Accepted for Presentation at the International Conference on Case-Based Reasoning, Vancouver, August, 2001

Helping a CBR Program Know What it Knows

Bruce M. McLaren[1] and Kevin D. Ashley[2]

[1] OpenWebs Corporation
2403 Sidney Street
Pittsburgh, Pennsylvania 15203-2116
bmclaren@openwebs.com

[2] University of Pittsburgh
Intelligent Systems Program
3939 O'Hara Street
Pittsburgh, PA 15260
ashley+@pitt.edu

Abstract. Case-based reasoning systems need to know the limitations of their expertise. Having found the known source cases most relevant to a target problem, they must assess whether those cases are similar enough to the problem to warrant venturing advice. In experimenting with SIROCCO, a two-stage case-based retrieval program that uses structural mapping to analyze and provide advice on engineering ethics cases, we concluded that it would sometimes be better for the program to admit that it lacks the knowledge to suggest relevant codes and past source cases. We identified and encoded three strategic metarules to help it decide. The metarules leverage incrementally deeper knowledge about SIROCCO's matching algorithm to help the program "know what it knows." Experiments demonstrate that the metarules can improve the program's overall advice-giving performance.

Introduction

Humans typically know the limits of their own expertise. They can recognize when they have seen problems before and whether the solutions to the past problems are likely to lead to a solution in a new situation. Self-knowledge of the limitations of ones expertise is a hallmark of intelligence and a characteristic that would greatly benefit expert systems, especially case-based ones.

Earlier work in AI addressed the issue of an expert system's knowing the limits of its expertise using *meta-knowledge* [Lenat *et al.*, 1983, chapter 7; Davis and Buchanan, 1985]. In that work, one could "draw upon a very useful body of knowledge: [a] model of what it is you do and do not know, what tasks you can and cannot tackle, and how long it would take you to solve some problem." [Lenat *et al.*, 1983, p. 234] A program so equipped could respond to some situations by simply admitting its ignorance instead of venturing risky advice. This line of research led to the development of systems such as TEIRESIAS [Davis, 1982].

While the problem of imbuing a program with strategic knowledge of its own limitations is still important, it has been largely ignored in recent years. In particular, the case-based reasoning community has focused little attention on the issue. This may be due, in part, to the goal of many CBR systems to provide "quick-and-dirty" solutions: such an objective does not appear to comport well with the overhead of employing deeper strategic knowledge. It may also be a result of the emphasis by most CBR programs on finding the most relevant past case or cases, given a current case base. A past case may be the most similar of *known* cases, but it still may not be similar enough at a deeper level to warrant its use as the basis of a solution. Providing criteria for making that decision, however, may require representing more knowledge than is customary in most CBR approaches.

In applying CBR techniques in sensitive or dangerous domains, such as medicine or air traffic control, knowing what one knows is a particularly important characteristic. It is one thing to take a "best guess" at a real estate assessment, quite another to suggest a dangerous medical procedure on the basis of a weakly supported diagnosis. A CBR program operating in the latter domain would be well advised to admit to its uncertainty in such situations.

One could use techniques to construct and maintain "competent" case bases, which cover a range of cases that might be presented to a system [Smyth and Keane, 1995; Smyth and McKenna, 1999]. The case base essentially becomes a sample distribution of possible target problems. This approach relies on techniques for deleting from and editing a case base so that the case base itself defines what the program "knows." Competent case bases appear to be useful in domains in which the retrieval technique is well understood (e.g., nearest neighbor) and in which a single “best case” is retrieved, but they may have less import in domains in which retrieval is performed at both a surface and structural level and in which multiple cases support a result. In such situations, computing competence may be a prohibitively difficult task.

We have been working with a problem domain, engineering ethics, in which incorrect suggestions could be considered sensitive and damaging. Engineering ethics deals with ethical dilemmas and problems that occur in professional engineering, and these problems typically involve delicate issues such as public safety, confidentiality between an engineer and a client, and duty to an employer [Harris *et al*., 1999]. We have developed a CBR program, SIROCCO, which provides advice for engineering ethics problems by applying a two-stage retrieval algorithm [McLaren, 1999; McLaren and Ashley, 2000]. Because of its sensitive nature, our chosen domain could clearly benefit from a program that knows its limitations. At the same time our two-stage retrieval algorithm, which involves both surface and structural mapping, requires an approach other than competent case bases.

After analyzing and experimenting with SIROCCO's retrieval algorithm and representation, we have devised certain metarules to assess whether a given retrieval has been successful. Given the most relevant known cases, the metarules assess whether the program has previously encountered cases enough like the problem to warrant giving advice. The metarules leverage strategic knowledge about the operation of the retrieval algorithm and about what successful retrievals look like.

As noted, SIROCCO's two-stage algorithm matches at two levels: a surface level and a deeper structural level. In addition, the algorithm depends on collecting a *set* of reasonably matched cases rather than a single best match. This led to defining three

metarules. Given a target case and a set of retrieved source cases, each rule provides evidence that the program cannot suggest advice for the target case:

- **Metarule 1**: If the best superficial match between the target and source cases is a weak surface-level match of critical and questioned facts, then SIROCCO may be inadequate for the task.
- **Metarule 2**: If the top N superficially-matched source cases do not share enough citations to the same underlying ethical principles (i.e., codes), then SIROCCO may be inadequate for the task.
- **Metarule 3**: If the best deep match between the target and source cases is a weak structural match, then SIROCCO may be inadequate for the task.

