Case-Based Diagnostic Analysis in a Blackboard Architecture*

Edwina L. Rissland, Jody J. Daniels, Zachary B. Rubinstein, and **David B. Skalak**
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
rissland@cs.umass.edu

Abstract

In this project we study the effect of a user's high-level expository goals upon the details of how case-based reasoning (CBR) is performed, and, *vice versa*, the effect of feedback from CBR on them. Our thesis is that case retrieval should reflect the user's ultimate goals in appealing to cases and that these goals can be affected by the cases actually available in a case base. To examine this thesis, we have designed and built FRANK (Flexible Report and Analysis System), which is a hybrid, blackboard system that integrates case-based, rule-based, and planning components to generate a medical diagnostic report that reflects a user's viewpoint and specifications. FRANK's control module relies on a set of generic hierarchies that provide taxonomies of standard report types and problem-solving strategies in a mixed-paradigm environment. Our second focus in FRANK is on its response to a failure to retrieve an adequate set of supporting cases. We describe FRANK's planning mechanisms that dynamically re-specify the memory probe or the parameters for case retrieval when an inadequate set of cases is retrieved, and give an extended example of how the system responds to retrieval failures.

Introduction

This project places case-based reasoning (CBR) in a workaday context, as one utility for gathering, analyzing, and presenting information in service of a user's high-level task and viewpoint. A user's ultimate task might be to prepare a medical consultation as a specialist to an attending physician, to write a legal memorandum as a lawyer to a client, or to create a policy brief as an advisor to a decision-maker. For each task, what the writer (and his or her audience) plans to do with the information gathered affects the kind of information desired, the way it is found and analyzed, and the style in which it is presented. For instance, to generate a balanced, "pro-con" analysis of a situation, one would present in an even-handed manner the cases, simulations, and/or other analyses that support the various points of view. On the other hand, to create a "pro-position" report that advocates one course of action over all others, one would present information deliberately biased toward that point of view. Furthermore, if, in either situation, the retrieved cases only partially or meagerly support the intended presentation form, the user may have to temper his or her high-level goal by the information actually found, perhaps to the extent of radically revising a presentation stance or even abandoning it. Such revision may be required, for instance, if the cases destined for a balanced report are heavily skewed toward one side of an argument, or compelling cases for an opposing viewpoint subvert a proposed one-sided presentation.

To accommodate a variety of user task orientations, strategies, and viewpoints, we have designed and implemented a blackboard architecture that incorporates case-based and other reasoning mechanisms, a hierarchy of "reports" appropriate to different tasks, and a flexible control mechanism to allow the user's top-level considerations to filter flexibly throughout the system's processing. Our system, which is called FRANK (Flexible Report and Analysis System), is implemented using the Generic Blackboard toolkit (GBB) [Blackboard Technology Group, Inc., 1992] in the application domain of back-injury diagnosis.

Specifically, our goals in pursuing this project focus on two kinds of evaluation and feedback:

1. To investigate the effect of a failure to find useful cases upon the current plan or the user's task orientation; and, *vice versa*, the effects of the context provided by the user's task and viewpoint on case retrieval and analysis.
2. To build a CBR subsystem that can dynamically change its case retrieval mechanisms in order to satisfy a failed query to case memory.

We first give a broad sense of FRANK's overall architecture in the System Description and Implementation section, where we describe its control and planning mechanisms, particularly the two kinds of evaluation and feedback within the system. That section also describes the task hierarchies that are used by the control modules of the system: a reports hierarchy, a problem-solving strategies hierarchy, and a presentation strategies hierarchy. We follow this with an extended example where we present a scenario of FRANK's

*This work was supported in part by the National Science Foundation, contract IRI-890841, and the Air Force Office of Sponsored Research under contract 90-0359.

responses to case retrieval failure. A discussion of related research and a summary close the paper.

System Description and Implementation

Overview of FRANK

FRANK is a mixed-paradigm, report planning and generation system with a CBR component that has been implemented in a blackboard architecture. While we have selected the diagnosis of back injuries as the initial domain to illustrate its capabilities, the architecture is not specific to diagnostic tasks. In this domain, the user provides a description of a patient's symptoms and selects from a hierarchy of report types the type of report the system is to generate. The output of the system is a natural language report with appropriate supporting analysis of the problem.

