Case Representation, Acquisition, and Retrieval in SIROCCO

Bruce McLaren and Kevin Ashley

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

Abstract. As part of our investigation of how abstract principles are operationalized to facilitate their application to specific fact situations, we have begun to develop and experiment with SIROCCO (System for Intelligent Retrieval of Operationalized Cases and CODes), a CBR retrieval and analysis system applied to the domain of engineering ethics. SIROCCO is intended to retrieve decided engineering ethics cases and previously applied ethics codes to assist engineers and students in analyzing new cases. Here we describe a limited but expressive language designed to represent a wide range of ethics cases in SIROCCO, a world-wide web tool developed to perform case acquisition and support a measure of consistency in representation, and an experiment to validate the initial phase of SIROCCO's retrieval algorithm and test its sensitivity to small variations in case description.

Submitted to and accepted for the International Conference on Case-Based Reasoning, Munich, Germany, 1999

Introduction

Developing methods for representing cases and problems is still a serious challenge for CBR. On the one hand, a case representation must be expressive enough for users to accurately describe a problem or case. On the other hand, CBR systems must reason with cases in a computationally tractable fashion. We have been directly confronted with this problem in our attempts to represent cases in engineering ethics.

Professional engineers face a wide range of factual scenarios that raise ethical issues. The National Society of Professional Engineer's Board of Ethical Review (NSPE BER) has analyzed and published over 400 ethics cases. A review of the titles of just some of the 64 cases we have already represented suggests the enormous scope of the scenarios: Gifts, Responsibility for Public Safety, Political Contributions, Supplanting Another Engineer, Plagiarism, Criticism of Another Engineer, Engineer's

Disclosure of Potential Conflict Of Interest, Declining Employment After Acceptance, and Misrepresentation of Firm's Staff.

In addition, the ethics code provisions that the NSPE BER employ to analyze the cases and rationalize its recommendations are very abstract. The NSPE code of ethics comprises 74 provisions involving issues such as public safety, conflicts of interest, confidentiality, and more. Most of these provide only very general guidance. Often, code provisions provide guidance which conflicts with that of other provisions.

Given the wide range of specific scenarios and code provisions, it is interesting to observe how principles, typically too abstract to apply deductively, are nevertheless applied systematically to the scenarios. Our study of the NSPE BER cases revealed that the Board employed a variety of *operationalization techniques* to bridge the gap between abstract principles and specific fact situations (McLaren and Ashley, 1998). (Mostow, 1983) first introduced the notion of operationalization in the comparatively well-defined domain of card playing. However, the NSPE BER's operationalization techniques are applied in a far less structured and more complex problem domain.

Our goal is to make these operationalization techniques explicit in a computational model and leverage them for the retrieval of past cases and analysis of new ethical problems. We have been developing SIROCCO (System for Intelligent Retrieval of Operationalized Cases and CODes), a computational model intended to retrieve decided cases and previously applied principles in order to frame analyses of new engineering ethics cases. SIROCCO is not intended to reach conclusions for the new cases but, rather, to identify relevant information for the analysis of the cases. Ultimately, our goal is to deliver SIROCCO to engineers and engineering students as a tool for improving access to an on-line resource of ethics experience. We also intend to incorporate SIROCCO into an intelligent tutoring environment for engineering students.

We are not the first to address how cases can be used to enrich the meaning of abstract rules. To some extent, researchers in case-based legal reasoning all must address this issue. (Branting, 1994), for instance, sketches a computational model to bridge the gap between legal theories (similar in some respects to principles) and specific case facts focusing on how to determine a precedent's controlling effect. CATO (Aleven, 1997) employs a Factor Hierarchy that relates specific factors to more abstract factors and ultimately to legal issues. BankXX (Rissland et al, 1996) searches a legal network including legal theories for information benefiting a side in a dispute.

In our own earlier work on ethics, we attempted to model components of case-based or casuistic ethical argument (Ashley and McLaren, 1995). Casuistry is a form of ethical reasoning in which decisions are made by comparing a problem to paradigmatic case examples of high level principles (Jonsen and Toulmin, 1988). As compared to legal reasoning, the domain of engineering ethics appears to involve a less well-defined and explicit model of argumentation. In addition, ethical problems are not constrained to only two solutions (e.g., plaintiff won or lost).

A limitation of our earlier work, however, was its impoverished scheme for representing cases. We represented cases at a more abstract, or issue, level. In the current work, we wanted to let case enterers describe, in a somewhat specific manner, what actually occurred in a problem scenario and when. None of the case

representations in the above work supported the representation of a range of cases as wide as those presented in our domain nor dealt with as wide a range of abstract rules. None enables users to specify event time ordering, information which often appears to be important in analyzing moral obligations. Nor has the AI & Law research focused on operationalization of abstract rules, as we have.

Although natural language is the ideal medium for describing a scenario, especially in domains like ethics and law where cases typically are communicated textually, CBR systems cannot yet process complex textual case descriptions. Recent work in textual CBR approaches the problem through improved methods of processing case texts. To the extent that the work has focused on interpreting or adapting complex case texts, it is promising but still limited (Daniels and Rissland, 1997; Bruninghaus and Ashley, 1997). As a result, there is still a need for developing alternative means of representing cases. Indeed, even textual CBR methods often assume that an underlying representation has been developed and will ultimately support case-based reasoning.