The rules are listed in order of increasing sensitivity to depth of analysis. The first assesses the surface-level match with respect to certain important facts. The second accounts for how well the source cases overlap conceptually with respect to underlying ethical principles. The third assesses the quality of the structural mapping to source cases. While we believed that the third rule provides SIROCCO with the strongest evidence of the limitations of its knowledge, we also believed that the first two rules could provide useful evidence.

We constructed specific implementations for each metarule and performed a series of experiments to better understand their potential benefits, alone and in various combinations. In the remainder of this paper, we describe our program, provide examples of the usefulness of the metarules in deciding when to provide advice, and report the results of our experiments.

Overview of SIROCCO

SIROCCO, an interpretive CBR program, extends techniques that have been applied to the legal domain [Ashley, 1990; Branting, 1991, Rissland *et al.*, 1996] to the domain of engineering ethics. SIROCCO contributes a detailed, narrative case representation, including temporal relations between facts, and an extensional model of general principles and cited cases. It can retrieve cases over a wider range of factual scenarios than the AI & Law programs, but unlike those programs it does not make arguments for or against a conclusion. Rather, it provides suggestions that can help a human construct a reasoned argument.

The program accepts a target case expressed in the Ethics Transcription Language (ETL) and produces suggestions about relevant code provisions and past cases. A sample target case is shown in Figure 1; it deals with an engineer who has discovered structural defects in an apartment building but has been told he must keep that information confidential. SIROCCO's output for the case is shown in Figure 2, at the top of which is a textual description of the case facts. SIROCCO's output is essentially a series of "possibly relevant" codes and past cases that the engineer should consider in analyzing the target case.

1. Apartment Building <**may be hazardous to safety**>.	Pre-existing fact
2. Apartment Building Owner **<owns>** Apartment Building.	Occurs during 1
3. Residents of Apartment Building **<reside in>** Apartment Building.	Occurs during 1, 2
4. Residents of Apartment Building <**file a lawsuit or arbitration action against**> Apartment Building Owner <**because**> (Apartment Building <**may be hazardous to safety**>).	Occurs during 3
5. Apartment Building Owner **<is legally represented by>** Owner's Attorney**.**	Occurs during 4
6. Owner's Attorney **<hires the services of>** Engineer A **<for>** (Engineer A **<inspects>** Apartment Building).	Occurs during 4, 5
7. Engineer A **<inspects>** Apartment Building.	Occurs during 6
8. Engineer A <**discovers that**> (Apartment Building <**fails standards and may be hazardous to safety**>).	Occurs during 7
9. Engineer A <**knows**> (Government Authority <**should be informed about the hazard or potential hazard**>).	Occurs during 8
10. Engineer A <**informs**> Owner's Attorney <**that**> (Apartment Building <**fails standards and may be hazardous to safety**>).	Immediately after the conclusion of 8
11. Owner's Attorney **<instructs>** Engineer A **<to>** (Engineer A **<withholds information from>** Anyone Else **<regarding>** Apartment Building).	After the conclusion of 10
12. Engineer A <**does not inform**> Anyone Else <**that**> (Apartment Building <**fails standards and may be hazardous to safety**>). *[Questioned fact]*	After the conclusion of 11

Fig. 1. The Fact Chronology of Case 90-5-1

***** SIROCCO is analyzing Case 90-5-1**

Facts: Tenants of an apartment building sue the owner to force him to repair many defects in the building that affect the quality of use. The owner's attorney hires Engineer A to inspect the building and give expert testimony in support of the owner. Engineer A discovers serious structural defects in the building, which he believes constitute an immediate threat to the safety of the tenants. The tenants' suit has not mentioned these safety-related defects. Upon reporting the findings to the attorney, Engineer A is told he must maintain this information as confidential as it is part of a lawsuit. Engineer A complies with the request of the attorney.

Question: Was it ethical for Engineer A to conceal his knowledge of the safety-related defects in view of the fact that it was an attorney who told him he was legally bound to maintain confidentiality?

***** SIROCCO's suggestions for evaluating Case 90-5-1:**

*** ***Possibly Relevant Codes:***

I-4: Act as a Faithful Agent or Trustee
III-4: Do not Disclose Confidential Info. Without Consent
I-1: Safety, Health, and Welfare of Public is Paramount
II-1-A: Primary Obligation is to Protect Public (Notify Authority if Judgment is Overruled). …
II-1-C: Do not Reveal Confidential Info. Without Consent
III-2-B: Do not Complete or Sign Documents that are not Safe for Public …

*** ***Possibly Relevant Cases:***

76-4-1: Public Welfare - Knowledge of Information Damaging to Client's Interest
89-7-1: Duty To Report Safety Violations
84-5-1: Engineer's Recommendation For Full-Time, On-Site Project Representative

Fig. 2. SIROCCO's Output for Case 90-5-1 (excerpts)

SIROCCO represents ethics cases as narratives, expressed in a limited language. As shown in Figure 1, ETL represents the actions and events of a scenario as an ordered list (i.e., a *Fact Chronology*) of individual sentences (i.e., *Facts*), each consisting of:

1. *Actors and Objects*, instances of general actors and objects which appear in the scenario (e.g., "Owner's Attorney," "Engineer A," "Apartment Building"),
2. a *Fact Primitive*, the action/event in which the actors/objects participated (e.g., "owns," "inspects," "discovers that"), and
3. a *Time Qualifier*, a temporal relation between the Fact and other Facts (e.g., "Occurs during," "After the conclusion of").