The system's architecture is divided into the three basic components of control, domain reasoning, and report generation (see Figure 1). Control is provided by a planner that selects an appropriate plan from its library and then performs the hierarchical planning needed to instantiate it. Plan selection is based on the report type. Domain reasoning capabilities currently implemented include a CBR module with several processing options (e.g., similarity metrics) and an OPS5 production system, as well as knowledge sources that incorporate procedural reasoning. The domain reasoning capabilities are flexibly invoked by the planner to execute the plan. In particular, different types of case retrieval probes are created as necessary to complete query tasks set up by the plan. Finally, a report generator uses rhetorical knowledge to generate a report for the user. To support the various components, we have developed several hierarchies, which we describe next.

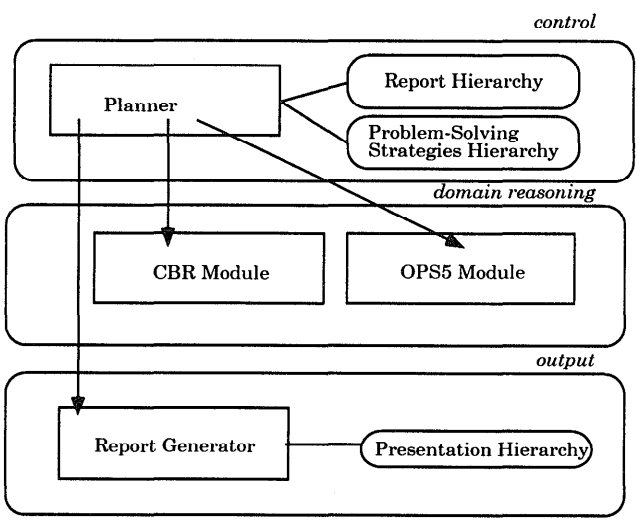

Figure 1: Overview of FRANK Architecture

Hierarchies

To support different expository goals, we have devised three hierarchies. The first hierarchy – the *Report hierarchy* – differentiates among reports based on expository goals. The second – the *Problem-Solving Strategies hierarchy* – represents the different problem-solving strategies inherent in finding, analyzing, and justifying the data that go into a report. A third hierarchy – the *Presentation Strategies hierarchy* – contains the methodologies and policies for presenting the material in its final form. The first two hierarchies support the planner, while the third helps guide report generation.

Report Hierarchy. Our first consideration in classifying reports is their overall goals. This is reflected in the first level in our hierarchy (see Figure 2). Reports are categorized based on whether they are *argumentative* or *summarizing* in nature, although in this paper we discuss the argumentative reports only. Argumentative reports are further subdivided into those that take a *neutral stance* and those that are *pro-position*, that is, endorse particular positions. Further subdivisions within the argumentative reports that take a neutral stance differentiate between reports that provide conclusions and those that do not.

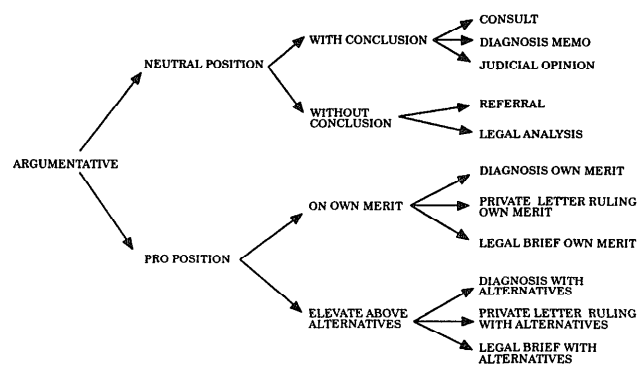

Figure 2: Report Hierarchy (partial)

Within the portion of the hierarchy that supports pro-position argumentative reports, there is a subdivision between reports that justify a position based soley on similar resulting conclusions (the *on own merit* category) and those that justify by additionally examining and discounting possible alternatives (the *elevate above alternatives* category). Examples of reports in these two subcategories are medical reports written from a pro-position viewpoint where there is a predisposition toward a particular conclusion: the *Diagnosis-Own-Merit* and the *Diagnosis-with-Alternatives* reports. A *Diagnosis-Own-Merit* report justifies a diagnosis, in part, by drawing analogies between the current situation and like cases. A *Diagnosis-with-Alternatives* report not only draws analogies to like cases but also discounts or distinguishes alternative diagnoses. Besides these reports from the medical domain, our report hierarchy contains similarly categorized reports for law [Statsky and Wernet, 1984] and policy analysis.

