This is to certify that the

dissertation entitled

CONCEPTUAL RETRIEVAL FROM CASE MEMORY
BASED ON PROBLEM SOLVING ROLES:
A GENERIC TASK ARCHITECTURE
WITH APPLICATION TO
JUSTIFICATORY REASONING IN TAX LAW

presented by

Vernon Eugene Wallingford II

has been accepted towards fulfillment
of the requirements for

Doctor of Philosophy degree in Computer Science

Major professor

Date 9 Nov 92

# CONCEPTUAL RETRIEVAL FROM CASE MEMORY BASED ON PROBLEM SOLVING ROLES: A GENERIC TASK ARCHITECTURE

## WITH APPLICATION TO JUSTIFICATORY REASONING IN TAX LAW

By

Vernon Eugene Wallingford II

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

1992

# ABSTRACT

CONCEPTUAL RETRIEVAL FROM CASE MEMORY
BASED ON PROBLEM SOLVING ROLES:
A GENERIC TASK ARCHITECTURE

WITH APPLICATION TO
JUSTIFICATORY REASONING IN TAX LAW

By

Vernon Eugene Wallingford II

Given an understanding of the domain and the data stored, how can one locate in an immense data base only those records or documents semantically relevant to a particular topic of interest? This question frames the *conceptual retrieval* problem. Rather than trying to solve the conceptual retrieval problem solely in terms of general knowledge about a domain, this research advances the view that knowing the use to which retrieved items will be put provides guidance in developing more useful indexing vocabularies and retrieval methods. This dissertation focuses on conceptual retrieval for the purpose of case-based justification in tax law, in particular for the area of captive insurance taxation. In this context, a theory of conceptual retrieval is presented that elaborates an index vocabulary and organization based on the roles that cases can play in justification.

This dissertation describes three primary products of this research: (1) a methodology for the Functional Representation of justifications, (2) a model of conceptual retrieval —an index vocabulary, an index organization, and a retrieval method — motivated by this methodology for representing justifications, and (3) a conceptual memory of arguments based on this model. This conceptual memory is integrated with a Generic Task architecture for justifying legal classifications. Finally, these concepts are implemented in a knowledge-based system called CRISTA.

This research addresses the conceptual retrieval problem from a task-specific perspective. The result is the identification of an indexing methodology that is closely

related to a particular problem-solving task and a particular case representation. In essence, this work unites a task-specific theory of problem solving with ideas from case-based reasoning about indexing in order to achieve a more complete picture of conceptual memory. To related disciplines, one of the important contributions of this research lies in its description of how one can employ knowledge of a device and its teleology in constructing more effective and efficient case memories.

To my dearest Mary.

If all I ever accomplish
is to be worthy of your precious love,
I will consider myself a successful man.

# ACKNOWLEDGMENTS

I now understand why people always recognize the contributions of so many others to their thesis work. This is a long and grueling process, and I certainly needed a lot of help and support to make it this far. I thank Jon Sticklen, my advisor, for his guidance and intellectual stimulation these last five years. So much of what I learned will never appear in this document. Thanks also to my dissertation committee, Rich Hall, Bill McCarthy, and Tony Wojcik. Each in his own way has left an indelible mark on my professional and personal views. Special thanks are due to Bill, for finally convincing me that this problem was worth attacking. He was right.

I owe so much to my parents, Vernon and Linda Wallingford, my brother Anthony, and my sisters, Nancy and Cathy. They have always believed in me — even when I was not so sure myself. Their love and support often urged to me to try harder, and I thank them for their encouragement.

Finally, I thank my wife, Mary. I can truthfully say that, without her love, encouragement, support, and occasional prodding, this dissertation would never have been completed. She has buoyed my spirit during down times and brought me incomparable joy during up times. She is my partner and best friend.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 The Conceptual Retrieval Problem

The arrival of the Information Age has radically changed how we do — and conceive of doing — most knowledge-intensive tasks. With the widespread availability of digital computers and expansive, inexpensive primary and secondary memory, industries and individuals have accumulated vast stores of raw numeric data, full-text documents, and interpreted knowledge. In many domains, such as molecular biology, the challenge involves searching unrefined data for patterns that will lead to an understanding of the domain itself. These fields require advances in database technology that will support evolving and conflicting data models and the search for meaningful patterns. Other domains, while sharing some of these concerns, face another primary challenge: given an understanding of the domain and the data stored, how can one locate in an immense data base only those records or documents relevant to a particular topic of interest? This question frames the *conceptual retrieval* problem.

Traditional approaches to information retrieval (IR) have relied on syntactic phenomena, employing key-word indices into a data base of full-text or structured-text items. However, such key-word schemes cannot capture the rich semantic structure of domains such as the law or medicine, and as a result efficient use of these tools can be attained only by practitioners experienced in the creation of appropriate key-word queries. In order to provide effective access to voluminous data for a broader array of users, these domains require models of the data that incorporate semantic as well as syntactic

relationships. Users could then query the data base using the vocabulary of the domain and rely on the conceptual model to identify documents relevant to the identified concepts. Such a model will necessarily support a vocabulary that is abstract enough to reflect how users conceptualize the domain but also is supported by a more concrete understanding of domain relationships.

The problem of conceptual information retrieval parallels a human information-processing problem that people solve routinely every day, that of episodic memory retrieval. People continually face situations in which they are reminded of similar past experiences; they can often use these experiences as aids for resolving the new situation. While many such "remindings" involve experiences related only by surface features to the current situation, other remindings reflect deeper conceptual relationships. This latter phenomenon is especially marked in focused, problem-solving circumstances. Individuals highly experienced in certain domains and tasks develop skill in identifying critical conceptual similarities and differences among situations that determine how best to solve new problems. Capturing this kind of knowledge about a domain would greatly enhance the conceptual retrieval of information stored in computers.

### 1.1.1 Conceptual Retrieval for Legal Analysis

Nowhere is the problem of conceptual retrieval more important than in the task of legal analysis. Given a description of a situation and a target classification desired for the situation, one must generate a justification that supports the desired treatment. A justificatory line of reasoning is an essential component of any legal classification. Without substantiation, a legal claim carries no force in a legal system that is adversarial by design.