At least one Fact in the Fact Chronology is designated as a Questioned Fact; this is an action or event corresponding to an ethical question raised in the scenario. ETL conforms to the well-formed grammar defined in [McLaren and Ashley, 1999].

The *Time Qualifiers* are disjunctive compositions of Allen's temporal constraints [1983], and SIROCCO uses a time propagation system, TIMELOGIC [Koomen 1989], to compute time relations not explicitly provided.

SIROCCO's source cases (i.e., the cases in its case base) are represented as Fact Chronologies together with templates that represent the case analysis. Each analysis comes from the text of a large corpus of cases published by the National Society of Professional Engineers Board of Ethical Review (NSPE BER). Each template contains attributes for the conclusion (i.e., ethical, unethical, or undecided), the protagonist whose action is questioned, the general argument structure, and information about each code or past case cited in the analysis of the case. Figure 3 shows part of the template filled in for Code II.1.a.[1], cited in Case 90-5-1.

Code	II.1.a
Code Status	Violated
…	**…**
Why relevant	Engineer's judgment is overruled in a particular professional circumstance. [11] Overruling the Engineer's judgment may lead to the endangerment of the safety, health, property or welfare of the public [8, 9]
Why violated, not violated,…	In the given situation, Engineer does not hold paramount the safety, health, property, and welfare of the public. [12]

Fig. 3. Extracts from the template for Code II.1.a in Case 90-5-1

A key aspect of SIROCCO is its use of operationalization techniques. Case analyses provide extensional definitions of ethical principles and case citations, and SIROCCO operationalizes those definitions to help it focus on the most important facts of cases. *Code* and *case instantiations*, the two most important of the nine techniques used by SIROCCO, relate a questioned fact, critical facts, and the temporal sequence of those facts in the citing case to the cited code or case. For instance, in

[1] Code II.1.a. states: "Engineers shall hold paramount the safety, health, and welfare of the public. If engineers' judgment is overruled … they shall notify their employer or client and such authority as may be appropriate."

Figure 3, Code II.1.a. is operationalized. The numbers in brackets in the last two rows refer to the facts of Case 90-5-1 (Figure 1) that are critical to the code's application and explain why it was violated (or not). Code II.1.a. is thus connected extensionally to a real case's relevant facts and chronology in a way that SIROCCO can reuse.

The program uses two abstraction hierarchies to determine similarities between Facts and Codes. The *Action/Event Hierarchy* clusters and generalizes similar Fact Primitives. The *Code Hierarchy* clusters codes dealing with similar issues.

SIROCCO's case base consists of 242 cases. 184 of the cases are referred to as the *foundational cases*. These were used to design, implement, and refine the program. The other 58 are *trial cases*. These were represented by independent case enterers and were used to run earlier experiments, reported in [McLaren, 1999; McLaren and Ashley, 2000], in which we compared SIROCCO's capabilities to a full-text retrieval program and an ablated version of the program. (SIROCCO performed significantly better than both the full-text retrieval system and the ablated version of the program in those experiments.) The cases deal with a variety of issues including public safety, confidential information, duty to employer, credit for engineering work, proprietary interests, and honesty in reports and public statements.

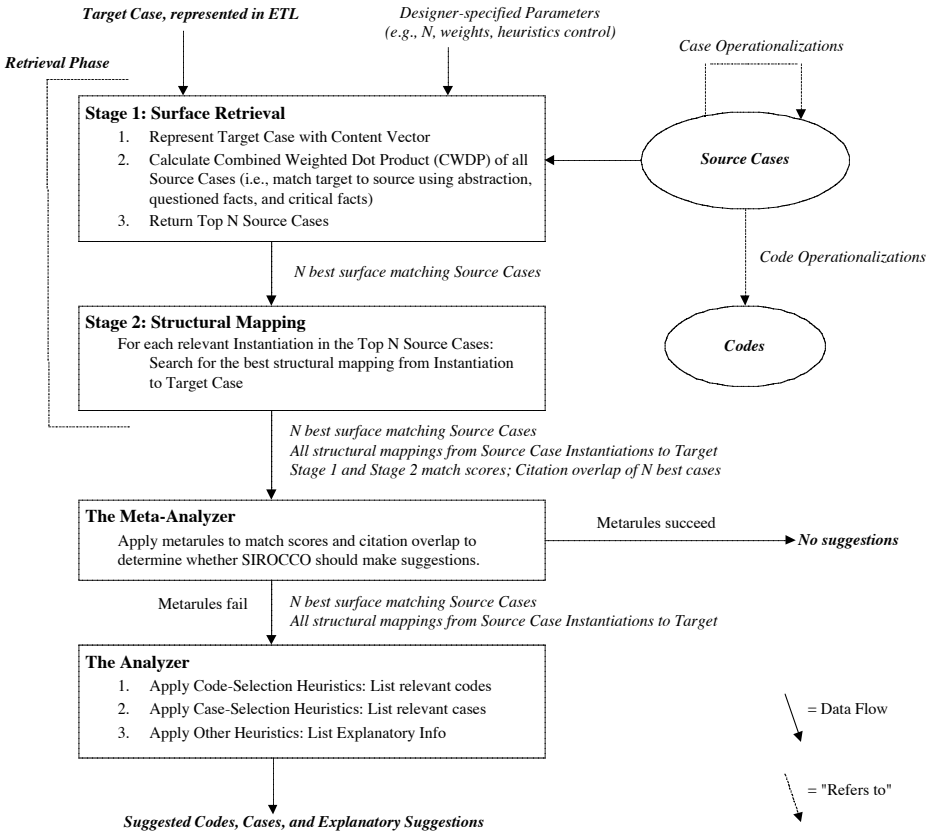

Fig. 4. SIROCCO's Architecture

SIROCCO's Two-Stage Retrieval Process

As indicated in Figure 4, once SIROCCO receives a target case represented in ETL, it engages in a two-stage process to retrieve relevant code provisions and source cases from its database. Stage 1 rapidly matches the Fact Primitives of the target case to those of all possible source cases. Stage 2, employs A* search in a more-refined structural mapping.