Associated with each report on a leaf node in this hierarchy is a list of groups of strategies. Each group serves as a retrieval pattern for accessing plans to carry out the processing needed to create the report. Currently, the plan that matches the greatest number of strategies is selected first.

Problem-Solving Strategies Hierarchy. Problem-solving strategies encode knowledge about how to perform the analysis necessary to generate a report. These strategies provide guidance on such matters as how to deal with contraindicative data and anomalies, the domain indices (e.g., factors) to use, how extensively to pursue the search for relevant data, what methodologies to use when correlating the data (e.g., pure CBR or CBR with rule-based support), and whether to include or exclude arguments that support alternative conclusions (see Figure 3).

Figure 3: Problem-Solving Strategies Hierarchy (partial)

Presentation Strategies Hierarchy. Presentation strategies guide the system in which aspects of a case to discuss and how to do so. They cover how to handle contraindicative information and anomalies in the output presentation, as well as how to report weaknesses in a position. Presentation strategies also suggest alternative orders for presentation of material within a report. Example presentation strategies are: (1) give the strongest argument first while ignoring alternatives, (2) state alternatives' weaknesses, then expound on the strengths of the desired position, or (3) concede weaknesses if unavoidable and do not bother discussing anomalies.

Control Flow

Top-Level Control. The top-level control flow in FRANK is straightforward. Each processing step in FRANK corresponds to the manipulation by knowledge sources (KSs) of data (units) in its short-term memory, which is implemented as a global blackboard (see Figure 4). The following knowledge sources represent the steps in the top-level control:

1. *Create-Input-KS:* Initially, FRANK is provided with a problem case, the domain, and user preferences for problem-solving and report presentation. The user may also specify the report type and a position to take as part of the preferences. This information is stored on an Input unit.
2. *Process-Input-KS:* This KS analyzes the problem case for quick, credible inferences that are then also stored on the Input unit. In addition, it creates a Report-Envelope unit that contains pointers to the Input unit, the problem case, and the domain, and has additional slots to maintain information about the current state of the problem solving. The Report-Envelope unit represents the context for the current problem-solving session.
3. *Select-Report-KS:* Using information from the Input unit and the Report-Envelope unit, this KS selects a report type and stores it on the Report-Envelope. Currently, the preferred report type must be input by the user.
4. *Extract-Strategies-KS:* Associated with each report type is a list of groups of strategies; these groups act as indices into a library of plans. This KS extracts a group from the list of groups and adds it to a Report-Envelope unit.
5. *Select-Plan-KS:* Using the extracted group of strategies, this KS selects and then stores a specific plan on the Report-Envelope unit. Initially, the plan having the greatest overlap of strategies with the Report-Envelope strategies is selected. (The selection process is described in the Plan Re-Selection section.)
6. *Execute-Plan-KS:* This KS instantiates the selected plan into a set of Goal units. The plan is executed by activating the top-level Goal unit. Leaf Goal units can use a variety of reasoning mechanisms such as model-based reasoning, OPS5, procedural reasoning, or CBR to achieve their respective tasks. The first step in all plans is to create a Report unit that specifies the presentation template to be used and contains slots for the necessary information to complete that template.
7. *Evaluate-Plan-Integrity-KS:* Upon plan completion, the results are evaluated by this KS to determine if the overall cohesiveness of the report is acceptable. Various alternatives, such as switching the plan or report type, are

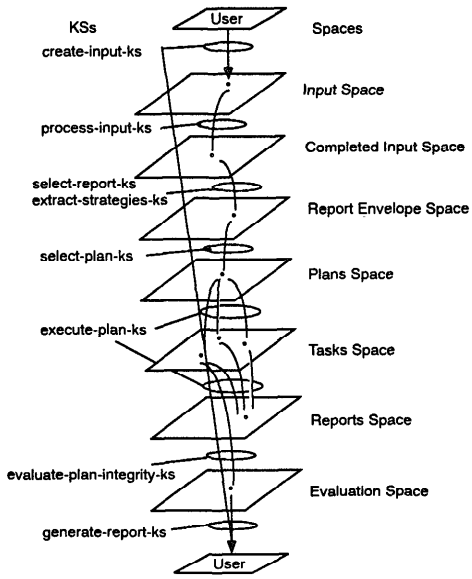

Figure 4: Top-Level Control Flow

available should the results be unacceptable.