Like others before us (e.g., Branting 1990), we have opted for defining a limited, yet expressive, language for representing a case. As with any limited language, there are trade-offs: Do the limitations constrain case enterer's descriptions enough so that a program can identify similarity between cases? On the other hand, do the limitations constrain users so much that they can no longer adequately describe what happened? For instance, while GREBE's relational representation allowed users to describe scenarios flexibly, variations in the way users described similar scenarios threatened to foil its structure mapping algorithm. Our use of a case acquisition tool supports representational consistency (by, for instance, presenting case examples, guidance, and definitions), and our language allows us to address a wider range of cases than, for example, GREBE's workman's compensation domain.

Here we report on our development of a language for representing ethics case scenarios. The language supports users in describing case events and their time ordering. The web-based case acquisition tool has various features to help ensure that different users describe similar cases and problems consistently enough for SIROCCO to match similar cases. In addition, SIROCCO employs a two-stage retrieval algorithm intended to match similar cases flexibly enough despite inevitable small variations in the way cases are described. While our two-stage approach is based on that of several researchers in analogy (e.g., Thagard et al, 1990; Forbus et al, 1994), we focus more specifically on coverage of time-dependent scenarios and leveraging of goal-specific knowledge. To date, we have implemented the first stage and report the results of an experiment designed to evaluate it.

Case Acquisition and Representation in SIROCCO

As noted above, in some ethics cases multiple principles apply with conflicting results. In these cases, the BER needs to determine not only whether and what code principles apply and what conclusions follow from the applicable principles, but also which principles are paramount in the given circumstances. For instance, Case 92-6, figure 1, pits an obligation to one's client against an obligation to public safety.

As previously discussed, the NSPE board employs operationalization techniques to resolve such ethical dilemmas. For instance, in Case 92-6 the board employed a technique we call "Define Code Superiority" to determine that codes related to Engineer B's obligation to the public override codes related to his obligation to the client. In this circumstance, the board decided that Engineer B should have been more forthright and reported the potential hazard of the drums to the client or appropriate authorities.

> Technician A is a field technician employed by a consulting environmental engineering firm. At the direction of his supervisor Engineer B, Technician A samples the contents of drums located on the property of a client. Based on Technician A's past experience, it is his opinion that analysis of the sample would most likely determine that the drum contents would be classified as hazardous waste. If the material is hazardous waste, Technician A knows that certain steps would legally have to be taken to transport and properly dispose of the drum including notifying the proper federal and state authorities.
>
> Technician A asks his supervisor Engineer B what to do with the samples. Engineer B tells Technician A only to document the existence of the samples. Technician A is then told by Engineer B that since the client does other business with the firm, Engineer B will tell the client where the drums are located but do nothing else. Thereafter, Engineer B informs the client of the presence of drums containing "questionable material" and suggests that they be removed. The client contacts another firm and has the material removed. (NSPE, 1958-1997)

Fig. 1. Facts of Case 92-6

We have devised a web site (www.pitt.edu/~bmclaren/ethics) to facilitate users in transcribing cases like this into the Ethics Transcription Language (ETL), a standard format that SIROCCO can process. The web site contains a Participant's Guide with instructions on how to transcribe ethics cases into ETL. It has a Reference Shelf including a standard vocabulary and an example set of 47 transcribed cases.

ETL's standard vocabulary comprises: (1) *Actor & Object Types*, a list of the types of actors and objects which may appear in the engineering ethics scenarios, (2) *Fact Primitives*, a list of the actions and events in which the actors and objects may participate, and (3) *Time Qualifiers,* a list of temporal relations which specify how the actions and events relate to each other in time. Currently, ETL has 70 Actor & Object Types, 190 Fact Primitives, and 11 Time Qualifiers.

In ETL a case is described as an ordered list (i.e., the Fact Chronology) of short sentences. Each is a Fact and satisfies the grammar of a <Fact> as shown in figure 2.

In essence, each Fact Phrase's Fact Primitive is a verb phrase that indicates a specific action or event involving actors, objects, or similarly constituted Fact-Phrases. It is treated, in effect, like a function with up to three arguments (Fact-Primitive arg1 [arg2] [arg3]), where arg1 is the Actor-Or-Object serving as the subject of the verb phrase. In the Fact Chronologies, human case enterers put arg1 before the Fact Primitive as they would the subject of a verb. Each Fact is listed in a table in approximate chronological order as indicated by its Fact-#. Time Qualifiers specify more specific information about the chronological ordering.

While it is by no means easy or quick to transcribe a new case into ETL, the web site offers some amenities to ease the task. First, the Participant's Guide offers a step-by-step tutorial. It instructs the case enterer on how to (1) identify the actors (e.g. engineers, client firms) and objects (e.g., test samples) involved in each scenario, (2) transcribe the scenario into a set of chronological facts, and (3) identify the questioned facts, and the actor or actors whose ethical behavior is questioned. Second, the

Reference Shelf makes it easy to browse through and select from the lists of possible Actor & Object Types, Fact Primitives and Time Qualifiers. To make the former two lists easier to search, they are organized hierarchically by categories. Third, each term in the above lists comes complete with helpful information (e.g., a description of usage, cross references to other related terms, standard variations in form such as inverse, plural, negative) and hyper-linked examples of its use in representing other cases. Fourth, when the transcribed case is submitted, a computer program scans the ETL transcript, identifying any errors in structure and syntax and translating the transcript into the internal representation used by SIROCCO.