In both stages, the code and case instantiations help SIROCCO focus attention on the most critical facts. Stage 1's accuracy improves by giving more weight to the instantiations' Fact Primitives. Focusing on only a subset of a source case's Facts, those critical Facts cited in an instantiation, makes Stage 2's structural mapping routine more efficient and accurate.

For Stage 1, the target case and all of the source cases are represented as content vectors [Forbus *et al.*, 1994]. Each vector summarizes the Fact Chronology of a single case. It specifies the Fact Primitives, and their abstractions, and a count of how many times each one appears. Figure 5 shows two content vectors for Case 90-5-1. The content vector on the left represents the lowest level of abstraction; the vector on the right is one level higher in the Action/Event Hierarchy. There are a total of four predefined levels in the hierarchy. The facts with three asterisks ("***") correspond to the questioned fact(s) of the case.

Fact-Primitive-CV:	**Fact-Group-CV:**
(May -be-Hazardous-to-Safety 1)	(Deal-with-Potential-Dangers-or-Hazards 1)
(Owns 1)	(Own-Something 1)
(Resides-in 1)	(Specify-Location-of-Residence 1)
(Files-a-Lawsuit-or-Arbitration-Action-Against 1)	(Initiate-Legal-or-Arbitration-Proceedings 1)
(Is-Legally-Represented-by 1)	(Has-Legal-Representation 1)
(Hires-the-Services-of 1)	(Work-as-an-Employed-or-Contract-Professional-Engineer 1)
(Inspects 1)	(Perform-Engineering-Analysis-Review-or-Testing-Work 1)
(Discovers-That 1) (Knows 1)	(Know-or-Believe-Something 2)
(Informs-That 2) ***	(Disclose-Information 2) ***
(Instructs-to 1)	(Order-Subordinate-to-Perform-Task 1)

Fig. 5. Two of the Content Vectors for Case 90-5-1

Stage 1 computes a *combined weighted dot product* (CWDP) for every source case and outputs a list of the top N source cases (experimentation reported in [McLaren, 1999] determined N = 6 to be optimal) ranked by descending CWDP scores. Pre-defined weights are assigned as parameters to matches at the four abstraction levels and to matches of a source case's critical and questioned facts.

Since, as described in the next section, our implementation of metarule #1 deals with CWDPs, it is worth explaining them in some detail. Given a target case (T) and a source case (S_n), the CWDP for S_n is:

$$CWDP = WDP + (QFW_x * MWDP) + (CFW_y * MWDP) \quad (\mathbf{1})$$

$$WDP = (W_1 * DP_1 / MDP_1) + (W_2 * DP_2 / MDP_2) + (W_3 * DP_3 / MDP_3) + (W_4 * DP_4 / MDP_4) \quad (\mathbf{2})$$

Where:

- WDP is the weighted dot product; 0 <= WDP <= 1.0
- QFW_x is the pre-defined questioned fact weight at the most specific match level; QFW_x <= 1.0
- CFW_y is the pre-defined critical fact weight at the most specific match level; CFW_y<= 1.0
- MWDP is the maximum weighted dot product (WDP) over all source cases.
- $W_1 + W_2 + W_3 + W_4 = 1.0$ (Pre-defined weights corresponding to each abstraction level).
- $DP_1 \ldots DP_4$ are the normalized dot products of T and S_n at each abstraction level.
- $MDP_1 \ldots MDP_4$ are the maximum dot products at each abstraction level over all source cases.

The dot products (i.e., DP_1 to DP_4) are calculated as in standard vector arithmetic, by summing the product of matching vector elements. Since the calculation tends to favor long fact chronologies, the dot products are normalized by dividing by the sum of the content vector elements, which is roughly the length of a fact chronology. Each dot product is also normalized by dividing it by the maximum dot product (MDP_1 to MDP_4) over all source cases at that abstraction level.

Here is the intuition behind the CWDP. The weighted dot product (WDP) captures how well facts of two cases match both exactly and at more abstract fact levels. The contribution of an exact match is more than that of an abstract match (the standard weights applied by SIROCCO are $W_1 = 0.53$, $W_2 = 0.27$, $W_3 = 0.14$, and $W_4 = 0.06$, each level twice the weight of the previous level, considering rounding). The combined weighted dot product (CWDP) accounts for matching of questioned and critical facts between the target and source case. For both questioned and critical facts an extra element is added to the WDP that is equivalent to the best WDP over all source cases, times a weight that corresponds to the abstraction level of the match. Questioned facts are considered more important than critical facts since they are the focus of the ethical analysis of a case. This difference in relative importance is reflected in the standard abstraction-level weights: 1.0, 0.5, 0.25, and 0.125 for questioned facts, 0.333, 0.111, 0.036, and 0.012 for critical facts. Matching a questioned fact can have a dramatic effect; the weighting factor is applied to the *highest* WDP over *all* source cases.