In a common law tradition, past classifications serve as the principal source of justification for new classifications. Common law thus leads to a proliferation of case records, which are then accumulated and organized for future reference. Specialists in

particular case law domains seem to organize their understanding of cases according to the themes of prevailing lines of reasoning. That is, the use to which they put past cases affects their understanding of the body of case law. Such specialists cannot themselves remember *all* of the cases in a large domain, though, and must then turn to exhaustive catalogs of past cases, whether in the printed literature or on-line computer data bases.

Legal analysis consists of two stages, and case research plays a role in each stage. First, the researcher analyzes the current situation with respect to all sides of the law. The goal of this stage is not to justify a particular classification but rather to consider all possible treatments and lines of precedent. The researcher actively seeks out all past cases that could reasonably bear on the outcome of the case at hand. From the assembled collection of relevant statutes and case law, the researcher outlines potential conclusions regarding the law and the grounds upon which such conclusions might be based. Second, the lawyer evaluates these lines of the reasoning and writes a brief that justifies a particular treatment of the case at hand. The brief will be written to emphasize elements of the most favorable precedents and to downplay elements of the most unfavorable ones. It may also seek to counter some of the arguments anticipated from the opposing counsel.

### 1.1.2 Conceptual Retrieval in Tax Accounting

Legal analysis plays a significant role in the practice of tax accounting. Tax accountants have two primary concerns: the application of tax law in filing tax forms, and the planning of future transactions to minimize tax effects. In unsettled areas of the law, such as the taxation of captive insurance corporations, these tasks are made more difficult by the uncertainty entailed in classifying new situations. Through legal analysis and case research, the tax specialist can evaluate potential treatments of a situation or proposed action. Consideration of past cases and the risk involved in adopting a particular

classification enables the accountant to make the best choices in the interest of his client, within the bounds of the law.

Tax law offers an interesting domain in which to consider legal analysis. Like other areas of the law, tax law evolves as new situations are considered and classified. However, unlike most areas of the common law, taxation has a large body of statute that defines and relates concepts in the domain. Tax law statute often does not rely on common-sense notions of human relationships, as do "natural" law domains such as contracts and torts. Rather, it creates a set of abstract entities and relations for the purpose of directing tax policy. This abstraction makes the domain ideal for initial investigations of the legal analysis task. By considering legal analysis in tax law first, one can focus on the task itself,within the context of the artificial world in which tax law operates. The models of legal analysis developed through study of tax law domains can then be applied to the law as a whole, at which time one will be better equipped to focus on complexities native to more natural legal domains.

## 1.2 Requisites of a Solution

Solving the conceptual retrieval problem will likely require moving beyond key-word approaches to the modeling of semantic structure and task features in particular domains. This type of domain modeling has long been one of the central themes of the Artificial Intelligence (AI) community. Though AI research has often focused on the development of problem-solving systems for tasks such as legal argumentation, some researchers now believe that AI will make its greatest practical impact, at least initially, on issues of information retrieval: "In the short run, AI's principal contribution to society may be to provide intelligent access to our vast data bases of information, in particular, helping us to select and organize information that is relevant." [Ashley 1990, page 6]

Yet such a contribution will likely profit from the understanding of task structures gained through research on problem solving. This research aims to identify vocabularies that are useful for describing and performing different types of problem solving. One of the central advances of problem-solving research is the recognition that task-level terms define particular roles for domain knowledge to play in performing the task. In this way, these vocabularies provide leverage for organizing knowledge in the domain to support a particular problem-solving task. The sheer volume of information to be searched in domains such as the law demands the judicious use of such conceptual vocabularies as a means for organizing computer memory.

Any useful model of conceptual retrieval will include three components: an index vocabulary, an index organization, and an algorithm for retrieval. The index vocabulary specifies the terms in which one formulates queries. In key-word systems, the index vocabulary consists of all words appearing in a stored document. The index organization specifies how terms in the vocabulary relate to one another. Key-word systems rely solely on syntactic relations among words. Finally, the retrieval algorithm specifies how to identify and select relevant items given a particular query. In key-word systems, search is conducted using pre-built inverted files of words that point to their exact locations in documents.

By making explicit the *semantic* relationships among index terms in a given domain, one can tailor a retrieval algorithm that most effectively exploits the structure of domain knowledge. Ideally, this sort of theory will generalize to the class of tasks and domains characterized by the same types of knowledge structures. In order to support such abstraction, the model must provide some mechanism for identifying appropriate index terms in the domain and for identifying appropriate combinations of index organization and search algorithm.

### 1.2.1 The Context of Tax Law

One such mechanism applicable to the task of legal analysis involves understanding justifications and the roles they play in the domain. Justifications serve not only to support particular treatments of tax actions but also to implicitly define terms left unspecified in statute. This latter role indicates that a justification may be relevant to queries about concepts that are refined by it. In this way, justifications offer definitional content to a domain's concepts.

Likewise, justifications play a role in abstracting away detail from more settled issues to allow closer examination of the critical issues at hand. By citing a relevant precedent that establishes a particular classification, the new justification need not include the detailed reasoning that supports the classification; it can focus on another issue that is more problematic in the current case. This role also indicates how an understanding of tax arguments can positively influence the selection of an index vocabulary. To support such screening of detail, the index vocabulary should employ abstraction links among justifications based on their conceptual relationships. In this way, an analysis of justifications in the domain — coupled with an analysis of how domain specialists understand the justifications — can provide significant guidance for organizing and searching cases in a conceptually-organized data base.

Furthermore, knowledge of how specialists solve problems in a domain also guides the development of effective case organization and retrieval schemes. This idea has received considerable attention in the study of case-based reasoning, and the investigation of problem-solving methods in conjunction with case memory offers new ideas for the solution of the conceptual retrieval problem. The goals of a tax law specialist in classifying a situation serve as valuable indicators of when certain past justifications will be of use. Having an understanding of the structure of the problem-solving process itself furnishes even more information about how cases can best be organized for efficient retrieval. Going

beyond design of index vocabularies to this sort of attention to index organization represents a critical step in the realization of effective and efficient conceptual retrieval tools.

## 1.3 Intellectual Influences

This research reflects a variety of influences from a number of related fields, among them knowledge-based systems, information retrieval, case-based reasoning, cognitive psychology, AI in the law, and epistemology. The effects of these influences will appear throughout this volume. However, a smaller number of ideas have had an important impact on both the framing of the research problem and the conceiving of the proposed solution. These pivotal ideas are described briefly below.

