Diagnostic Case Retrieval Guided by Evaluation and Feedback *

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 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 examples of how the system responds to retrieval failures.

Introduction

This project investigates how expository goals and justification strategies affect how case-based problem solving is performed, and conversely, how the intermediate results derived from using cases to solve a problem dynamically influence ultimate goals and the strategies applied to achieve them. The system described in this paper adopts the role of a domain expert charged with the task of analyzing a situation and creating a report of a form that reflects certain expository constraints. Such constraints could include the perspective or bias that the report should reflect, the level of the writer's expertise, and the report's intended audience.

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 designed 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 effects of the context provided by the user's task and viewpoint on case retrieval and analysis; and, *vice versa*, the effect of a failure to find useful cases upon the current plan or the user's task orientation.
2. To build a case-based reasoning (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. It 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 limited 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 pseudo-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.

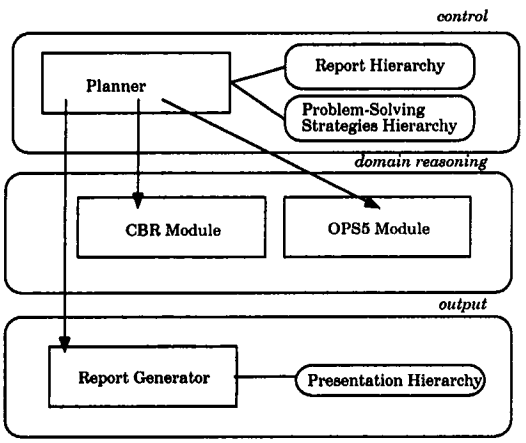

Figure 1: General Description of FRANK Architecture

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 Strategy hierarchy* – represents the different problem-solving strategies inherent in finding, analyzing, and justifying the materials 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. 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 *proposition*, that is, endorse particular positions (see Figure 2). Further subdivisions within the argumentative reports that take a neutral stance differentiate between reports that provide conclusions and those that do not.

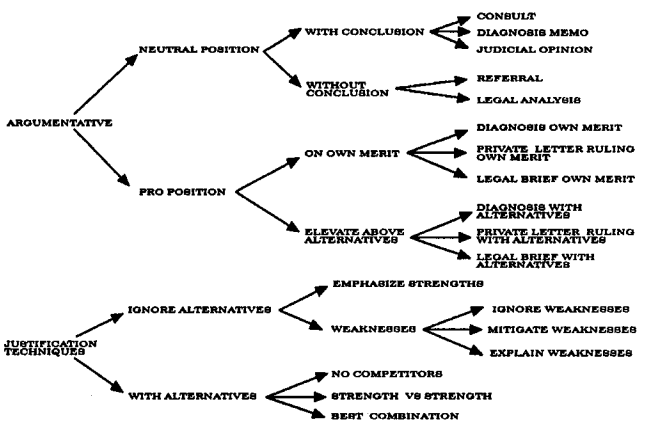

Figure 2: Partial Hierarchies: Report (top) and Problem-Solving (bottom)

Within the portion of the hierarchy that supports proposition argumentative reports, there is a subdivision between reports that justify a position based solely 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 solely 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.

Problem-Solving and Presentation Strategies Hierarchy Problem-solving strategies encode knowledge about how to perform the analysis necessary to generate a report (see

Figure 2) while Presentation strategies guide the system in which aspects of a case to discuss and how to do so. Groups of strategies drawn from these two hierarchies are associated with each leaf node on the report hierarchy. These groups serve 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.

Control Flow

Top-Level Control 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. The top-level control flow in FRANK is straightforward. Basically, FRANK is provided with a problem case, the domain, and user preferences, which are analyzed for quick, credible inferences. A Report-Envelope unit is created that represents the context for the current problem-solving session. Next, a KS selects a report type, a group of strategies, and a plan. The selected plan instantiates into a set of Goal units that are then activated. The first step in all plans is to create a Report unit that specifies the presentation template to be used. Upon plan completion, the overall cohesiveness of the report is evaluated. Various alternatives, such as switching the plan or report type, are available should the report be unacceptable. Finally, the report is generated.