<Fact> :=	<Fact-#> <Fact-Phrase> [(Questioned Fact <X>)] <Time-Qualifier> [,<Time-Qualifier>, ...]

<Fact-Phrase> :=	<Fact-Primitive> [<Fact-Modifier>] <Actor-Or-Object>
	  [<Actor-Or-Object> | (<Fact-Phrase>)]
	  [<Actor-Or-Object> | (<Fact-Phrase>)]

<Fact-#> :=	<Positive-Integer>
<Fact-Primitive> :=	*An instance of a Fact-Primitive*
<Actor-Or-Object> :=	*An instance of an Actor or an Object*
<Fact-Modifier> :=	partially | substantially | limited | extensive
<Time-Qualifier> :=	Pre-existing fact | After the start of <Fact-#> [, <Fact-#>, ...] | Starts at the same time as <Fact-#> [, <Fact-#>, ...] | <Time-Period> after the start of <Fact-#> [, <Fact-#>, ...] | After the conclusion of <Fact-#> [, <Fact-#>, ...] | Immediately after the conclusion of <Fact-#> [, <Fact-#>, ...] | <Time-Period> after the conclusion of <Fact-#> [, <Fact-#>, ...] | Ends <Fact-#> [, <Fact-#>, ...] | Occurs during <Fact-#> [, <Fact-#>, ...] | Occurs as part of <Fact-#> [, <Fact-#>, ...] | Occurs concurrently with <Fact-#> [, <Fact-#>, ...] |
<Time -Period> :=	<Y> Days | <Y> Weeks | <Y> Months | <Y> Years
<X> :=	Empty | <Positive-Integer>
<Y> :=	Many | Several | <Positive-Integer>
<Positive-Integer> :=	1 ... N

Key: | = Alternative; [ ] = Optional; < > = Grammar element
Regular font indicates literal placement of language (e.g., "Pre-existing fact ")
Italicized font indicates a general description (i.e., "*An instance of a Fact-Primitive* ")

Fig. 2. The Grammar for the Ethics Transcription Language (ETL)

Submitting a new problem situation to SIROCCO involves the steps described above. Submitting a case to the case base, complete with outcome and analysis, requires an additional step. As led by the Participant's Guide, the case enterer notes the board's conclusions as well as the codes and cases the board cites to justify their conclusions. While the web site supports this task, we will not further describe it here. Figure 3 shows a Fact Chronology for Case 92-6. The table of actors and objects is shown in figure 4. The case enterer has designated Facts 11 and 12 as the Questioned Facts; these facts correspond most closely to the questions stated by the board.

The Fact Primitives are categorized into three types to ensure that Time Qualifiers are used consistently: Events, States, and Terminating Events. Events have relatively short duration, States have relatively long duration, and Terminating Events are special events that typically end a State. In selecting Fact Primitives, Case Enterers are instructed to take the primitive's type into consideration. For example, <is employed by> in Fact 3 in figure 3 is a State primitive. It represents a relatively long time period during which some of the other events of this case occur. If the case facts had focused on the events of being offered and accepting employment, then certain Event Fact primitives would have been more appropriate (e.g., <is offered employment by>, <accepts an offer of employment from>).

1. Client Y **<owns facility>** Client Y Site.	Pre-existing fact
2. Client Y **<hires the services of>** Consulting Environmental Firm X.	After the start of 1
3. Technician A **<is employed by>** Consulting Environmental Firm X.	Pre-existing fact
4. Technician A **<has supervisor>** Engineer B.	Occurs during 3
5. Technician A **<collects test samples>** Collected Samples X **<from>** Client Y Site.	Occurs during 2, 4
6. Technician A **<believes>** (Collected Samples X **<may be hazardous material>**).	After the conclusion of 5
7. Technician A **<knows>** (Client Y Site **<may be hazardous to safety>**).	After the conclusion of 5, Occurs concurrently with 6
8. Technician A **<knows>** (Government Authority **<should be informed about the hazard or potential hazard>**).	Occurs concurrently with 7
9. Technician A **<informs>** Engineer B **<that>** (Technician A **<believes>** (Collected Samples X **<may be hazardous material>**)).	After the conclusion of 6
10. Engineer B **<instructs>** Technician A **<to>** (Technician A **<records the existence of>** Collected Samples X.)	After the conclusion of 9
11. Engineer B **<provides limited information to>** Client Y **<regarding>** Client Y Site. *[Questioned Fact 1]*	After the conclusion of 10
12. Engineer B **<does not inform>** Client Y **<that>** (Technician A **<believes>** Collected Samples **<may be hazardous material>**). *[Questioned Fact 2]*	Occurs as part of 11
13. Client Y **<hires the services of>** Engineering Firm Y.	After the conclusion of 11
14. Engineering Firm Y **<removes material from>** Client Y Site.	Occurs during 13

Fig. 3. Fact Chronology of Case 92-6

1. Technician A --> **Engineering Technician.**
2. Consulting Environmental Firm X --> **Engineering Firm.**
3. Engineer B --> **Engineering Manager.**
4. Client Y --> **Client Firm.**
5. Client Y Site --> **Facility or Site.**
6. Collected Samples --> **Test Samples.**
7. Government Authority --> **Government Authority.**
8. Engineering Firm Y --> **Engineering Firm.**

