.1 Dimensional Analysis

In actual trade secret cases, the courts have decided a number of legal issues. For each issue decided, the court frequently identifies certain facts that it deems significant in making its "holding" in favor of a party [Levi, 1949]. The holdings of prior cases may be grouped into general categories that represent **dimensions** along which a hypo can be modified in ways that have legal significance for one or the other party. The dimensions factor a legal domain into basic modifications that affect the relative strengths of the parties' arguments and organize the prior cases in terms of how they can be used to guide modifying a hypo or to support a hypothetical party's argument.

Dimensions that have been identified in the trade secret case law [Gilburne & Johnston, 1982] and implemented in the program include the following:

1. **Unfair Competitive Advantage**: Plaintiff's argument is strengthened if the alleged trade secret information allowed defendant to gain a competitive advantage over plaintiff.

2. **Generally Known**: Plaintiff's argument is weakened if the alleged trade secret information is generally known within the industry.

3. **Learnable Elsewhere**: If the information was learned by an employee in his work for the plaintiff and he could have learned the information working for some other employer, plaintiff's argument is weakened.

4. **Vertical Knowledge**: Plaintiff's argument is weakened if the alleged trade secret information was about a vertical market. For example, cases imply that knowledge about a vertical market, such as knowledge of the structure of the banking industry, that an employee might learn in the course of developing computer programs for that market is not protectible as trade secret information.

5. **Telltale Signs of Misappropriation**: Plaintiff's argument is strengthened if there are certain telltale signs that the defendants sought to misappropriate the plaintiff's alleged trade secret information, e.g., that the corporate defendant paid a very high bonus to get the employee to bring with him a copy of the code he worked on for the plaintiff.

6. **Noncompetition Agreement**: Plaintiff's argument is strengthened if the employee entered into an agreement not to work for plaintiff's competitors.

7. **Accessible by Others**: Plaintiff's argument is weakened to the extent that plaintiff did not keep secret its alleged trade secret information by allowing an increasing number of other persons to have access to the information.

8. **Confidentiality Agreements Constraining Access**: Plaintiff's argument is strengthened to the extent that the persons with access to the trade secret information entered into agreements not to disclose the information to others.

3.2 Dimension- and Example-Directed Modification

Our HYPO program can modify a hypothetical case in favor of either party along any of the above dimensions. For instance, one simple way to modify the hypo in favor of plaintiff is to introduce the fact that SWIPEINC developed the competing software after 1982, the date when Soleil joined the company, at a considerable saving in time and money compared to plaintiff's expenditures. Such a modification is done so as to reflect the fact situation of an actual case in the knowledge base, thus allowing one to argue analogically for or against a party's position.

Suppose that there is a trade misappropriation case in the EKB, *JCN Corp. v. TEREX* , where the court held for plaintiff JCN and TEREX took two years and $1,000,000 to develop a product that JCN took four years and $2,000,000 to develop. The modification procedure simply computes the relative savings in development time and expenditures in the case from the EKB and modifies the appropriate slots in the hypo so that SWIPEINC also saved relatively the same amounts in developing the program. See *figure 2.* Under the new facts of the hypo, RCAVICTIM's attorney could cite *JCN Corp. v. TEREX* in favor of his client's position. If on the other hand, the hypo were to be modified in favor of the defendant, the procedure would decrease the relative savings in development expenditures so that SWIPEINC's attorney could distinguish *JCN Corp. v. TEREX* on the basis that his client did not save as much in development costs as TEREX did.