8. *Generate-Report-KS:* The report is generated by filling out the template with the information stored on the Report unit.

Planning Mechanism

The planning mechanism directs the system's overall efforts to achieve the top-level expository goal. Plans can have dependent, independent, ordered, and unordered goals. Goal parameters may be inherited from supergoals and updated by subgoals. Ultimately, leaf goals invoke tasks through specified KS triggerings. Like plan goals, KSs can be dependent, independent, ordered, or unordered.

The KSs triggered by the leaf goals may use any of a variety of reasoning paradigms suitable for solving the corresponding leaf goal. A goal may be solved by procedural, rule-based, model-based, or case-based reasoning. Procedural and model-based reasonings are Lisp and CLOS modules. The rule-based element is OPS5. The CBR component is the CBR-task mechanism, described below.

Evaluation and Feedback

The analysis and report generation process has several layers. From the highest-level processing abstraction down to the actual executables, there are: reports, plans, goals/subgoals, and tasks/queries (see Figure 5). Currently, a user selects the report type, which indexes an initial plan based on a group of problem-solving strategies. The plan consists of a hierarchy of goals with leaf goals submitting tasks or queries for execution. The tasks/queries are the methodology-specific plan steps such as making inferences using rule-based reasoning (RBR) or finding the best cases using CBR. Replanning may be done at each level to achieve the report's expository goals.

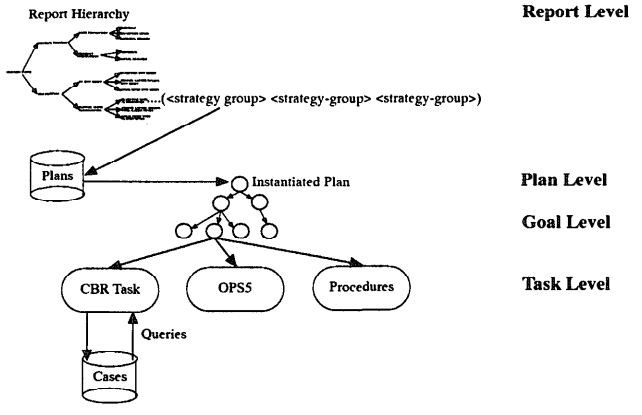

Figure 5: Levels of FRANK

There is a need to provide evaluation and feedback throughout the entire process of gathering, analyzing, and presenting information, rather than waiting until a report is finished to review the final product. Lower-level tasks attempt to rectify any problems they observe in order to lessen the impact on higher-level ones. However, if a process at a lower level has exhausted all its possible remedies, then it returns that feedback to its superior, which can then try a different approach. This type of feedback occurs at all levels of the system. For example, if a report relies on a particular type of information, like good supporting cases to make a pro-position argument, and the necessary information is not found through an (initial) query, then the report (initially) fails at the lowest level. Immediate reparations (e.g., changing particular CBR retrieval parameters) could be made at this level based on evaluation feedback to allow processing to resume without having to abort the whole reporting process.

The mechanism supporting this evaluation-feedback cycle is modeled on operating system vectored interrupts. In our case, the interrupts are unmet expectations detected by goals and the interrupt service routines (ISRs) are the remedies for the various interrupts. Instead of maintaining a single, global table of ISRs, FRANK supports multiple tables, found at the various levels, to permit specialization. When an interrupt occurs, the most local ISR is found by looking first at the table associated with the current goal, then the table of next super goal and so on. If no ISR is found, then a global table is used. While the global table is created at system initialization, the goal ISR tables are specified as part of the goal definition and are created at goal instantiation time.

Report Failures. A report fails only after all the possible plans for it have failed. If the user has not requested a specific report type, FRANK will automatically switch to a potentially more suitable type based on feedback from the failed plans. Otherwise, the user is prompted concerning the deficiencies in the report and he or she can request a new report type.

Plan Re-Selection. There are two general ways to select a new plan when the current plan fails. In the first, a priority-based (or local search of plans) approach, if there are more plans available under the current group of strategies stored on the Report-Envelope unit, the system can use one of these. Failing that, the system checks if there are any other groupings of strategies available under the report type to use as indices in selecting a new plan. Finally, if no other plans are available, then failure occurs, the problem(s) noted, and the system attempts to change the report type.