Fig. 4. Actors and Objects in Case 92-6

After the case enterer has identified the Facts and marked the Questioned Facts, he or she (she henceforth) assigns Time Qualifiers to clarify the chronological

relationships among Facts. Each Time Qualifier has information to guide the choice (e.g., its intended use, what one needs to know to apply it, links to other possible Qualifiers and to examples from other cases.) Each fact in the chronology must have at least one Time Qualifier. The Fact Chronology of Case 92-6, figure 3, shows a number of examples of Time Qualifiers. The most critical temporal relationships are that Facts 11 and 12, in which Engineer B fails to fully inform Client Y of the hazardous materials, occur after Fact 9, in which Technician A informs Engineer B of the existence of the materials. The Time Qualifiers "After the conclusion of" and "Occurs as part of" in Facts 11 and 12 capture this temporal information.

The goal is for the case enterer to represent the important events in the case as accurately as possible, given the limited set of Fact Primitives, Actors, and Objects. The case enterer is asked to do her best despite the limitations. Occasionally, case enterers identify important missing Fact Primitives or Actor or Object Types. They apprise us and we add the new terms into the vocabulary as appropriate. However, at this stage, the size of the vocabulary seems to have leveled off.

The Fact Chronology of figure 3 is only one interpretation of the facts of Case 92-6. For a variety of reasons, different case enterers may produce different interpretations. Despite the guidance provided in the Participant's Guide, a case enterer must still make judgments such as: (1) Deciding which facts are relevant. For instance, Facts 13 and 14 in figure 3 could have been considered irrelevant and deleted from the Fact Chronology. (2) Adding facts implied but not explicitly stated in the text. (3) Deciding whether a sentence should be multiple facts. The optional arguments of a Fact Primitive may be filled by other <Fact-Phrase>s as in Facts 6-10 and 12 in figure 3. This flexibility may lead to alternative formulations. (4) Selecting terms. The Fact Primitives contain a number of synonyms and related terms.

It is an empirical question whether ETL is expressive enough, and the similarity metrics flexible enough, to support SIROCCO in effective case retrieval despite the limitations discussed above. We provide empirical evidence in this paper that at least Stage 1 of SIROCCO's retrieval algorithm works reasonably well[1].

SIROCCO situates all primitives within an abstraction hierarchy known as the *Fact Hierarchy*. For instance, the primitives <hires the services of> (steps 2 and 13 of figure 3) and <employs> (step 3 of figure 3) are both sub-types of the more abstract fact <Work as an Employed or Contract Professional Engineer>. The Fact Hierarchy, developed through an analysis of the NSPE corpus of cases, is a characterization and abstraction of the most important actions and events that typically occur in engineering ethics scenarios. The Fact Hierarchy is an important component of SIROCCO's retrieval algorithm. Cases potentially may be retrieved and matched based on similarity at higher levels of the hierarchy. This will help with the problem of multiple interpretations of the facts, discussed above.

The provisions of the NSPE code of ethics are also cast in an abstraction hierarchy, the *Code Hierarchy*. For instance, Code I.1. ("Engineers ... shall: Hold paramount the

---

[1] By the way, we do not consider the current web site the last word in supporting case entry. With some additional Java or Java Script programming, the web site could automate more of the case entry processes. But the current web site does make case entry feasible via the WWW and at least six case enterers across the U.S. have used it to add 35 cases to the case base.

safety, health, and welfare of the public in the performance of their professional duties."), highly relevant to Case 92-6, is one of 8 codes in the code abstraction group "Duty to the Public." The Code Hierarchy is adapted from a subject reference list found in (NSPE, 1958-1997) and is an important component of the benchmark function used in the experiment discussed in this paper.

SIROCCO's Retrieval Algorithm

Given its case base of engineering ethics dilemmas transcribed into ETL, SIROCCO:
1. Accepts a new fact situation, also transcribed into ETL;
2. Retrieves and matches existing cases using a two-stage algorithm; and
3. Frames an analysis of the new fact situation using operationalizations of past cases and codes.

We have fully implemented a version of ETL and SIROCCO's Stage 1 retrieval algorithm. Stage 1 is a fast and efficient, but somewhat coarse, retrieval based on matching Fact Primitives. Stage 2 is a more expensive structural mapping that focuses on the most critical facts, on the chronology of facts, and on the types of actors and objects in the scenario. We have begun implementing Stage 2 and have designed, but not yet implemented, the analysis portion of the program. In this paper we report on an evaluation of SIROCCO's Stage 1 algorithm and, thus, focus our description on that aspect of the program. For a more complete description of the phases and ultimate output of SIROCCO see (McLaren and Ashley, 1998).

Stage 1 retrieves a preliminary set of matching cases using *content vectors* (Forbus, et al. 1994), data structures associated with each case that summarize the Fact Chronologies. A content vector represents a count of each Fact Primitive that appears in a chronology. For instance, the content vector of Case 92-6, the example discussed in the previous section, associates the value 2 with Hires-the-Services-Of because the primitive <hires the services of> appears twice in the Fact Chronology (see steps 2 and 13 in figure 3). Figure 5 depicts the content vector of 92-6, as well as the content vector of Case 89-7, a case that is relevant to 92-6[2].