### 1.3.1 The Generic Task Approach

Chandrasekaran [1983, 1987, 1990] has advanced a task-specific theory of problem solving founded on the notion of task/method/sub-task analysis of information processing. For a given information processing task, characterized by its inputs and outputs, one identifies different methods available for solving it. Each method is characterized by a set of objects to be manipulated, a set of operators, and knowledge for selecting and applying operators to objects. Some operators are not "primitive," in the sense that they establish sub-tasks to be solved by other means. This leads to a recursive decomposition of tasks that halts upon reaching methods for which all operators are directly applicable. In this sort of analysis, tasks correspond to steps in the solution of a problem, and methods correspond to ways of realizing particular tasks, whether by decomposition or by direct action.

The application of this methodology to a variety of real-world tasks and domains has led to the identification of a set of ubiquitous methods for solving certain tasks that

arise in a variety of different circumstances. Chandrasekaran has termed these task/method pairs *generic tasks*. Generic tasks have proven especially useful as models of problem-solving types because they explicitly delineate (1) the types of domain knowledge required for application of the method and (2) the nature of the control strategy for applying its operators. Examples of generic tasks include *hierarchical classification* [Bylander and Mittal 1986; Sticklen, Chandrasekaran, and Josephson 1987], *routine design* [Brown 1987], and *structured matching* [Bylander, Johnson, and Goel 1991]. These problem-solving types play particular roles in the analysis of tax law argumentation proposed here.

### 1.3.2 Toulmin's Model of Argument

A philosopher of science, Toulmin [1958] questioned the usefulness of traditional work on logic and deductive inference for assessing the sorts of arguments actually made in most scientific fields. Recognizing that most argument did not correspond to the notion of absolute proof, Toulmin abandoned mathematics as the foundation of logic and instead adopted jurisprudence as his model. This move led to his extending the classical syllogism to explicate more accurately the nature of persuasive argument — that is, to specify the various roles that assertions can play in an argument and the relationships among these roles. Toulmin's model greatly enriches the vocabulary available for describing an argument to include such natural terms as data, backing, warrant, qualification, and conclusion. On this view, logic deals not with techniques of inferring but rather with retrospective justification of a claim.

### 1.3.3 Functional Device Understanding

The Functional Representation (FR) of Sembugamoorthy and Chandrasekaran [1986] provides a language for describing devices at multiple levels of abstraction based on the device's known functions or goals. In the FR, devices are decomposed into their

components, whose own functions can then be composed in order to understand the functioning of the composed device. The causal behaviors of each device or sub-device are indexed according to the functions they realize. Since functions of the higher-level device can be expressed in terms of the functions of its components, the FR supports an abstraction of behavioral detail between different levels of the device decomposition. This sort of representation originally aimed at understanding physical devices but has since been extended to the comprehension of "abstract" devices such as computer programs [Allemang 1990] and biological ecosystems [Sticklen and Tufankji 1992].

One can view a justification as an abstract device with the function of supporting some claim given an initial set of assertions. Different components of the justification play specific roles, such as to rebut a counterargument or to propose a hypothetical situation, that in concert achieve the main goals of the justification. Considered this way, legal analysis can be modeled with an FR in a way that is strongly reminiscent of Toulmin's model of argument. This functional representation can then serve as the basis for a scheme to index a memory containing justifications and justification fragments. Goel [1989] demonstrated the utility of using an FR as an index vocabulary for a case-based memory used in a task of designing physical artifacts. However, he did not address the problem of organizing and searching a large data base of designs. The great affinity between the FR and Toulmin's model of argument in large part inspired the work reported here aimed at addressing that problem.

## 1.4 Research Toward a Solution

### 1.4.1 Objectives

This research focuses on conceptual retrieval for the purpose of legal analysis in taxation, in particular, in the area of captive insurance corporations. In this problem-

solving context, a theory of conceptual retrieval will be presented that elaborates an index vocabulary and organization based on the functional roles of justifications. This theory is then generalized to the class of tasks for which it is appropriate. The present research has four primary objectives:

(1) To develop a methodology for the functional representation of justifications based on the ideas of Toulmin.

(2) To develop a model of conceptual retrieval — index vocabulary, index organization, and retrieval method — motivated by this methodology for representing justifications.

(3) To design a conceptual memory of legal cases based on this model and integrate it with a problem-solving architecture for justifying classifications in the domain of tax law.

(4) To construct, as a proof of principle, a working knowledge-based system based on these concepts.

### 1.4.2 Issues

Rather than trying to solve the conceptual retrieval problem solely in terms of general knowledge of a domain, this project reflects the view that knowing how retrieved items will be used provides extra guidance in developing more useful indexing vocabularies and methods. Such task-specific structures augment general-purpose indices with a richer vocabulary for representing and retrieving information. Furthermore, vocabularies derived from task structures can often elucidate the nature of the terms used in more general-purpose indices. Certain theoretical questions arise concerning the relationship between different models for conceptual retrieval, each based on different intuitions and principles.

To this end, this research proposes a view of conceptual retrieval that relates general-purpose and task-specific models.

The work presented here addresses several issues of importance to two related research areas, AI in law and case-based reasoning. While this research does not propose a new theory of legal analysis, it does formalize a view of legal analysis that is widely accepted in the legal AI community. The generic-task problem-solving architecture thus proposed embodies particular notions regarding the interaction between case memory and a legal justification problem solver. The theory of conceptual retrieval advanced in this work is relevant to many issues in the field of case-based reasoning, in particular the selection of an index vocabulary, the organization of cases in memory, and the use of cases in problem solving. With respect to these two research disciplines, the central contribution of this work lies in its description of how one can employ knowledge of a device and its teleology in constructing more useful and efficacious case memories.

## 1.5 A Synopsis of This Work

The remainder of this volume describes in detail work undertaken to achieve the objectives outlined above. The dissertation comprises three organizational sections: problem analysis, problem solution, and theoretical discussion of the problem and solution.

Chapters 2 and 3 present the problem analysis. Chapter 2 analyzes the conceptual retrieval problem itself and considers research aimed at resolving it from the information retrieval, cognitive psychology, and case-based reasoning communities. Chapter 3 then turns to the task of legal analysis in the domain of tax accounting. A review of several AI approaches to legal classification and justification is given, followed by a conceptual review of tax accounting and issues involving captive insurance corporations.