Given SIROCCO's standard weights, the maximum possible CWDP is 2.333: 1.0 for the WDP, 1.0 for the questioned fact addend (QFW_x * MWDP), and 0.333 for the critical fact addend (CFW_y * MWDP). The CWDP only occasionally exceeds 2.0. For the example case (Figure 1), one on which SIROCCO performs well, the top 6 CWDP scores ranged from 1.97 to 2.27. A more typical range is 1.2 to 1.9.

Stage 2 employs a heuristic A* search for each target case / candidate source case instantiation pair to attempt a structural mapping between the two. Unlike the use of

A* search in GREBE [Branting 1991], SIROCCO takes temporal relations into account and supports abstract matches. The goal is to map each of the Facts of the source instantiation to a corresponding Fact in the target case while maintaining a one-to-one and consistent mapping between Actors and Objects across the cases.

Since, as described below, our implementation of metarule #3 tests values of the cost function, $f(n)$, of the A* search, we describe in some detail how it is calculated. The initial node of the search space maps the source's questioned Actor to the target's. Each subsequent node corresponds to a new mapping of a Fact in the source instantiation to a Fact in the target. The solution depth is equal to the number of Facts in the source instantiation. Expanding a new node requires four conditions to be met: (1) a match of a pair of Facts at the same level of abstraction, one from the source instantiation and one from the target, (2) preservation of a one-to-one Fact mapping between the source and target (i.e., no Fact can be mapped more than once), (3) preservation of a one-to-one, consistent set of Actor and Object mappings entailed by the Fact mappings, and (4) consistent temporal relations between mapped Facts of the source and target. Temporal relations are consistent if the Allen relations of every pair of mapped source Facts intersect with the Allen relations of the corresponding pair of target Facts.

As new nodes are expanded a *mismatch score* is calculated for each. The score is based on the abstraction level at which the target and source Facts match: 0.0 is assigned for an exact match, 0.4 for a match at the first abstraction level, 0.6 for the second level, 0.9 for the third level, and 1.0 for a complete mismatch. The mismatch score at a node is cumulative; it sums all of the mismatched values between target and source Facts along the current search path. The mismatch score is used by the A* cost function, $f(n) = g(n) + h'(n)$, to calculate the "goodness" of a node. The $g(n)$ function is the mismatch score divided by the current search depth. The $h'(n)$ function provides the most optimistic possible completion of the mapping from node n to the goal node; it is the mismatch score divided by the solution depth.

The goal node is reached when the current depth equals the solution depth and either the current node has the lowest $f(n)$ score of all open nodes or the nodes list is empty.

Mapping Level	**Facts of Source Case Inst. III.4. (Case 76-4-1)**	**Mapped Target Facts (Case 90-5-1)**
1st abstraction level	2. Engineer Doe **<reviews and analyzes>** Discharge	7. Engineer A **<inspects>** Apartment Building
Exact Match	3. Engineer Doe **<discovers that>** (Discharge **<fails standards and may be hazardous to safety>**).	8. Engineer A **<discovers that>** (Apartment Building **<fails standards and may be hazardous to safety>**).
Exact Match	11. Engineer Doe **<does not inform>** Control Authority **<that>** (Discharge **<fails standards and may be hazardous to safety>**).	12. Engineer A **<does not inform>** Anyone Else **<that>** (Apartment Building **<fails standards and may be hazardous to safety>**).

Fig. 6. Mapping of Code Instantiation III.4. of Case 76-4-1 to Case 90-5-1

Consider the search node in Figure 6, showing a mapping of a source instantiation, Code III.4, to the example target case in Figure 1. SIROCCO has succeeded in mapping all three Facts of the instantiation. In fact, the node in this figure represents the goal node that was reached in this search. Only one of the three Facts was not matched exactly, the 1st abstraction level match of the first Fact, and, since this is a goal node, the current depth and solution depth are both equal to 3. Thus, the cost function for this node is (0.4 / 3) + (0.4 / 3) = 0.27. It is usually more convenient to think of f(n) as a *match percentage*. Because 0.27 measures mismatch and the maximum possible mismatch is 2.0, the match percentage of an instantiation is (2.0 – *f(n)*) / 2.0. Thus, the match percentage of the node in Figure 6 is 1.73 / 2.0 = 86.7%.

The Meta-Analyzer and Analyzer

SIROCCO's decision to provide advice based on how well the target matches the source cases is made in the Meta-Analysis step of the SIROCCO architecture depicted in Figure 4. As described in the next section, this step applies the metarules.

Assuming that SIROCCO decides to provide suggestions, i.e., the metarules fail, its final stage (the Analyzer shown in Figure 4) uses other knowledge to decide which codes to suggest to the user. For instance, the Code-Selection Heuristic "Good Match to Questioned Facts of Code Instantiation" is satisfied if the f(n) is sufficiently low (<= 1.0) and a Questioned Fact in the source and/or target is matched. Because the example mapping of Figure 6 meets this criterion (step 11 of Case 76-4-1 and step 12 of Case 90-5-1 are Questioned Facts of their respective cases), SIROCCO suggests Code III.4 as relevant (see Figure 2).