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 3). 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.

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.

The mechanism supporting this evaluation-feedback cycle is modeled on vectored interrupts. In this case, the interrupts are unmet expectations detected by goals and the interrupt service routines (ISRs) are the remedies for the various interrupts. Instead of having just a global table of ISRs, FRANK supports multiple tables at the various levels to permit specialization. When an interrupt occurs, the

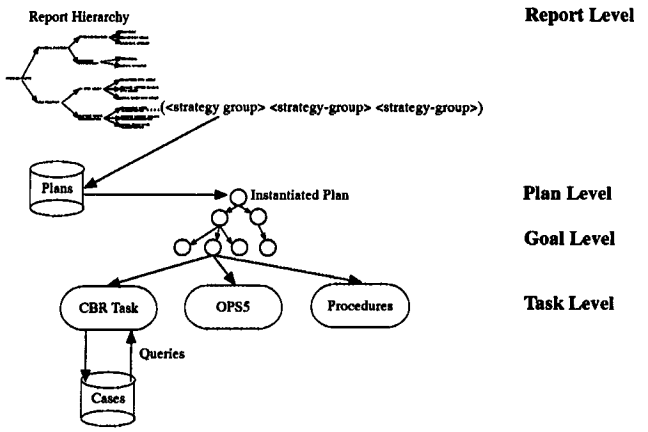

Figure 3: Levels of FRANK

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 tables are specified as part of the goal definition and are created at goal instantiation time.

Report and Plan Failure and Re-Selection 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 notified of report deficiencies and may request a new report type.

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, 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 used when, based on the information about the failure, no relevant plan can be found under the existing report type. In this case, a new report type is chosen and plan selection proceeds. For example, if a plan fails because it cannot find any good supporting cases, then no plan associated with a Diagnosis-With-Alternatives report type will be successful. Based on the information about the failure, the system switches the report type to Diagnosis-Own-Merit, which does not require supporting cases.

Leaf Goals and CBR Methodologies Not only does FRANK support RBR and CBR at the leaf goal level, it also supports a variety of CBR methodologies. Currently, two basic methodologies have been implemented: nearest-neighbor and HYPO-style CBR [Ashley, 1990]. Each CBR methodology brings with it a different means of retrieving cases, measuring similarity, and selecting best cases. Hav-

ing multiple types of CBR allows the system to benefit by the types mutually supporting each other and lending credibility to solutions. It also allows the system to draw upon the best type of reasoning for a particular context. For example, if there is no importance ranking for a set of features, but a set of relevant factors can be identified, then a HYPO claim lattice will yield a set of best cases, where trying to use a nearest neighbor metric may not be reasonable.

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. Queries to the CBR-task mechanism are subdivided into types according to what they are asking and what they need as input, such as those seeking Best cases or a particular position. Associated with each query type is specific information about processing a partial query of this type. The mechanism completes any partial query with viable defaults and then submits it.

If a query is unsuccessful, then the CBR-task mechanism alters the values it has the latitude to manipulate and resubmits the query. This process continues until either favorable results are achieved, or the mechanism has no further reasonable parameter values to try and failure occurs. An example of an unreasonable parameter is one that further restricts a search after a result of "no cases found" is returned by the query. When either a result is found or all options have been exhausted, the query returns to the leaf goal. At this point, the goal evaluates the results and, if necessary, generates an interrupt. In the case of a query-prompted interrupt, the query is passed along with the interrupt and provides feedback to the ISR regarding what was already attempted.

Extended Example

We next show by example how evaluation and feedback can guide the reparation of queries. We demonstrate how results from a CBR query can influence the altering of top-level goals. Further, we illustrate FRANK's flexibility in responding to top-level goals and how these goals influence the type of CBR analysis done. A second example can be found in [Rissland *et al.*, 1993].