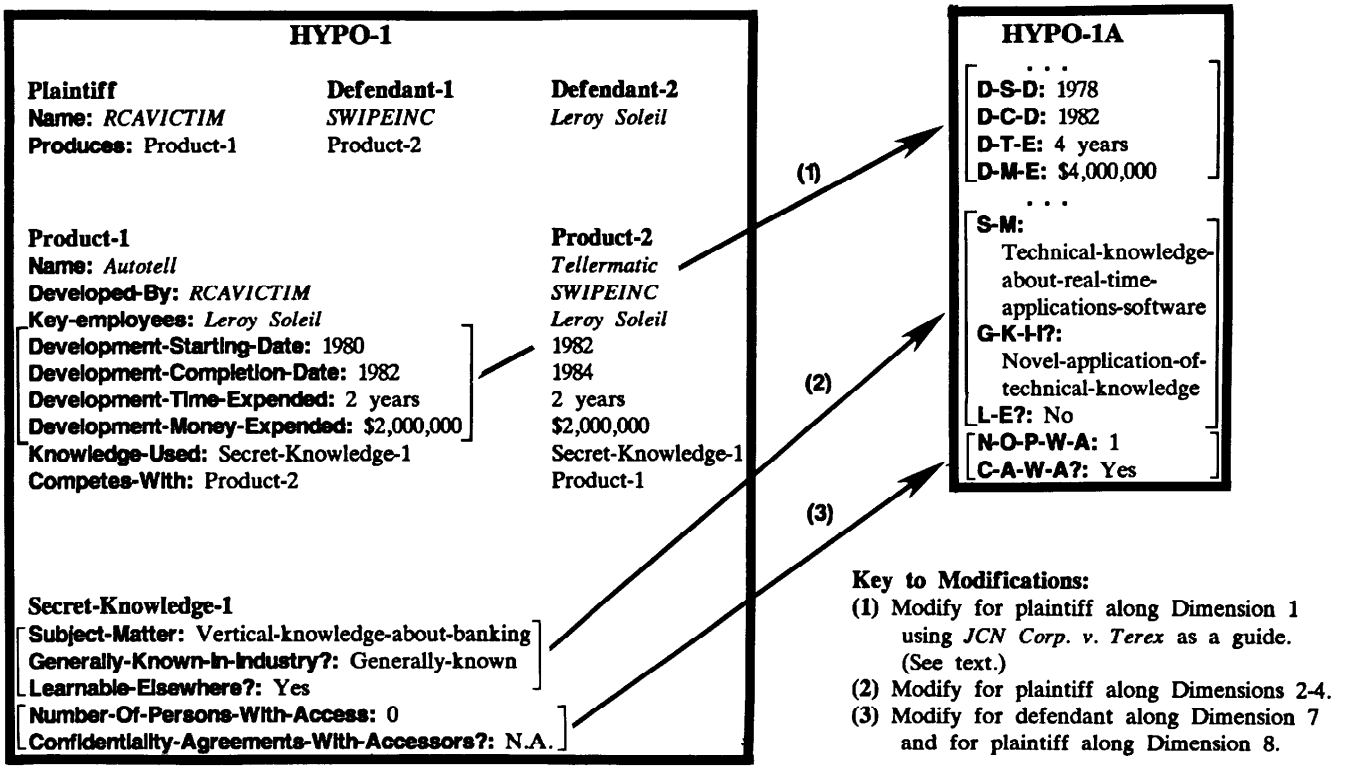

Figure 2. Hypo-1, *RCAVICTIM v. SWIPEINC & Leroy Soleil*, and Modifications

Figure 2 illustrates other modifications that strengthen plaintiff's argument in the hypothetical. The subject matter of the claimed trade secret can be characterized as technical information, e.g., about software engineering issues relevant to real time applications, as opposed to vertical knowledge about the banking business (Dimension 4). The technical knowledge can be characterized as applied in a novel way, newly discovered by plaintiff's personnel, or combined in a unique way with other technical knowledge, as opposed to being generally known within the industry (Dimension 2). It may be taken that Leroy Soleil would not have been able to learn such knowledge while working with any one other than the plaintiff (Dimension 3), or that SWIPEINC paid him a bribe to enter its employ (Dimension 5), or that Soleil brought with him a copy of the source code of plaintiff's program when he switched to SWIPEINC's employ (Dimension 5).

Modifications along each of the dimensions affect the values in some subset of the frame slots representing the hypo. The dimensions provide access to those cases in the database that could be cited or distinguished by virtue of the modification.