SIROCCO blends the results of multiple source cases into its output, rather than relying on a single best match. It attempts to find a structural mapping to the instantiations of *all* the top N superficially matched source cases, and can provide suggestions based on all of them. Another of SIROCCO's Code-Selection Heuristics, "Frequent Occurrences in the Top Cases," is based on this feature. If a code is cited by 33% of the top N cases, evidence accumulates for the selection of this code in SIROCCO's suggestions. As described below, the implementation of metarule #2 also employs common code citations as evidence of whether the program has found good source cases for a target.

Helping SIROCCO Know What it Knows

To imbue SIROCCO with the knowledge of its limitations we carefully considered its architecture, experimented with the foundational cases, and added a new step, the Meta-Analyzer, to the program (Figure 4). Our goal was to determine which results from the retrieval phase provide the best indicators of when the program should or should not provide advice for a case.

Our hunch was that SIROCCO's deepest knowledge, i.e., the results it generates in the second stage, would be the most useful. However, it is clear that even the program's first stage contains some deeper knowledge, since the dot product algorithm

accounts for abstract matches and matches to critical and questioned facts. Moreover, we wanted to test whether the program could make use of critical information across the top cases it retrieves, such as shared code and case citations.

To explore these issues we identified 30 foundational cases that SIROCCO processed correctly (i.e., cases for which the program made accurate code and case suggestions, according to the experiment described in [McLaren and Ashley, 2000]) and 28 foundational cases that SIROCCO processed incorrectly. We collected data about SIROCCO's processing of each of these target cases, including the stage 1 scores of the top 6 matching source cases, the amount of code and case citation overlap between the top 6 stage 1 source cases, and the top stage 2 scores. By analyzing and experimenting with a number of variants, we came up with the following three metarules, which are implementations of the general metarules discussed previously. These are the rules applied in the Meta-Analysis step depicted in Figure 4.

- **Metarule 1**: If the best Stage 1 CWDP <= 1.2, then SIROCCO may not have good enough source cases for this target case.
- **Metarule 2**: If at least 3 of the top 6 Stage 1 cases do not share a code citation, then SIROCCO may not have good enough source cases for this target case.
- **Metarule 3**: If the best match percentage of a source instantiation < 80%, then SIROCCO may not have good enough source cases for this target case.

These rules screened out a reasonably high percentage of the target cases for which there were no sufficiently good source cases, while at the same time not eliminating many of the targets for which there were good source cases. While comparing internally-computed values against thresholds is a simple way of applying and testing meta-knowledge, the important contribution of the above rules is the identification of the important aspects of the problem solver to be monitored and tested.

As an illustration of how the metarules can help, consider the effect of some variations of the example case in Figure 1. SIROCCO handles Case 90-5-1 well, and thus one would expect that none of the metarules would block the program from providing suggestions. In fact, none of the three metarules fires for the original version of Case 90-5-1.

Suppose, however, that we scramble Case 90-5-1 such that all of the same Facts are used but their time ordering is reversed. For instance, Fact 12 becomes a Pre-existing fact, Fact 11 occurs "After the conclusion of 12," Fact 10 occurs "After the conclusion of 11," and so on. This obviously produces a nonsensical, but superficially similar, version of the original case. Because of the identical Facts in the two cases, SIROCCO's Stage 1 process retrieves the same set of source cases and assigns the same CWDP scores for the scrambled case as for the original case. Thus, metarules 1 and 2 do not fire. However, because the temporal sequence has been altered in the new case, SIROCCO is not able to find a strong structural mapping to any source case instantiation. As a result, metarule 3 fires, correctly suggesting that SIROCCO should not attempt to make suggestions for this case.

Now consider a plausible variation of Case 90-5-1 in which the time sequence of the Facts is not perturbed, but new Facts are added and the Questioned Fact is changed. Engineer B is a fellow employee of Engineer A at Company X. Instead of the owner's attorney hiring Engineer A directly, the attorney hires Company X. As in the original scenario, Engineer A discovers problems with the apartment building but

does not inform anyone, including his colleague Engineer B. Since Engineer B does not know about the defects, he obviously does not inform anyone of them either. However, Engineer B's action of "not informing" is questioned in the new version of the case. In this variation, the scenario is clearly plausible, but the ethical question raised (i.e., whether an engineer who doesn't know something should report what he doesn't know) is probably not what one would expect to see in the case base. Again, metarules 1 and 2 do not fire, because of the substantial amount of superficial Fact overlap between the original and variant case. However, changing the Questioned Fact from Engineer A's action to Engineer B's action substantially alters SIROCCO's search process, and no source case instantiation provides a good match to the variant target case. Again, the deepest metarule, rule 3, correctly fires, providing evidence that SIROCCO should not suggest codes and source case citations for this target case.

These examples focus on metarule 3 and, in fact, this rule appears to have the greatest utility. However, the intuitions underlying metarules 1 and 2 are fairly clear. Metarule 1 can eliminate target cases that lack abstract, questioned fact, or critical fact matches that may still have enough structural match to qualify under metarule 3. In practice (and in the experiments reported below), however, we found this metarule makes a fairly small contribution. Metarule 2 takes into account two key aspects of SIROCCO's approach: using multiple cases in a solution and using common citations to underlying code provisions as evidence that the source cases deal with a relevant issue. The experiments showed that this rule has more utility than metarule 1.