The second method of selecting a new plan is to use information about the failure to select a better alternative. For example, the *Diagnosis-with-Alternatives* report type requires cases supporting the advocated position to compare favorably to cases supporting alternative diagnoses. If no cases supporting the advocated position can be found, then no plan associated with the *Diagnosis-with-Alternatives* report type will be successful. If this failure occurs, the system switches the report type to *Diagnosis-Own-Merit*, which does not require supporting cases for the advocated position.

Leaf Goals and CBR Methodologies. FRANK supports several CBR methodologies. Currently, two basic methodologies have been implemented: nearest-neighbor and HYPO-style CBR [Ashley, 1990]. Each CBR methodology brings with it different means of retrieving cases, measuring similarity, and selecting best cases. Having multiple types of CBR available allows the system to invoke each type to support the others and to lend credibility to solutions. The flexibility of having different CBR approaches to draw upon also allows the system to apply the best type in a particular context. For example, if no importance rankings can be attached to a set of input-level case features, but a collection of important, derived factors can be identified, a HYPO claim lattice can identify a set of best cases, whereas nearest neighbor retrieval based on input-level features may be less successful if many features are irrelevant. A HYPO claim lattice is a data structure used to rank cases in a partial ordering by similarity to a problem situation according to higher-level domain factors [Ashley, 1990].

FRANK tries to satisfy the current leaf goal's CBR requirement with the most suitable methodology. Should feedback indicate that another methodology may be better suited to the problem, FRANK automatically makes the transition while retaining the feedback about each method. Should no method be able to satisfy the requirement, or only partially satisfy it, then the higher-level goals receive that feedback and decide how to proceed.

CBR-Task Mechanism. The CBR-task mechanism is one of the KSs operating at the lowest level of the planning hierarchy. It controls queries to the case base. Depending on the plan being used to generate a report, a CBR query may be more or less specific or complete. By grouping queries into classes according to what they are asking and what they need as input, this mechanism is able to fill out partial queries. It completes them with viable defaults and then submits them. If a query is unsuccessful, then the CBR-task mechanism alters the values it initially set and resubmits the query, unless prohibited by the user. Again, this level of processing provides feedback concerning the success, partial success, or failure to the next higher process.

Extended Example

The following example demonstrates some of the flexibility of FRANK. In particular, it shows evaluation and reparation at the various levels of the system. The overall motivation for this example is to illustrate how top-level goals influence the type of CBR analysis done and how results during CBR analysis can affect the top-level goals.

Suppose the user wants a pro-position report justifying the diagnosis of spinal stenosis for a problem case. The first step is to select a report type and problem-solving strategies. Given the user input, FRANK selects a pro-position report type called *Diagnosis-with-Alternatives*. Since the user does not specify a problem solving strategy, a default one is used. FRANK selects stronger problem-solving strategies when given a choice. In this case, the default strategy is to make an equitable comparison of an advocated position and the viable alternatives. In particular, the advocated position is considered justified if it is supported by "Best Cases."

There are a variety of definitions for "Best Case" and, as for the problem-solving strategies, FRANK is predisposed to selecting stronger (less inclusive) definitions over weaker (more inclusive) ones. Initially, a Best Case must satisfy three criteria: (1) be a Most On-Point Case (MOPC), (2) support the advocated position, and (3) not share its set of dimensions with other equally on-point cases that support an alternative position (i.e., there can be no equally on-point "competing" cases). In turn, FRANK currently has two definitions for a MOPC: (1) a case that shares the maximal number of overlapping symptoms with the problem situation ("maximal overlap"), or (2) a case in the first tier of a HYPO claim lattice. (These definitions are distinct because cases in the first tier of a claim lattice can share different subsets of dimensions with the problem and these subsets may have different cardinalities.)

FRANK uses the default problem-solving strategy and the above Best Case definition to select a plan associated with the *Diagnosis-with-Alternatives* report type. The selected plan generates goals for finding the Best Cases, collecting the diagnoses associated with them, and then comparing and contrasting these cases.

First Query. The subgoal for finding the Best Cases creates a *Best-Cases* query, specifying the above Best Case definition. Two other case retrieval parameters, (1) whether nearly applicable ("near miss") dimensions are considered during case retrieval and (2) the MOPC definition, are unassigned and are set by the CBR-task mechanism to the defaults of "no near misses" and the maximal overlap definition. The first attempt to satisfy the query results in no Best Cases because all the MOPCs found support diagnoses other than spinal stenosis, thereby violating criterion (3) of the Best Case definition.