We derive this example from a patient's experience when he or she walks into a general practioner's office with a set of symptoms. The patient expects the doctor to objectively analyze his or her symptoms and conclude the most probable diagnosis. As initial input, FRANK receives the patient's symptoms and expectations. FRANK uses the expectations to choose the appropriate report type. In this case, FRANK chooses the *Diagnosis* report type, a report with a neutral stance that draws a conclusion. FRANK uses the groups of problem-solving strategies associated with the Diagnosis report type to select a plan. If the patient has a preference for one style of justification over another, FRANK receives that preference as initial input and uses it to filter the groups of strategies, selecting only the groups that are compatible with the constraints. Of the resulting groups of strategies, the first group is used as indices to select a plan from the repository of plans. For this example, assume that some of the preferences are for using CBR to find candidates then RBR for selection, and the extent of information gathering is broad. Additionally, contraindicative information will be used when relevant, as will anomalies.

In this case, the selected plan has a goal to make the diagnosis, which is composed of two sequential subgoals. The first subgoal is to find the Most On-Point cases (MOPCs). The second is to choose a diagnosis from the MOPCs based on their coverage of the patient's symptoms. Currently, FRANK has two definitions for MOPCs: (1) any case that shares the maximal number of overlapping symptoms ("maximal overlap"), or (2) any case in the first tier of a Claim Lattice as described in HYPO ("claim lattice".) Cases that are in the first tier can share different subsets of dimensions with the problem and these subsets may have different cardinalities. The subgoal for finding the MOPCs creates a *Find-MOPCs* query, without specifying a particular MOPCs definition. The CBR-task mechanism fills in the definition by choosing the first of the available definitions. In this case, the "maximal overlap" definition is used. The query returns with a set of cases.

The next subgoal finds the potential diagnoses by taking the union of the diagnoses found in the MOPCs. Associated with each diagnosis is the union of all the symptoms of the found cases with that particular diagnosis. Each diagnosis is compared to the problem case by determining the overlap of the diagnosis' symptoms and the patient's symptoms. The diagnosis with the largest overlap is considered the most probable. For this patient, there are multiple probable diagnoses. This plan breaks ties by appealing to other reasoning paradigms such as rule-based reasoning. For example, if a diagnosis is very rare, then it is devalued as a probable diagnosis. In this case, all the potential probable diagnoses are equally likely and no definitive diagnosis can be made.

CBR-Task Mechanism Interrupt. At this point, an interrupt is generated to handle the ambiguity of equally likely diagnoses. The interrupt is caught by the ISR table related to the goal of making a diagnosis and a remedy to find a better set of MOPCs is attempted. The Find-MOPCs query is modified to try another definition of MOPCs. The query is submitted without a specified definition but removing the "maximal overlap" from the possible definitions. The CBR-task mechanism fills in the definition with the "claim lattice" MOPC definition and the new set of MOPCs is returned from the query.

New MOPC Definition. As with the previous set of MOPCs, the current ones are analyzed and compared to the problem case in terms of the possible diagnoses and their coverage. Unfortunately, in this example, this set of

MOPCs fares no better than the first set and is unable to yield a definitive diagnosis. Again, an interrupt is generated and the remedy to find a different set of MOPCs is tried. In this case, the query is resubmitted, but there are no more available definitions for MOPCs and the query fails to find any cases.

Global Interrupt. Another interrupt is generated, this time signifying the inability to find suitable MOPCs. Since the selected plan contains no remedies for this problem, the interrupt is caught by the global ISR table. The corresponding remedy for "diffused support" it to try another report type that does not require a conclusion. FRANK chooses the Referral report type because it remains neutral but does not require a definitive conclusion. It proceeds by executing a new plan that is selected by using the problem-solving strategies associated with the Referral report type. This new plan is successful and FRANK generates a Referral.

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, agenda-based approach 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-based 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]. Although 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, it does not try to repair plans. The possible remedies are strategies for selecting a new plan, not modifying an existing plan.

Summary

Our general focus in this paper has been the interaction between a user's high-level expository goal and the supporting tasks, such as CBR, needed to support it. 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.

Rissland, Edwina L.; Daniels, Jody J.; Rubinstein, Zachary B.; and Skalak, David B. 1993. Case-Based Diagnostic Analysis in a Blackboard Architecture. In *Proceedings, The 11th National Conference on Artificial Intelligence*, Washington, D.C. AAAI. Forthcoming.

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.