Evaluation of SIROCCO's Ability to Know What it Knows

Two series of experiments were performed to test the capabilities of the metarules: one series that tested them against the foundational cases and the other tested them against the trial cases. Each series compared the results of tests of six combinations of the metarules with the experimental results reported in [McLaren and Ashley, 2000], and with a test in which SIROCCO rejected target cases at random. In each of the six metarule tests, a different combination of metarules was used to screen out targets for which there were no good source cases. For the foundational tests, each of the 184 cases was run as a target case with the remaining cases as source cases. For the trial tests, each of the 58 cases was run as a target against 241 source cases, the remaining 57 trial cases plus the 184 foundational cases. The six metarule tests were: three tests in which each of the rules was applied by itself (eliminating the target case if satisfied), a test in which rules 2 and 3 were applied (eliminating the target case if either rule was satisfied), a test in which all three rules were applied (eliminating the target case if any rule was satisfied), and a test in which all three rules were applied, but at least 2 of the 3 rules would have to be satisfied to eliminate the target case. The random test consisted of running all target cases through SIROCCO 10 times, with 1 of every 3 cases randomly eliminated during each of the 10 runs. The random result was calculated as an average of the 10 runs.

SIROCCO's performance for each target case was quantified by calculating overlap of the program's suggested codes and cases with the NSPE BER's code and case citations for the same case. We used the *F-measure,* an information retrieval metric

that combines precision *P* and recall *R*: $F = (ß^2 + 1)PR / (ß^2 P + R)$ with ß = 1.0 [Lewis *et al.*, 1996]. Two F-Measure values were computed for each target case, one representing exact matches of codes and source cases between SIROCCO's solution and the Board's and one representing inexact matches. Inexact matches of codes were determined using the Code Hierarchy. Inexact matches of source cases were determined using a citation overlap metric, inversely related to the shortest citation path between two cases.

The results are shown in Figures 7 and 8. The tables show the mean exact and inexact F-Measure for each test, the number of target cases that remained after applying the metarules, the number of target cases that were eliminated by the metarules, and the mean exact and inexact F-Measures for the eliminated cases. (For experimental purposes, SIROCCO always provided suggestions – and we calculated F-Measures – even for the "eliminated" cases.) In Figure 7 note that there are only 179 foundational cases (target cases remaining plus cases eliminated); five cases were dropped from the original 184 because the board cited either no code or obsolete codes.

Metarules Applied	Mean Exact F-Measure (Remaining)	Mean Inexact F-Measure (Remaining)	Target Cases Remaining	Target Cases Eliminated	Mean Exact F-Meas. (Eliminated)	Mean Inexact F-Measure (Eliminated)
MR 2 & 3	0.368	0.552	67	112	0.246	0.426
MR 1, 2, & 3	0.368	0.552	67	112	0.246	0.426
MR 3	0.331	0.538	95	84	0.247	0.4
2 of 3 MR	0.316	0.501	149	30	0.168	0.336
MR 2	0.324	0.49	128	51	0.211	0.43
MR 1	0.304	0.491	167	12	0.125	0.226
Random	0.292	0.473	118.2	60.7	0.287	0.469
No metarules	0.292	0.473	179	0	NA	NA

Fig. 7. Results of Applying Metarules to the Foundational Cases

Metarules Applied	Mean Exact F-Measure (Remaining)	Mean Inexact F-Measure (Remaining)	Target Cases Remaining	Target Cases Eliminated	Mean Exact F-Measure (Eliminated)	Mean Inexact F-Measure (Eliminated)
MR 2 & 3	0.238	0.541	30	28	0.184	0.378
MR 1, 2, & 3	0.217	0.539	27	31	0.208	0.396
MR 3	0.233	0.501	36	22	0.178	0.4
2 of 3 MR	0.227	0.476	52	6	0.078	0.342
MR 2	0.221	0.496	50	8	0.155	0.255
MR 1	0.215	0.471	49	9	0.194	0.415
Random	0.218	0.466	36.9	21.1	0.200	0.451
No metarules	0.212	0.462	58	0	NA	NA

Fig. 8. Results of Applying Metarules to the Trial Cases

Discussion

It is clear from the tables in Figures 7 and 8 that the metarules had a positive effect. Every metarule test improved SIROCCO's F-Measures as compared to SIROCCO running without metarules and SIROCCO running with random elimination of cases.

The results also indicate that the combination of metarules 2 and 3 yield the best results. For the foundational cases, MR 2 & 3 led to approximately a 26% improvement in exact matching and a 16% improvement in inexact matching over SIROCCO running with no metarules. For the trial cases, the improvement was 12% and 17%. Since the test of the trial cases had a bigger case base, one would expect the coverage to be greater and thus the elimination of fewer target cases. In fact, notice that while MR 2 & 3 eliminated approximately 2/3 of the foundational cases, only 1/2 of the targets were eliminated in the trial cases version of this test.

Perhaps not surprisingly, adding metarule 1 to metarules 2 and 3 produces little effect (see the first two rows of the tables in both Figures 7 and 8). The latter two rules take into account a deeper level of knowledge, so adding rule 1 to the mix does not help SIROCCO better identify cases it cannot properly handle. Metarule 1 used alone, however, does lead to a small improvement over the use of no metarules.

The contrasting F-Measure results between the remaining and eliminated cases for every test indicate that SIROCCO is making more good choices than bad in deciding whether to eliminate cases. In particular, for every test the mean F-Measures of remaining cases are higher than the corresponding F-Measures of the eliminated cases. On the other hand, the numbers indicate a trade-off. Since the F-Measures of the remaining cases are not dramatically higher than the corresponding F-Measures of the eliminated cases, it is clear that SIROCCO sometimes makes the mistake of eliminating a target case for which it could have provided good suggestions.