To perform retrieval, SIROCCO takes the content vector of the input case and computes the dot product of that vector against all of the other cases' content vectors. Since all content vectors are pre-computed and stored in a hash table, computation is fast. Figure 5 shows the dot product between Cases 92-6 and 89-7. 89-7 is highly relevant to 92-6, and the dot product calculation bears this out. With a dot product of 10, 89-7 is ranked as the 4th highest case in our corpus of 63 cases (excluding 92-6).

From an architectural viewpoint, given the two-stage retrieval, it is more important for Stage 1 to retrieve as many relevant cases as possible than it is to avoid retrieving irrelevant cases. Stage 2 will not be able to seek additional relevant cases beyond those Stage 1 finds. Stage 2, on the other hand, will be capable of making finer distinctions between cases, such as comparing the temporal relations between steps. Of course, too

---

[2] Note that content vectors are also defined for higher level facts in the fact hierarchy for each case. This facilitates computation of similarity at more abstract fact levels.

many irrelevant cases retrieved by Stage 1 would flood the computationally more expensive Stage 2. The right number of cases to retrieve in Stage 1 and pass to Stage 2 is an empirical question; we currently estimate 10.

Content Vector of Case 92-6:

((Owns-Facility . 1) (Hires-the-Services-Of . 2) (Employs . 1) (Supervises . 1) (Collects-Test-Samples-From . 1) (Believes . 1) (Knows . 2) (Informs-That . 2) (Instructs-To . 1) (Provides-Limited-Information-To-Regarding . 1) (Removes-Materials-From . 1)))

Content Vector of Case 89-7:

((Hires-the-Services-Of. 1) (May-Be-Hazardous-To-Safety . 1) (Instructs-To . 1) (Inspects . 1) (Discovers-That . 1) (Knows . 1) (Informs-That . 2) (Writes Paper/Article . 1) (Provides-Limited-Information-To-Regarding . 1)))

Dot Product of Cases 92-6 and 89-7:

+ 2	[2 (92-6) * 1 (89-7)]	Hires-the-Services-Of
+ 2	[2 (92-6) * 1 (89-7)]	Knows
+ 4	[2 (92-6) * 2 (89-7)]	Informs-That
+ 1	[1 (92-6) * 1 (89-7)]	Instructs-To
+ 1	[1 (92-6) * 1 (89-7)]	Provides-Limited-Information-To-Regarding
----		
10		

Fig. 5. Content Vectors of Cases 92-6 and 89-7 and the Resulting Dot Product

We also expect *Code Instantiations* to enhance the effectiveness of both retrieval stages, but we have not evaluated their contribution here. A Code Instantiation relates a questioned fact, certain critical facts, and the order of those facts to a code; the case enterers identify them as they transcribe cases. Code Instantiations will help Stage 1 by offsetting the current default assumption, implemented in the dot product computation, that all Fact Primitives are equal. Code Instantiations will increase the dot product of those cases matching the most critical facts. In Stage 2, they will focus the structural mapping routine on a smaller number of critical facts.

An Evaluation of SIROCCO's Stage 1 Retrieval Algorithm

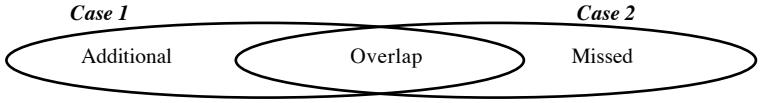
We undertook a formative evaluation to assess how well SIROCCO's Stage 1 retrieval algorithm works. Because of the architectural assumption above, the experiment focused on Stage 1's capability to retrieve relevant cases, as opposed to its capability of avoiding the retrieval of irrelevant cases. In addition, we compared SIROCCO's exact match retrieval algorithm with versions that match at higher levels of the Fact Hierarchy. Intuitively we expect that relevant cases might be missed if only exact fact matches were attempted.

The Benchmark Function: Citation Overlap

To evaluate SIROCCO we first needed a benchmark to objectively compute the similarity between cases. Because to our knowledge no such function exists, either in

a comparable CBR system or within the domain of engineering ethics, it was necessary to define our own. From an analysis of our corpus, it was clear that the most objective and feasible similarity measure is *citation overlap*. When two cases cite the same code, or codes from the same category (i.e., *code citation overlap*), there is a strong indication that the cases are relevant to one another. Likewise, when one case directly cites another, or when two cases share a citation to a third case (i.e., *case citation overlap*), there is strong indication of relevance. Citation overlap has also been used as a metric in other work (Cheng, Holsapple, and Lee, 1996).

Our benchmark function had to account for overlaps of both code and case citations and their relative contributions to relevance. For code citation overlap, we used the information retrieval metrics *recall* and *precision*, combined by the weighted F-measure function (Lewis et al, 1996; van Rijsbergen, 1979, pp. 173-176) (See figure 6).