A solution to the conceptual retrieval problem in this context is described in Chapters 4 through 7. Chapter 4 presents a generic-task problem solving architecture for

the task of legal analysis. Chapter 5 details a technique for analyzing and representing justifications using a Functional Representation scheme based on the ideas of Toulmin. Using this technique as a foundation, Chapter 6 develops a functional model of conceptual retrieval, from the selection of index vocabulary to the organization and search of case memory. Finally, this model is integrated with the problem-solving architecture, and Chapter 7 describes CRISTA[1], a computer program that embodies the model of conceptual retrieval described here. Chapter 7 also presents examples of CRISTA's problem solving, focusing on its use of case memory.

Chapters 8 and 9 offer a theoretical discussion of this work. Chapter 8 evaluates the work presented here with respect to other approaches and discusses how this work might be united with work on some other active research issues in AI. Chapter 9 then concludes the dissertation with a consideration of the work's ultimate contributions and significant avenues for future research.

---

1 For "Conceptual Retrieval In the Service of Tax Argumentation."

# CHAPTER 2

# CONCEPTUAL RETRIEVAL: AN ANALYSIS

## 2.1 Introduction

In domains with large bodies of technical documents, computer-aided research has become a standard practice. The law offers an extreme example of this phenomenon. The jurisprudential tenet *stare decisis* mandates that prior decisions be given precedence in deciding new cases. This rule imparts great significance to the task of legal research, leading to the accumulation of vast numbers of case documents to be used in analyzing subsequent cases. In this context, computer-aided research poses a serious challenge: how can one locate in an immense data base just those records or documents relevant to a particular topic of interest?

The idea of systems that index and retrieve information using semantic knowledge of a domain has been termed *conceptual retrieval* (CR). This chapter discusses the CR problem and research aimed at solving it. The remainder of the chapter consists of five main parts. Section 2 describes the traditional approach to information retrieval, key-word search, and its principle shortcomings. Sections 3 through 5 consider three prominent lines of interdisciplinary work that offer potential solutions for CR — information retrieval, cognitive psychology, and case-based reasoning. Drawing on results from these research areas, Section 6 outlines the contributions that each approach offers toward the realization of conceptual retrieval systems. Finally, in the conclusion, the common themes that guide the work described in the rest of this dissertation are presented.

## 2.2 The Traditional Solution: Key-Word Search

Conventional information retrieval (IR) systems implement the "key word in combination" (KWIC) method introduced by Horty [1962] and his colleagues.[1] In the KWIC approach, the retrieval system creates an index on the full text of a document that lists the exact location of each significant word in the document.[2] Queries to the system consist of one or more text words connected using simple Boolean and adjacency operators, and the system retrieves all documents in which the specified words appear in the combination specified by the query. Thus, this approach defines its index vocabulary, index organization, and retrieval algorithm in purely lexical terms — the index vocabulary equals the set of significant words appearing in the document, the organization of these terms consists in an inverted file on these words, and documents are retrieved when they contain the words specified in the query.

Despite its simplicity, the KWIC method produced an important breakthrough in the field of information retrieval. KWIC made possible the construction of large IR systems consisting of thousands of pages of text. In addition, it spawned a wealth of research activity focused on the efficient generation and use of index files from text files. A large majority of the information retrieval research community still devotes its primary efforts to improving the utility of KWIC approaches.

However, systems based on this methodology suffer from one crucial problem: *there is no necessary connection between a word and the meaning of a text in which the word appears or does not appear.* This truth follows from the duality of expressiveness and ambiguity in natural language [Krovetz 1985]. Generally, one can express a concept in

---

1 See also the work of Kehl, Horty, Bacon, and Mitchell [1961]. For a general discussion of early retrieval systems, see Buchanan and Headrick's seminal paper on AI applications in the law [1970, pages 41-46] .

2 Only words such as articles, auxiliary verbs, and prepositions are considered "insignificant."

many different ways, and each word may have multiple meanings. These semantic ambiguities can be resolved only in the context of a particular usage.

The result of this problem for KWIC is a nearly inevitable trade-off between precision (the relevance of retrieved documents) and recall (the retrieval of relevant documents). A query specific enough to insure that most documents retrieved will be relevant often leaves many relevant items unretrieved, but a query general enough to insure high recall will usually also lead to the retrieval of many irrelevant documents. Even in a field characterized by technical discourse, such as the law, this problem persists. Blair and Maron [1985] found that attorneys using key-word retrieval systems perceived that recall from their queries exceeded 75% when only 20% of the relevant items were actually retrieved.

This shortcoming of the KWIC approach gives rise to a number of practical problems in its use. Hafner [1981] described the infeasibility of specifying all possible word combinations that could express a legal issue or relation, thus indicating the difficulty of achieving nearly-complete recall in practice. On the other end of the spectrum, Bing [1978] found that lawyers tend to make critical errors in specifying complex key-word queries. These errors typically involve the incorrect combination of AND, OR, and NOT operators. The results of Bing's study dispel hopes for a high degree of precision.

Other practical problems arise, too. If a new legal decision or statute coins a phrase in discussing a particular issue, relevant cases decided thereafter will likely contain the phrase, but relevant earlier cases will not. Or one may wish to retrieve documents that only partially match a given fact pattern, for instance, in trying to find a larger body of potentially relevant precedent for a new concept. Generating a useful key-word query for this situation will be difficult, if not impossible. Key words either appear in a document, or they do not; there is no room for gradation.

Ultimately, these difficulties can be traced to the fact that key-word systems force *users* to focus on syntactic phenomena — the lexical details of documents — rather than on the meaning of the documents they seek. The user must not only understand the domain and its semantics but also know how to translate such understanding into lexical queries of the data base. By placing this burden on the user, the key-word system serves only as a mechanized index to a body of documents, providing little additional assistance for the task of computer-aided research.

The rapid growth in the number of documents to be searched, the importance of the research task in many professions, and the documented shortcomings of KWIC systems, however, all indicate a clear need for systems that go beyond text search to information retrieval based on the semantics of the domain and its documents. Such an "intelligent" research assistant would mediate between the conceptual vocabulary of the user and the lexical vocabulary of stored items. In doing so, the system would subsume much of the implementation knowledge involved in generating queries.