Another trade-off is apparent. The versions of SIROCCO that achieve higher F-Measures (e.g., MR 2 & 3, MR 1, 2, & 3, MR in both tables) eliminate the most cases. Arguably, this is a deleterious side effect. If the program can provide suggestions for only 47% of the problems it is provided, as is the case with MR 1, 2, & 3 in the trial test, users may be disinclined to use the program. At this time, however, we do not know that this figure is unreasonable given our case base, whose coverage may well be less than we imagined. Since SIROCCO's Mean F-Measures on the eliminated cases for this test are not dramatically below the remaining cases, it appears that SIROCCO could have correctly processed at least some of the eliminated targets. Thus, in some instances, metarule 3 by itself or the "2 of 3" metarules approach might be preferred, since both eliminate far fewer cases yet still achieve better scores than SIROCCO without metarules. Yet these approaches also appear to eliminate some good targets.

Conclusion

This work addresses the question of how a CBR system can know the limitations of its expertise. Having found the known source cases most relevant to a target problem, how can a system assess whether those cases are similar enough to the target to

warrant venturing advice? We have devised, implemented and evaluated three metarules to help the SIROCCO program make this assessment. The metarules leverage strategic knowledge of how the retrieval algorithm operates and of what successful retrievals look like. Our evaluation confirms that the rules make a positive contribution to SIROCCO's retrieval ability, subject to some tradeoffs.

Our approach addresses some of the same concerns as the use of competent case bases [Smyth and Keane, 1995; Smyth and McKenna, 1999] but is more appropriate for less well-structured domains like ethics, which require the kind of case representation and retrieval mechanisms in SIROCCO. The retrieval algorithm is multi-staged and based on mapping complex data structures, rather than nearest neighbor matching and attribute-value pairs. The cases provide explanations that cite other cases and general rules. Also, SIROCCO doesn't rely on a single best match. Instead, it uses a set of similar cases to help it produce a solution. It benefits from having redundant (i.e., similar) cases in its case base in a domain in which it is not easy to define what a "correct" solution is.

References

[Allen 1983] James F. Allen. Maintaining Knowledge about Temporal Intervals. In the *Communications of the ACM* 26(11), 832-843.

[Ashley 1990] K. D. Ashley. *Modeling Legal Argument: Reasoning with Cases and Hypotheticals*. Cambridge: MIT Press.

[Branting 1991] L. K. Branting. Building Explanations from Rules and Structured Cases. *Int'l Journal of Man-Machine Studies*, 34 (6): 797-837.

[Davis 1982] R. Davis. Expert Systems: Where Are We? And Where Do We Go From Here? *AI Magazine* 3, no. 2:3-22.

[Davis and Buchanan 1985] R. Davis and B. Buchanan. "Meta-Level Knowledge," chapter 28 in B. Buchanan and E. Shortliffe (ed.)*Rule-Based Expert Systems*, pp. 507-530. Addison-Wesley: Reading, MA.

[Forbus *et al*. 1994] K. D. Forbus, D. Gentner, and K. Law. MAC/FAC: A Model of Similarity-based Retrieval. *Cognitive Science* 19, 141-205, Norwood, NJ: Ablex Publ.

[Harris *et al*. 1999] C. E. Harris, M. S. Pritchard, and M. J. Rabins. *Engineering Ethics: Concepts and Cases*. Belmont, CA: Wadsworth, 2nd edition.

[Koomen 1989] J. A. G. M. Koomen. *The TIMELOGIC Temporal Reasoning System*. Tech. Report 231, Computer Science Dept., University of Rochester, NY.

[Lenat *et al*. 1983] D. Lenat, R. Davis, J. Doyle, M. Genesereth, I. Goldstein, and H. Schrobe. "Reasoning about Reasoning," chapter 7 in F. Hayes-Roth, *et al*. ed. *Building Expert Systems*, pp. 219-239. Addison-Wesley: Reading, MA.

[Lewis *et al*. 1996] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training Algorithms for Linear Text Classifiers. In: *Proceedings of the 19th International ACM-SIGIR Conf. on Res. and Dev. in Information Retrieval*. Zurich.

[McLaren 1999] B. M. McLaren. *Assessing the Relevance of Cases and Principles Using Operationalization Techniques*. Ph.D. Dissertation, University of Pittsburgh, PA.

[McLaren and Ashley 1999] B. M. McLaren and K. D. Ashley. Case Representation, Acquisition, and Retrieval in SIROCCO. In: *Proc. of the 3rd International Conference on Case-Based Reasoning*, 248-262.

[McLaren and Ashley 2000] B. M. McLaren and K. D. Ashley. Assessing Relevance With Extensionally Defined Principles and Cases. In: *Proc. of the 17th National Conference on Artificial Intelligence*, 316-322.

[Rissland *et al*. 1996] E. L. Rissland, D. B. Skalak, and M. T. Friedman. BankXX: Supporting Legal Arguments through Heuristic Retrieval. *AI and Law* Volume 4. Kluwer.

[Smyth and Keane 1995] B. Smyth and M. T. Keane. Remembering to Forget: A Competence Preserving Deletion Policy for Case-Based Reasoning Systems. In: *Proc. of the 14th International Conference on Artificial Intelligence*, 377-382.

[Smyth and McKenna 1999] B. Smyth and E. McKenna. Building Compact Competent Case-Bases. In: *Proc. of the 3rd International Conference on Case-Based Reasoning*, 329-342.