Case 1 = The code citations in Case 1
Case 2 = The code citations in Case 2
Overlap = The code citations shared by both Case 1 and Case 2
Additional = The extra code citations in Case 1 but not in Case 2
Missed = The set of code citations lacking in Case 1 but found in Case 2

Recall (R) = Overlap / (Overlap + Missed) **Precision (P)** = Overlap / (Overlap + Additional)

Code Citation Overlap[3] (Case1, Case2) = $\frac{(\beta^2+1)PR}{\beta^2 P+R}$ (F-measure)

Fig. 6. Quantitative Measurement of Code Citation Overlap

Two codes can "overlap" abstractly by sharing a common ancestor in the Code Hierarchy (below the root node). Overlap is calculated by taking 1 divided by the number of levels between the codes and the level at which they share an ancestor.

Intuitively, the F-measure is a good choice for the code citation overlap because it measures the degree with which "issues" are shared between cases. Shared codes represent, roughly speaking, common issues between cases, while unshared codes represent issues relevant to one case but not the other. The more issues two cases share, proportionally speaking, the more likely they are to be relevant to one another. The F-measure computes this through a form of intersection, also imposing a penalty for lack of intersection (i.e., missing relevant codes, including irrelevant codes).

With respect to case citation overlap, we applied a shortest path algorithm between the cases. Viewing the case base as a network, with cases as nodes and case citations as edges, the shortest path between Case 1 and Case 2 is defined as the minimum number of edges between Case 1 to Case 2. Case citation overlap is computed as:

---

[3] ß < 1.0 gives greater weight to precision; ß = 1.0 gives equal weight to recall and precision; ß > 1.0 gives greater weight to recall. The code citation overlap ranges from 0.0 to 1.0.

$$\textbf{Case Citation Overlap}\ (\text{Case1, Case2}) = \frac{1}{ShortestPath(Case1, Case2)}\ \text{or 0, if there is no path}$$

Thus, for instance, a direct citation between two cases results in case citation overlap = 1.0 (1/1) and a case citation shared by two cases (i.e., a case node exists between the two cases) results in case citation overlap = 0.5 (1/2).[4]

The final step in computing citation overlap is combining the code citation overlap and the case citation overlap into a single measure. We implemented citation overlap as the weighted sum of the constituent overlap functions:

$$\textbf{Citation Overlap}\ (\text{Case1, Case2}) = \frac{(\alpha - 1)CodeCitationOverlap(Case1, Case2) + CaseCitationOverlap(Case1, Case2)}{\alpha}$$

To establish the benchmark it was necessary to fix the α and ß values of the citation overlap. We set ß = 2.0 to favor the recall component of the F-measure. Informal analysis of the corpus indicated that overlapping code citations are much stronger indicators of relevance than non-overlapping codes are of irrelevance. Although we have no rigorous empirical evidence, we calculated that 77.2% of the direct case citations in our corpus, those with the most obvious relevance to one another, also share at least one code citation. For the measures depicted in figure 6, ß = 2.0 establishes that Overlaps > Missed >> Additional in terms of importance.

We fixed α = 3.0, favoring the code citation overlap over the case citation overlap by a 2-1 ratio in our weighted function. This reflects the fact that while the presence of case citation overlap is a strong indicator of relevance, the absence of case citation overlap is not a strong indicator of irrelevance.

The Experimental Procedure

Our experimental procedure is summarized in figure 7. For each Case X in our corpus of 64 cases we first calculated the citation overlap against all other cases (excluding the case itself). We then sorted the resultant list of citation overlap values, keeping only the top N cases or those above a specified threshold, whichever resulted in a shorter list. We then applied SIROCCO's Stage 1 retrieval algorithm to Case X, retrieving the top N cases according to the dot product calculation. Next, we compared SIROCCO's list of retrieved cases with the citation overlap list by computing recall between the two lists. After performing these calculations for all cases in the corpus, we calculated the mean recall value across all cases.

For the experiments reported in this paper, N was set to 10. This value strikes a balance between retrieving too few and too many cases. Because of the relative

---

[4] The F-measure is not appropriate for case citation overlap because direct case citations, while rare, are "overriding" indicators of relevance. That is, if Case X cites Case Y, either directly or indirectly, we can conclude that Cases X and Y are relevant to one another. Whether Cases X and Y share additional case citations is far less important to the relevance assessment. The shortest path algorithm models this overriding importance of the citation path. (Note, however, that the shortest path algorithm completely ignores additional shared citations. On the other hand, this has negligible effect on the overlap calculation, since case citations are relatively sparse in the corpus.)

crudeness of the dot product calculation, we believed that setting N too low would allow too many relevant cases to go unretrieved. On the other hand, setting N too high would result in SIROCCO's Stage 2 algorithm being flooded with too many cases to process.

We ran the experiment varying threshold values from 0.0 to 1.0. Lower threshold values allow lower rated cases, according to the citation overlap metric, to be compared to SIROCCO's retrieval. Higher threshold values, on the other hand, filter out the lower rated cases, comparing SIROCCO's retrieval only to the *best* possible cases. We hypothesize (and hope) that SIROCCO would perform better at higher threshold levels, since in this condition we are more likely to be comparing SIROCCO to the most relevant cases in the case base.

```
Run-SIROCCO-Experiment (N, Citation-Overlap-Threshold)
1.    For Case X of All-Cases
2.        For Case Y of All-Cases (Excluding Case-X)
3.            Citation-Overlap-Results = Apply Citation-Overlap (Case X, Case Y)
4.            Add Citation-Overlap-Results to Citation-Overlap-List
5.        End For
6.        Citation-Overlap-List = Sort (Citation-Overlap-List, N, Citation-Overlap-Threshold)
7.        SIROCCO-Retrieval-List = Apply SIROCCO-Stage1-Retrieval (Case X, N)
8.        If (Citation-Overlap-List is not empty) then
9.            Comparison-Value [X] = Recall (SIROCCO-Retrieval-List, Citation-Overlap-List)
10.   End For
11.   Mean-Recall = Calculate-Mean (Comparison-Value [1 ... Length (All-Cases)])
```