### 2.2.1 Conceptual Approaches to the Retrieval Problem

Researchers from a variety of disciplines have investigated issues relevant to solving the CR problem. Three prominent lines of this interdisciplinary research are:

- information retrieval, which has applied its own lessons and some of those from AI to the particular problem of legal information retrieval;
- cognitive psychology, which through its investigation of analogical reasoning has identified important aspects of how humans retrieve and use past experiences in solving new problems; and

- case-based reasoning, which has developed a large body of results on how computer systems can index, store, retrieve, and use past cases in problem solving.

Each of these research tracks addresses the conceptual retrieval problem in its own context, and consideration of their different perspectives offers a more complete picture of the problem. By combining their individual contributions, one can arrive at an informed strategy for constructing CR systems.

## 2.3 Information Retrieval

A number of researchers in the information retrieval community have pursued the notion of conceptual retrieval systems. Focusing on particular domain needs, these lines of study seek either to extend the traditional model of text-based retrieval with knowledge-based constructs or to develop new techniques for addressing traditional problems. Legal information retrieval has provided an especially fertile area of study because of its great practical importance and its archetypal nature. Case law encompasses the full range of natural language problems inherent in full-text retrieval while also resting on a well-developed and continually evolving body of conceptual knowledge. In addition, researchers in AI have long been interested in legal problem solving, thus providing background for representing and manipulating legal knowledge.

Researchers interested in extending the traditional IR model have typically investigated the utility of front-end tools that assist the user and then create key-word queries for a full-text system. Bing [1987] has proposed a "norm-based thesaurus" approach in which the user traverses a hierarchy of norms (legal rules) and identifies concepts relevant to the situation at hand. From the selected nodes, the system would automatically generate a key-word query that incorporates different senses of each concept. For example, the concept *spouse* can be elaborated to include expressions such as *husband*,

*wife*, and *married*. While acknowledging that such a system would still be limited to providing synonym support, Bing suggests that the use of formalized legal rules will at least allow automatic generation of some complex queries.

Another system of this sort, RUBRIC, was first developed as a general intelligent retrieval methodology [Tong and Shapiro 1985] and was then applied to the problem of legal document retrieval [Tong, Reid, Crowe, and Douglas 1987]. Using a hierarchy of production rules to represent conceptual relationships such as implication and synonymy, RUBRIC allows for "fuzzy" definition of concepts in terms of domain text. An example of a RUBRIC production rule appears in Table 1.

**Table 1.** A RUBRIC Production Rule (Adapted from Tong *et al.* [1987])

```
( EVIDENCE financing-condition
        (       (OR     (PHRASE "obtain" "sufficient" "funds")
                        (PHRASE "sufficient" "financing" "obtained")
                        (PHRASE "arrange" "sufficient" "funds")
                        (PHRASE "availability" "sufficient" "funds")
                )
                0.8     )       )
```

This rule states that the concept of a *financing condition* is considered very relevant (0.8 on a 0-1 scale) in the presence of any of the standard legal phrases supplied. Given a query in natural-language text, RUBRIC can use such rules to identify concepts relevant to the query. The system then retrieves documents from memory that have been tagged by the requested concepts.[3] RUBRIC's builders argue that the user must still work hard to generate a good query, but at least retrieval is based on concepts rather than text.

---

3 The numeric "relevancy factors" are manipulated in much the same way as MYCIN's certainty factors. They allow RUBRIC to combine concepts and text expressions based on Boolean operators.

### 2.3.1 An AI-Based Model of Information Retrieval

Hafner [1981] conducted the earliest effort to develop techniques grounded in AI specifically to address the traditional problems of information retrieval. Hafner's "Legal Information Retrieval System" (LIRS) employed a model of legal classification knowledge, in the form of a restricted semantic network, defined on the area of negotiable instruments law. This domain model consisted of a network of approximately three hundred objects — agents, events, and various negotiable instruments, their components, properties, and roles. Connecting these nodes were links denoting membership, constituency, property, and event-condition relationships.

From this model, Hafner developed a *situation description language* (SDL) for describing documents in the data base and for expressing data base queries. The SDL allowed simple binary relationships among objects and properties and included primitives for expressing negation of relationships, a particularly well-developed result of this project. The LIRS data base contained entries for approximately 200 statutes and 200 cases, as well as entries for official comments on the statute by legal authorities. Each item in the data base was represented by a set of SDL descriptors and full legal citations. An entry's descriptors denoted information regarding legal setting, concept definition, and legal issues. In order to facilitate retrieval, each concept in the domain model held a set of pointers to data base items related to the concept in a particular way (for example, by presence or absence).

LIRS took a query, determined the indices corresponding to the concepts in the query, composed an appropriate descriptor based on the index types and the referred concepts, and compared this composed descriptor to the those of items indicated by the concepts index list. Entries matching the query descriptor were then returned by the data base. Complex queries — those including conjunction or disjunction operators — were decomposed into primitive queries, and the results of processing the primitive queries were

then re-combined according to the complex operators. In addition, the data base would match queries to documents at varying levels of semantic complexity, as specified by the user. These match types included an exact match as well as direct and inferred extensions of the query.

While the LIRS project addressed important questions regarding legal knowledge representation and conceptual retrieval, the project did not directly tackle the issue of how to organize the data base for efficient retrieval. Hafner [1987] subsequently refined and extended the LIRS approach to incorporate an *issue/case discrimination tree* for indexing a memory of legal cases. In this extended approach, individual case descriptors consist of constellations of concepts, drawn from a model of domain knowledge and assembled according to the legal roles they play in the case. Each concept in the domain has an associated set of decision rules that specifies the conditions under which the concept is present. These rules also constitute the primitives for representing the justifying theory of the case, one of the critical components of a case descriptor.[4]

Cases in memory are indexed by an issue/case discrimination tree (Figure 1), which depicts normative relationships among legal issues and relevant domain facts. Issue nodes denote legal conclusions to be drawn in deciding a case, and factor nodes denote types of facts that may influence the process of drawing a conclusion. Influence links connect issues to sub-issues and to factors relevant in deciding the issue. Each link reflects a positive or negative relationship between the connected concepts.