Local Modifications. The CBR-task mechanism then alters one of the parameters it set, in this case allowing near misses, and resubmits the query. Again, no Best Cases are found because of competing cases so the CBR-task mechanism changes the MOPC definition from "maximal overlap" to "HYPO Claim Lattice" and resubmits the query. The CBR-task mechanism continues to alter the parameters it has control over and resubmitting the query until either some Best Cases are found or it has exhausted the reasonable combinations it can try. In this example, the query fails to find any Best Cases and returns "no cases" back to the subgoal.

CBR-Task Mechanism Interrupt. When the query returns "no cases" back to the subgoal to find Best Cases, an interrupt is generated to handle the failure. The interrupt is caught by the ISR table related to the subgoal and a remedy to weaken the definition of Best Cases is tried.

New Best Case Definition. The *Best-Cases* query is modified to remove Best Case criterion (3) above, to include as Best Cases those for which equally on-point competing cases may exist. The revised query is submitted and, this time, it returns a set of Best Cases. The subgoal is marked as satisfied and the next goal of the plan is activated.

The next subgoal analyzes each diagnosis used in the set of MOPCs to determine which symptoms of the problem case are and are not covered by the diagnosis. That is, a table is created in which each row contains a viable diagnosis and the columns are the problem case's symptoms. If there is a MOPC for a diagnosis and the MOPC shares the problem case's symptom, then the entry is marked.

Since the current plan compares the strengths of the diagnoses, the symptoms covered by spinal stenosis are compared to the symptoms covered by the alternative diagnoses. The strategy employed here is to conclude that if a symptom is only found in the MOPCs supporting spinal stenosis, then the importance of that symptom is elevated. Unfortunately, in this example, all of the symptoms covered by the spinal stenosis diagnosis are also covered by other diagnoses.

Global Interrupt. Because there are no distinguishing symptoms, at this point an interrupt is generated signifying that the alternatives are too strong. The "too strong alternatives" interrupt is caught by the global ISR table and the corresponding remedy is tried, to try a report type based on the position's own merits. Since the user initially requested a comparison of her position against alternatives, FRANK asks the user if it can make a switch from *Diagnosis-with-Alternatives*. The user agrees and FRANK selects the *Diagnosis-Own-Merit* report type. FRANK now selects and instantiates a plan associated with the *Diagnosis-Own-Merit* report type. Suppose that this time the plan does complete satisfactorily. The resulting data representation of the justification is used by the text generation module to create the actual report and present it to the user.

Related Research

This work extends our previous work on case-based reasoning, mixed-paradigm reasoning, and argumentation, particularly our work on hybrid-reasoning systems that use a blackboard to incorporate a CBR component, including ABISS [Rissland *et al.*, 1991] and STICKBOY [Rubinstein, 1992]. FRANK uses opportunistic control analogous to HEARSAY II [Erman *et al.*, 1980] to better incorporate both top-down and bottom-up aspects of justification than in our previous, rule-based approach to control in CABARET [Rissland and Skalak, 1991]. FRANK also extends our task orientation from mostly argumentative tasks, as in HYPO and CABARET, to more general forms of explanation, justification, and analysis. Other mixed-paradigm systems using blackboard architectures to incorporate cases and heterogeneous domain knowledge representations are the structural redesign program FIRST [Daube and Hayes-Roth, 1988], and the Dutch landlord-tenant law knowledge-based architectures PROLEXS [Walker *et al.*, 1991] and EXPANDER [Walker, 1992].

ANON [Owens, 1989] uses an integrated top-down and bottom-up process to retrieve similar cases. Abstract features are extracted from a current problem and each feature is used to progressively refine the set of similar cases. As the set of similar cases changes, it is used to suggest the abstract features that may be in the current problem and used for further refinement.

TEXPLAN [Maybury, 1991], a planner for explanatory text, provides a taxonomy of generic text types, distinguished by purpose and their particular effect on the reader. This system also applies communicative plan strategies to generate an appropriately formed response corresponding to a selected type of text. TEXPLAN is designed as an addition to existing applications, rather than as an independent domain problem solver.