Fig. 7. Pseudo-code Description of SIROCCO's Experimental Procedure

We experimented with four different versions of SIROCCO's Stage 1 retrieval algorithm. The versions varied by level of abstraction at which facts were matched. The different versions were:

- *Primitive:* Exact matches between Fact Primitives.
- *Immediate Group:* Matches between Fact Primitives 1 level up the Fact Hierarchy.
- *Sibling Group:* Matches between Fact Primitives 2 levels up the Fact Hierarchy.
- *Root Group:* Matches between Fact Primitives at the "root group" level, i.e., one level beneath the root of the Fact Hierarchy.

As a baseline, we also compared a randomly selected set of cases against the citation overlap at each threshold value.

We chose recall as the metric to compare SIROCCO's retrieval with the citation overlap because of the architectural assumption discussed earlier. It is more important that Stage 1 retrieve a high percentage of relevant cases than filter out irrelevant cases.

Experimental Results and Analysis

Figure 8 depicts the results of our experiment, comparing SIROCCO's Stage 1 retrieval algorithm (4 different versions) against the citation overlap at threshold levels ranging from 0.0 to 1.0.

Notice, first of all, that Primitive, the exact matching algorithm, performed as well as or better than the other 3 versions of the algorithm, and much better than the random comparison, at every threshold level. In fact, in a paired difference t-test, comparing Primitive's recall to the random recall, we found that Primitive performed significantly better than random at every threshold level using a significance level of 0.001. Further, Primitive performed consistently, if marginally, better than the other versions of the algorithm at higher threshold levels. Notice also that SIROCCO's retrieval gradually, but consistently, improved up to approximately the 0.8 threshold level.

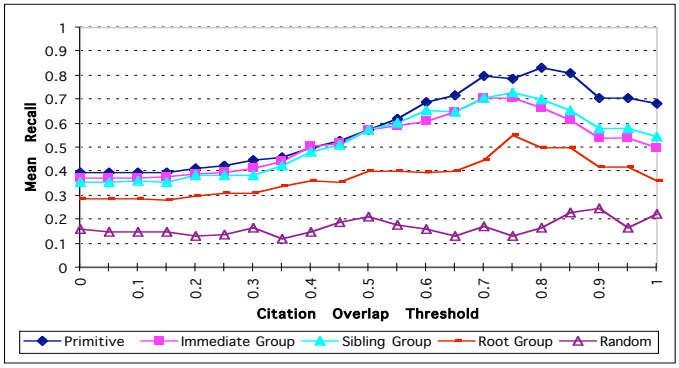


Fig. 8. Results of Comparing SIROCCO's Stage 1 retrieval against the Citation Overlap Benchmark

The results and differences between the various approaches were less impressive at the lower threshold levels. Here, however, SIROCCO's retrieval was compared to less selective sets of citation overlap cases, since the threshold was so low. Many of the cases rated in the top N by the citation overlap in this region are marginally relevant cases. Thus, we did not expect SIROCCO to perform particularly well against the benchmark under this condition. Still, as compared to random selection, the algorithm arguably performed reasonably well even in this region.

How does one assess the relative merits of the different versions of SIROCCO's retrieval algorithm? Clearly, the methods Primitive, Immediate Group, and Sibling Group performed almost identically up to the 0.55 threshold level, at which point the Primitive method performed only slightly better than the other two approaches at the remaining threshold levels. This was not altogether surprising. There are clearly situations in which an exact match is preferred, such as the matching of the most critical facts of two scenarios. On the other hand, there are also situations in which an abstract match may ultimately yield a more relevant case, for instance, in matching

highly similar primitives such as <is employed by> and <is hired to provide services for> between cases which exhibit other similarities.

To study this issue in more detail, we examined how several cases fared using the different retrieval algorithms at the 0.6 threshold level. In particular, we were interested in why certain cases yielded low recall (<= 0.5) using the exact match retrieval method (i.e., Primitive), but then improved to 1.0 using one of the more abstract retrieval methods. Two cases, 70-4 and 89-2, led to a clearer understanding of this behavior. In Case 70-4, the fact chronology is moderately short (7 steps) but, more telling, the 2 most critical facts of the scenario were not matched exactly in *any* other cases. However, moving up the Fact Hierarchy led to matches with abstractly similar, yet specifically different, cases, and recall improved from 0.0 for the Primitive method to 1.0 for the Sibling Group method. Case 89-2 is a similar situation: a chronology of only 7 steps in which 5 key steps were unmatched at the Fact Primitive level. Again, moving up the Fact Hierarchy led to improved matching, this time at the Immediate Group level.