---

4 The *theory* of a case decision demonstrates how the legal issues of the case relate to the case's background facts (which define the situation being considered) and the holdings of the court (the decision itself). One can view the theory of a case as a simple justification of the decision, with little elaboration of the inference.

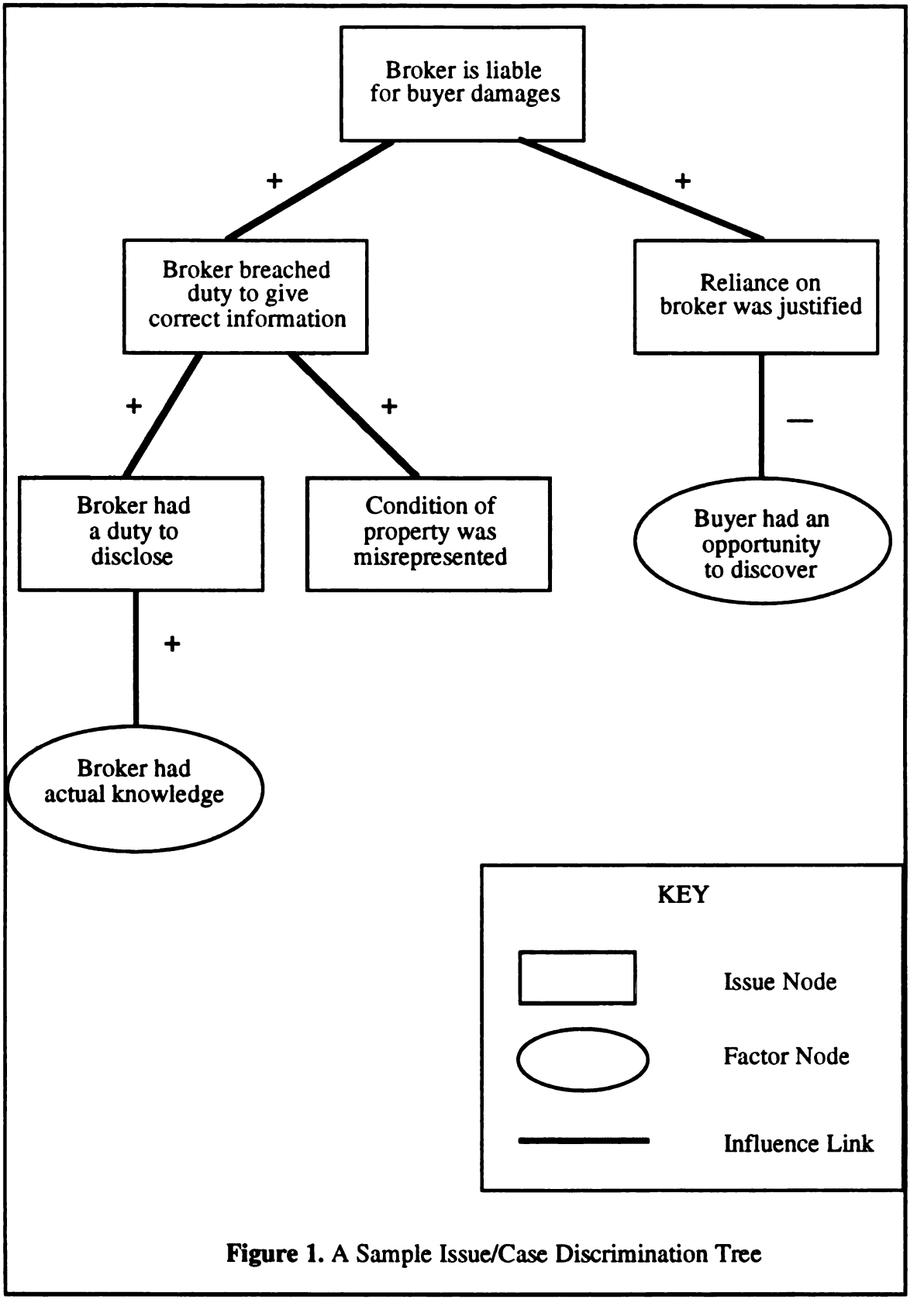


**Figure 1.** A Sample Issue/Case Discrimination Tree

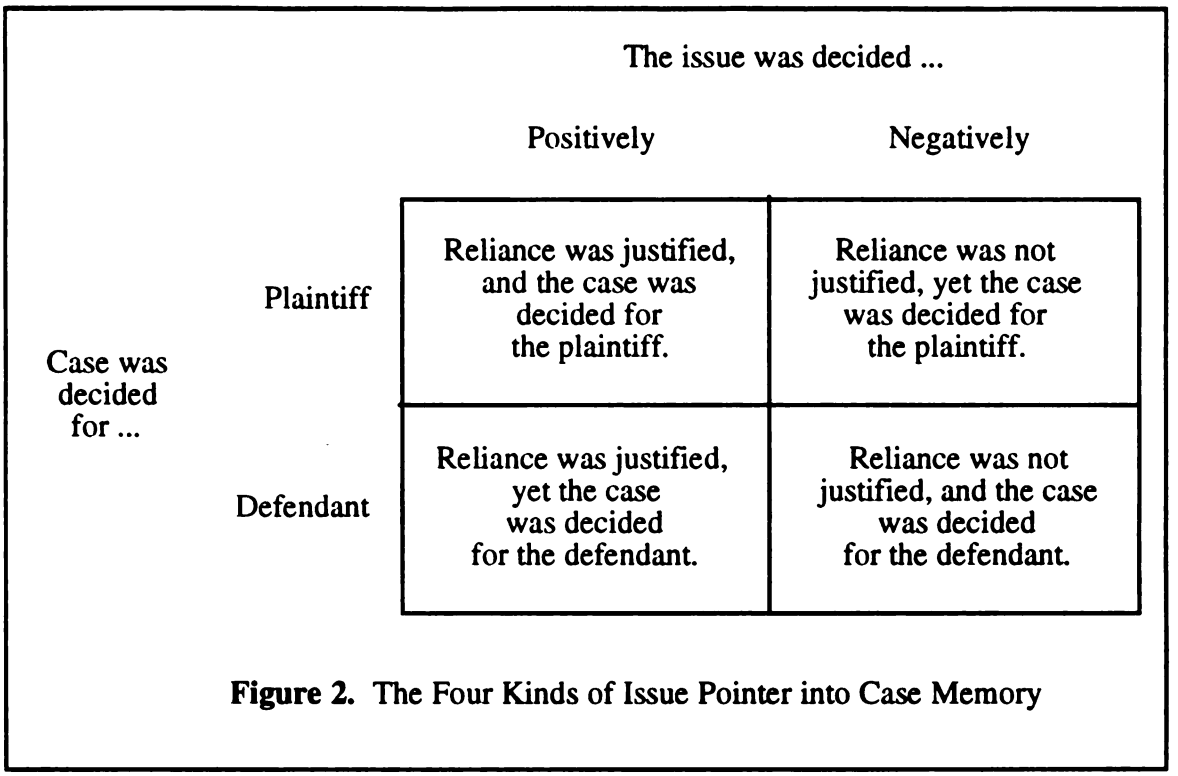


**Figure 2.** The Four Kinds of Issue Pointer into Case Memory

From this structure, two kinds of index into the case memory can be established. First, each node in the tree has a compound link to the case law collection (Figure 2). This link points to the set of all cases in which the issue or factor played a role. For example, the node for *justified reliance* points to all cases in which the buyer claims to have been justified in relying on the broker's presentation of facts. This set of cases is segregated into four groups, based on how the issue was decided and on how the case was ultimately decided. This kind of link relates how an issue was decided to the set of cases in which the issue was important.

Second, each *link* in the tree also has a compound pointer to the cases in which the influence relationship was relevant to disposition of the case. This set of cases is also divided into four groups, along two dimensions: (1) the presence or absence of the factor or sub-issue in the case, and (2) the effect its presence or absence had on the resolution of