3.3 Limitations and Applications

The effects of the modifications on the relative strengths of the parties are not necessarily independent. For example, the hypo could be modified in favor of the plaintiff along Dimension 6 so that Leroy Soleil and plaintiff had entered into an enforceable noncompetition agreement. Now the effect of modifications along Dimensions 1 through 5 that otherwise would favor the defendant is rendered moot. That is, eventhough the plaintiff has a weak argument (along Dimensions 1-5) that the claimed secret knowledge is protectible as a trade secret, he may still be able to enforce the noncompetition agreement.

Another example of a collision in the effects of modifications along dimensions can be illustrated by modifying the hypo in favor of the defendant along Dimension 2 and for the plaintiff along Dimension 3. As a result, the claimed secret knowledge is both generally known within the industry and not learnable by employees working any where but with the plaintiff, a contradiction.

The modification procedures can be used to generate a "slippery slope" type sequence of hypos, a common feature of legal argument. Suppose the hypo is modified along Dimension 7 in favor of the defendant so that one other person, let us say a customer, has access to the information that plaintiff claims is a trade secret. This weakens plaintiff's argument because it implies that plaintiff did not treat the information as secret. If the hypo is modified along Dimension 8 in favor of plaintiff, that customer is made subject to a contractual obligation not to divulge the information it received from the plaintiff; plaintiff's argument that it treated the information as secret is restored. Suppose this sequence of modifications were repeated so that instead of one customer's having access to the information, twenty did. Plaintiff could still prevail since the corresponding modifications along Dimension 8 impose confidentiality agreements on all twenty customers. Now suppose that the sequence were repeated so that the number of customers with access were twenty thousand, two hundred thousand, two million. The gambit of imposing confidentiality agreements on all of the customers may not continue to satisfy plaintiff's burden of showing that it had kept secret the information. If 200,000 customers have access, even if they have entered into confidentiality agreements, from whom is the secret being kept? To the distributors of software to the mass market this hypothetical fact situation is of more than academic interest.

Modifications along one dimension may make other dimensions applicable or inapplicable. For example, as has already been mentioned, the hypo can be modified along Dimension 6 to introduce a noncompetition agreement. As a result of this modification, another dimension becomes applicable to the case:

> 9. **Duration of Noncompetition Prohibition:** Plaintiff's argument is weakened if the noncompetition agreement purports to prohibit the employee from working for competitors for too long a period.

The hypo can be modified to increase the time period for which the agreement purports to prevent the employee from competing, eventually to the point where the agreement is no longer enforceable by the plaintiff.

How long a prohibition against competition by the employee is too long to be enforced? The legal rule which purports to answer that question is that the covenant not to compete will be enforced as long as its terms are not unreasonable. Obviously such a rule provides a program little guidance in the modification of the hypothetical. Legal cases in the EKB which are relevant to Dimension 9, however, constitute specific examples of the application of this rule, complete with actual time periods that courts have deemed reasonable and others deemed unreasonably long. The modification procedure will use the actual time periods as guides in strengthening or weakening the plaintiff's argument. The cases indexed under Dimension 9 can be cited to justify the interpretation of the effect of the modification and to fashion an explanation of the argument by reference to the general rule enunciated in the cited cases.

A case from the EKB may participate in more than one dimension and be applied to modify a hypo along a dimension eventhough the case differs substantially from the hypo in other respects. Where a dimension involves slots whose values are not quantitative, more complex methods of modifying the slot values are necessary. The modifications must be made consistently within the context of the hypo's other facts, particularly the time ordering of significant events in the hypo.

4. Summary

In this paper we have examined two lines of research sharing the theme of examples and example generation. In the first, on-line explanation systems, there is no distinction made between real and hypothetical examples as there is in the second, legal argumentation. Both research programs rely heavily on a pre-existing corpus of examples, structured and represented in an **Examples-Knowledge-Base (EKB)** and the use of domain-specific procedures to modify existing examples to create new ones.