These anecdotal findings suggest that it may be fruitful to heuristically combine the different retrieval methods. For instance, cases yielding low dot product matches at the Fact Primitive level (such as 70-4 and 89-2 did) may be improved by combining the retrieval scores from the Fact Primitive and the Immediate Group levels (and so on). We will experiment with such heuristics as we further develop SIROCCO.

We noted that retrieval may fail when Fact Chronologies were excessively long. For instance, SIROCCO did not retrieve a single overlapping case at threshold = 0.6 for Case 77-11, an unusually long 17-step chronology. Upon examining the detailed matching data, it was clear that 77-11's long chronology led to dot product "swamping" with many cases, undervaluing the few highly critical steps of the chronology. Our plan to extend Stage 1 through the use of Code Instantiations will be advantageous in these situations. Code Instantiations will focus more attention (and value) on the critical steps of a fact situation, rather than treating all steps equally.

Finally, it should be noted that in the experiment cases were dropped from consideration when there were no other cases with citation overlap >= threshold (See step 8 in figure 7). While this was reasonable as part of the experimental procedure, it begs the question: How would SIROCCO perform when there are no reasonably relevant cases to retrieve? Given our intent to represent a wide range of cases, such a situation is not unlikely. We believe, again, that the Code Instantiations will assist the program in performing sufficiently well when retrieval matches are weak. The Code Instantiations will allow the program to find other cases that, while not relevant at the overall case-level, may share relevance at the level of key facts

Conclusions

In this paper, we have described SIROCCO, a case-based retrieval system designed and developed as part of our exploration into how abstract principles are applied to specific fact situations. We have devised a limited but expressive case representation language that models the actors, objects, facts, and chronology of facts in a wide range of

engineering ethics scenarios. A web site acts as the case acquisition tool for the system and supports a measure of consistency in the representation. Finally, we described an experiment in which we tested an initial stage of SIROCCO's retrieval algorithm. Our results indicate that SIROCCO retrieves cases at an acceptable performance level and is not overly sensitive to small variations in case description. We also uncovered ways in which we could extend and improve SIROCCO's retrieval algorithm.

References

Aleven, V. (1997). *Teaching Case-Based Argumentation Through a Model and Examples*. Ph.D. Dissertation, University of Pittsburgh.

Ashley, K. D. and McLaren, B. M. (1995). Reasoning with Reasons in Case-Based Comparisons. In the *Proceedings of the First International Conference on Case-Based Reasoning* (ICCBR-95). Pp, 133-144. Lecture Notes in Artificial Intelligence 1010. Springer Verlag. Heidelberg, Germany.

Branting, L. K. (1990) *Integrating Rules and Precedents for Classification and Explanation: Automating Legal Analysis*. Ph.D. Dissertation. U. Texas at Austin AI Lab. AI 90-146.

Branting, L. K. (1994). A Computational Model of Ratio Decidendi. *Artificial Intelligence and Law* 2: 1-31. Kluwer Academic Publishers. Printed in the Netherlands.

Bruninghaus, S., and Ashley, K. D. (1997) Using Machine Learning to Assign Indices to Legal Cases. In the *Proceedings of the Second International Conference on Case-Based Reasoning* (ICCBR-97). Pp. 303-314. Lecture Notes in Artificial Intelligence 1266. Springer Verlag. Heidelberg, Germany.

Cheng, C. H., Holsapple, C. W. and Lee, A. (1996). Citation-Based Journal Rankings for AI Research: A Business Perspective.. *AI Magazine,* Vol. 17, No. 2.

Daniels, J. and E. Rissland. (1997) Finding Legally Relevant Passages in Case Opinions. In the *Proceedings of the Sixth International Conference on AI and Law* (ICAIL-97). Pp. 39-46. ACM Press: New York.

Forbus, K. D., Gentner, D. and Law, K. (1994). MAC/FAC: A Model of Similarity-based Retrieval. *Cognitive Science* 19, Pp. 141-205.

Jonsen A. R. and Toulmin S. (1988). *The Abuse of Casuistry: A History of Moral Reasoning*. University of CA Press, Berkeley.

Lewis, D. D., Schapire, R. E., Callan, J. P., and Papka, R. (1996). Training Algorithms for Linear Text Classifiers. In the *Proceedings of the 19th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*. Zurich.

McLaren, B. M. and Ashley, K. D. (1998). Exploring the Dialectic Between Abstract Rules and Concrete Facts: Operationalizing Principles and Cases in Engineering Ethics. In the *Proceedings From the Fourth European Workshop on Case-Based Reasoning*. Pp, 37-51. Lecture Notes in Artificial Intelligence 1488. Springer Verlag. Heidelberg, Germany.

Mostow, J. (1983). Machine transformation of advice into a heuristic search procedure. In *Machine Learning*, vol. 1.

NSPE (1958-1997). *Opinions of the Board of Ethical Review,* Vol. I - VII and *NSPE Ethics Reference Guide*. Published by National Society of Professional Engineers, Alexandria, Virginia.

Rissland, E. L., Skalak, D. B., and Friedman, M. T. (1996). BankXX: Supporting Legal Arguments through Heuristic Retrieval. *AI and Law* 4: 1-71. Kluwer, Dordrecht.

Thagard, P., Holyoak, K., Nelson, G., and Gochfeld, (1990). Analog Retrieval by Constraint Satisfaction, *Artificial Intelligence* 46, Pp. 259-310.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths, London, second edition