the issue. Figure 3 depicts the division of the case set for a particular influence link, the link between the issue *duty to disclose* and the factor *broker had actual knowledge*. This type of organization allows the system to provide meaningful answers to very specific queries, such as "In what cases did the broker have actual knowledge of a defect and yet the court decide that there was no duty to disclose?" Given this sort of link, the system can answer queries referring not only to relevant concepts but also to the relevance of the *relationships* among concepts in a case.

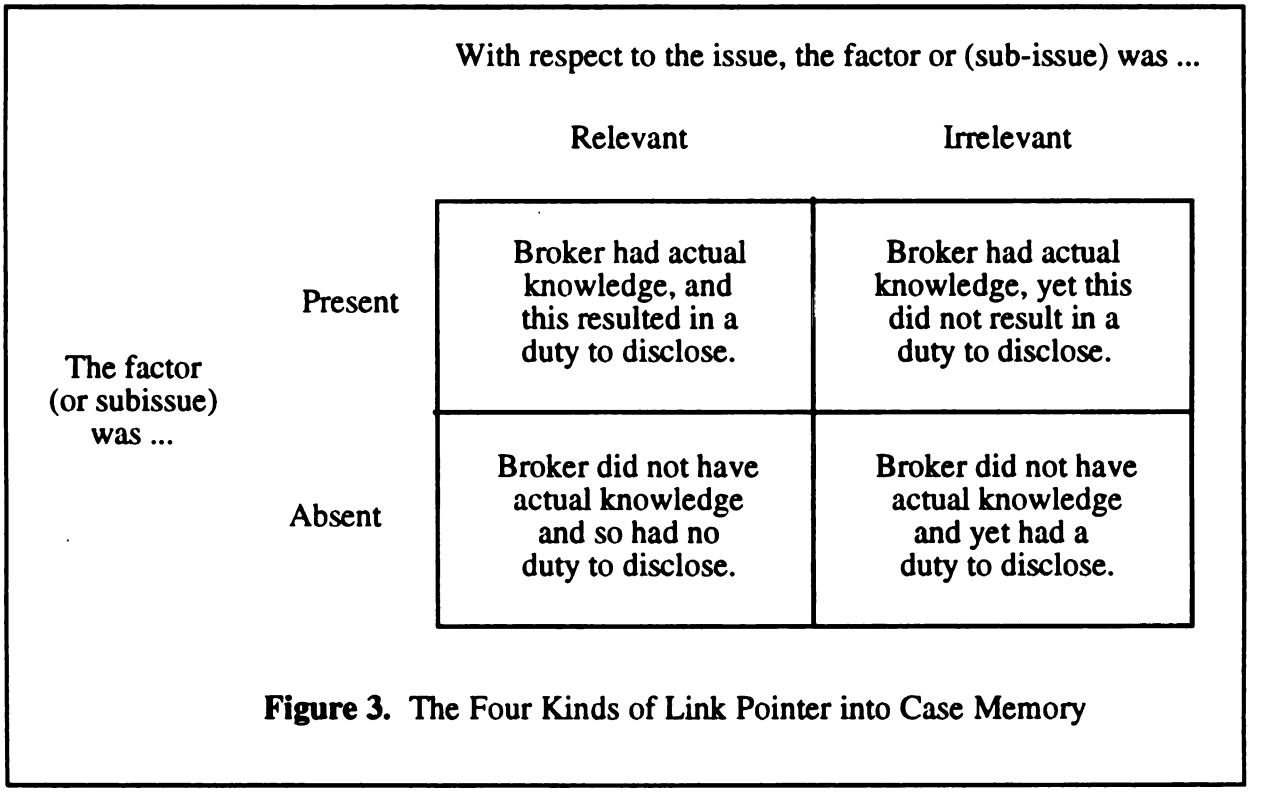


**Figure 3.** The Four Kinds of Link Pointer into Case Memory

### 2.3.2 AI-Based Approaches and Their Significance

Hafner's model demonstrates the power of pattern matching (through abstraction to domain concepts) for conceptual information retrieval. This idea is an old one, dating at least back to Raphael [1968] and his SIR program, and finds application in a variety of

forms. Zarri [1985] has proposed a methodology for intelligent information retrieval in which data in the memory are represented as frames composed of sentences from a case grammar that defines the domain. Queries are then decomposed along the dimensions of the cases in the grammar, and each access to the memory is guided by the semantic content of the classes of patterns associated with the case. From an entirely different perspective, Rose and Belew [1989, also Belew 1987] propose a connectionist approach to conceptual retrieval. In this approach, the retrieval system begins with a network constructed from an initial representation of the documents it contains, linking nodes that are instances of the important classes of objects in the domain. As users interact with the system, it automatically refines the weights on the network's connections to account for the utility of retrieved documents and to reflect the addition of new documents.