In each program, there are constraints on the selection and generation of new examples. In the case of on-line HELP, the constraints come from knowledge of the user, his task and context as well as the subject matter being explained. In the case of legal argumentation, the constraints come from internal consistency (e.g., of time) within the example, dimensional analysis, domain-specific doctrinal aspects, and the desired direction of the modification (i.e., stronger or weaker for plaintiff or defendant) with respect to the controlling case from the EKB. Particularly, in the argumentation examples there is the need to mediate between potentially conflicting constraints.

In our future work on argumentation and explanation, we plan to explore contextual knowledge, which relates to the constraints to be placed on the examples to be generated and on goal knowledge, of the user and arguer. Such research directions will involve deeper analyses of the structure of explanations and arguments, as well as of the knowledge and parties involved.

5. References

DEC (1978). *VAX/VMS Command Language User Guide.* Digital Equipment Corporation. Order No. AA-D023B-TE.

Dietterich, T., and Michalski, R. S. (1983). "A Comparative Review of Selected Methods for Learning from Examples". In Michalski, Carbonell & Mitchell (Eds.) *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing, CA.

Finin, T. W. (1983). "Providing Help and Advice in Task Oriented Systems". In *Proceedings IJCAI-83*. Karlsruhe, W. Germany.

Gilburne, M. R., and Johnston, R. L. (1982). "Trade Secret Protection for Software Generally and in the Mass Market". *Computer/Law Journal*. Vol III, No. 3 (Spring).

Houghton, R. C. (1984). "Online Help Systems: A Conspectus". *CACM*, Vol. 27, No. 2, February.

IBM (1983). *Disk Operating System by Microsoft, Inc.*. IBM Personal Computer Language Series, IBM Corp.

Kolodner, J. L. (1983). "Reconstructive Memory: A Computer Model". *Cognitive Science*. Vol. 7, No. 4.

Kuhn, T. S. (1970). *The Structure of Scientific Revolutions.* Second Edition. University of Chicago Press.

Lakatos, I. (1976). *Proofs and Refutations.* Cambridge University Press.

Levi, E. (1949). *An Introduction to Legal Reasoning.* University of Chicago Press.

Levine, L., and Woolf, B. (1984). "Do I Press Return?" In *Proceedings ACM-SIGCSE Symposium on Computer Science and Education*, Philadelphia, February.

McDonald, D. D. (1982). "Natural Language Generation as a Computational Problem: An Introduction". In Brady (Ed.) *Computational Theories of Discourse*, MIT Press.

Rissland, E. L. (1981). *Constrained Example Generation.* COINS TR 81-24, Department of Computer and Information Science, University of Massachusetts, Amherst.

Rissland, E. L. (1983). "Examples in Legal Reasoning: Legal Hypotheticals". In *Proceedings IJCAI-83*. Karlsruhe, W. Germany.

Rissland, E. L. (1984). "Learning to Argue: Using Hypotheticals". *Proceedings First Annual Workshop on Theoretical Issues in Conceptual Information Processing.* Atlanta, GA.

Rissland, E. L. (1978). "Understanding Understanding Mathematics" *Cognitive Science*, Vol. 2, No. 4.

Schank, R. S. (1984) "Explaining". Keynote talk at First Annual Conference on Theoretical Issues in Conceptual Information Processing, Atlanta, GA.

Shrager, J., and Finin, T. W. (1982). "An Expert System that Volunteers Advice". In *Proceedings AAAI-82*, Pittsburgh, PA, August.

Swartout, W., and Balzer, R. (1982). "On the Inevitable Intertwining of Specifications and Programs". *CACM*, Vol. 25, No. 7, July.

Wilensky, R. (1982a). "Talking to UNIX in English: An Overview of UC". In *Proceedings AAAI-82*, Pittsburgh, PA, August.

Wilensky, R. (1982b). *Talking to UNIX in English: An Overview of an On-line Consultant.* Report No. UCB/CSD82/104, Computer Science Division, University of California, Berkeley, September.