While FRANK explains failures as part of the evaluation and reparation it performs at various levels, the explanation is not used to determine the appropriateness of a case as in CASEY [Koton, 1988] and GREBE [Branting, 1988], nor is it used to explain anomalies as in TWEAKER [Kass and Leake, 1988] and ACCEPTER [Kass and Leake, 1988]. FRANK's use of explanation in plan failure is similar to CHEF's [Hammond, 1989] in that it uses the explanation of a failure as an index into the possible remedies. However, CHEF's explanation is provided by a domain-dependent casual simulation, whereas FRANK's failure analysis is based on the generic performance of its own reasoning modules, such as the failure of the CBR module to retrieve an adequate collection of supporting cases.

Summary

Our general focus in this paper has been the interaction between a user's high-level expository goal and its supporting subgoal tasks, such as to retrieve relevant cases. Having set ourselves two research goals in the introduction, we have shown first how the FRANK system, a hybrid blackboard architecture, can create diagnostic reports by tailoring case-based reasoning tasks to the user's ultimate goals and viewpoint. In particular, we have given an example of how FRANK uses feedback from tasks such as CBR to re-select a plan. Finally, in pursuit of our second research goal, we have demonstrated how FRANK can re-specify the way case retrieval is performed to satisfy a plan's failed request for case support.

References

Ashley, Kevin D. 1990. *Modelling Legal Argument: Reasoning with Cases and Hypotheticals*. M.I.T. Press, Cambridge, MA.

Blackboard Technology Group, Inc., 1992. *GBB Reference: Version 2.10*. Amherst, MA.

Branting, L. Karl 1988. The Role of Explanation in Reasoning from Legal Precedents. In *Proceedings, Case-Based Reasoning Workshop*, Clearwater Beach, FL. Defense Advanced Research Projects Agency, Information Science and Technology Office. 94–103.

Daube, Francois and Hayes-Roth, Barbara 1988. FIRST: A Case-Based Redesign System in the BB1 Blackboard Architecture. In Rissland, Edwina and King, James A., editors 1988, *Case-Based Reasoning Workshop*, St. Paul, MN. AAAI. 30–35.

Erman, Lee D.; Hayes-Roth, Frederick; Lesser, Victor R.; and Reddy, D. Raj 1980. The Hearsay-II Speech-

Understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys* 12(2):213–253.

Hammond, Kristian J. 1989. *Case-Based Planning: Viewing Planning as a Memory Task.* Academic Press, Boston, MA.

Kass, Alex M. and Leake, David B. 1988. Case-Based Reasoning Applied to Constructing Explanations. In *Proceedings, Case-Based Reasoning Workshop*, Clearwater Beach, FL. Defense Advanced Research Projects Agency, Information Science and Technology Office. 190–208.

Koton, Phyllis A. 1988. *Using Experience in Learning and Problem Solving.* Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA.

Maybury, Mark Thomas 1991. *Planning Multisentential English Text using Communicative Acts.* Ph.D. Dissertation, University of Cambridge, Cambridge, England.

Owens, Christopher 1989. Integrating Feature Extraction and Memory Search. In *Proceedings: The 11th Annual Conference of The Cognitive Science Society*, Ann Arbor, MI. 163–170.

Rissland, Edwina L. and Skalak, David B. 1991. CABARET: Rule Interpretation in a Hybrid Architecture. *International Journal of Man-Machine Studies* 1(34):839–887.

Rissland, E. L.; Basu, C.; Daniels, J. J.; McCarthy, J.; Rubinstein, Z. B.; and Skalak, D. B. 1991. A Blackboard-Based Architecture for CBR: An Initial Report. In *Proceedings: Case-Based Reasoning Workshop*, Washington, D.C. Morgan Kaufmann, San Mateo, CA. 77–92.

Rubinstein, Zachary B. 1992. STICKBOY: A Blackboard-Based Mixed Paradigm System to Diagnose and Explain Back Injuries. Master's thesis, Department of Computer and Information Science, University of Massachusetts, Amherst, MA.

Statsky, William P. and Wernet, R. John 1984. *Case Analysis and Fundamentals of Legal Writing.* West Publishing, St. Paul, MN, third edition.

Walker, R. F.; Oskamp, A.; Schrickx, J. A.; Opdorp, G. J. Van; and Berg, P. H.van den 1991. PROLEXS: Creating Law and Order in a Heterogeneous Domain. *International Journal of Man-Machines Studies* 35:35–67.

Walker, Rob 1992. *An Expert System Architecture for Heterogeneous Domains: A Case-Study in the Legal Field.* Ph.D. Dissertation, Vrije Universiteit te Amsterdam, Amsterdam, Netherlands.