Adopting a view of conceptual retrieval that rests foremost on principles of AI garners advantages unavailable to the approaches that rely primarily on traditional IR techniques. Tools such as those proposed by Bing [1987] and Tong [with Reid, Crowe, and Douglas 1987] suffice because of the nature of the domains in which they were used. RUBRIC succeeds because it is applied to a case law domain in which concepts are characterized by complex permutations of a relatively small number of text patterns. In a domain where concepts might be expressed with a broader range of text patterns, RUBRIC's production rules would quickly become too large and too numerous for the approach to be feasible. Bing's norm-based thesaurus succeeds because it is applied within the European model of law, with its strict adherence to statute and not case law. One may be able to express all the synonyms for a concept that might appear in statute, but the text of case decisions is typically too diverse for a purely synonym-based approach to offer much advantage.

These tools remain too faithful to traditional IR to avoid its principal problems. A text- or synonym-based approach makes generating queries involving relationships among

concepts nearly impossible, given the combinatorial explosion of different phrases that might be used to express the concepts and the relationship. For this reason, Hafner [1987] argues that the greatest promise for CR in the law lies in (1) understanding how legal experts remember and classify cases, and (2) converting this domain knowledge into appropriate data structures and algorithms.

## 2.4 Cognitive Psychology

Much work in cognitive psychology deals directly or indirectly with the retrieval of information from semantic memory. Studies of concept formation and recognition involve determining whether, and under what conditions, a given item in memory will be evoked in the presence of some input. From the time of Aristotle, concepts were defined by sets of necessary and sufficient conditions that characterized instances of the concepts. Classic psychological studies, such as those reported by Rosch [1975], describe a shift from this Aristotelian notion to a view in which concepts are characterized by prototypical instances. Related instances are linked to the prototypes according to the degree of similarity, with more similar instances "closer" than less similar instances.

In order to explicate this sort of theory, cognitive psychologists have been forced to consider the mechanism that enables one concept to evoke another. Similarly, research on how humans understand and resolve experience (for example, using scripts of typical events) and how they behave in the presence of incomplete knowledge has offered insights into how memory is organized and accessed. An especially interesting area of psychology influenced in this way has been the study of analogical reasoning.

Solving problems by analogy — recognizing that an already-solved problem is similar to a current one, and adapting the old solution to the new situation — plays an important role in many domains. One can apply old solutions in a variety of ways:

- by transferring the solution directly,

- by modifying the solution to account for differences between the problems,
- by transferring the method used to derive the solution and applying it to the new situation, or
- by creating an abstraction of the problem and solution and applying this generalization to the new situation.

According to Greeno [1978], analogies can be characterized more generally as problems of *inducing structure*: take the givens of an old situation, determine the structural relationship among its elements, and apply this relationship to the new situation. Analogy, then, requires that an appropriate analog be available before its structure can be induced and applied in a new situation. Under contrived conditions, such as in standardized testing, the analog is typically provided to the problem solver. Studies have shown that people are often quite adept at solving problems by analogy when given an appropriate analog from which to induce a structural relationship [Gentner 1989, Sternberg 1977]. More generally, though, a problem solver must remember or otherwise actively retrieve an analog for use.

It is in remembering useful analogs that people have difficulty. Gick and Holyoak [1980] sought to determine how to overcome this difficulty. They found that hints consisting of key phrases, such as "divide and conquer," promote the retrieval of relevant past experiences for use in a new situation. Along similar lines, Gentner and Toupin [1986] conducted studies in an attempt to discover the source of difficulty in recognizing analogs (in this case, stories). They found that subjects had difficulty recognizing the utility of a past story when it was not presented — or not understood — in a systematic way. This condition was most readily observable when the causal relationships among elements of the story were not clear. Thus, one of the central goals for understanding analogical reasoning involves determining what types of systematicity characterize the organization of human memory and best facilitate the retrieval of relevant analogs.

### 2.4.1 Memory Organization and Access

Two schools of thought exist among cognitive psychologists regarding memory organization. One group holds that memory is indexed only by surface features and that all retrieval is driven by sensory input. On this view, any use of higher-level concepts occurs after retrieval of analogs sharing similar surface features. The second camp proposes that concepts do play a role as indices into memory. Members of this camp argue that the use of such "abstract" features is directly tied to the kind of higher-level cognitive processes which have traditionally been hard to isolate in experimentation. The following sections discuss these two views of memory in greater detail.

*Retrieval based on Surface Features* A large body of evidence supports the view that, without outside facilitation of some sort, people generally access analogs based only on relatively simple surface feature similarities.[5] As a result, some researchers have proposed highly parallel models of memory that emphasize syntactic and low-level semantic similarity. Thagard and Holyoak [1989] offer a theory of retrieval based on the simultaneous satisfaction of three types of constraints on memory: semantic similarity, structural consistency, and pragmatic centrality. These constraints are not necessary requirements on retrieval but rather pressures that operate on the memory to different degrees. Waltz [1989] argues that massive parallelism over very low-level syntactic features best accounts for human behavior. From this viewpoint, concept-level indices are *outputs of*, not *inputs to*, the retrieval process. The point of such architectural models of memory access is, Waltz says, that people "can retrieve deep, evaluative features rapidly, given only readily extractable surface features." [page 41]

In order to account both for ample experimental evidence of retrieval based on simple features and for experimental and anecdotal indications that higher-level conceptual

---

5 Gentner [1989] describes a variety of such studies, in addition to those cited above.

features affect retrieval, some researchers have proposed theories that operate on two levels. Forbus and Gentner [1991] offer such a theory in their MAC/FAC[6] cognitive architecture. This approach was motivated by the insufficiencies of competing types of models for memory organization. AI models of memory have tended to be based on clever indexing methods that seem unlikely to scale to the demands placed on a large, evolving memory. Conversely, models of memory from cognitive psychology have typically employed simple representation schemes, such as feature vectors, that allow tractable large-scale search but which do not reflect the richness of the semantic features that people possess and use.

MAC/FAC blends the strengths of these types of model in a two-stage computation. First, the MAC stage uses a flat feature representation to retrieve a set of potential analogs at little computational cost. This kind of blanket search is often very imprecise, resulting in the retrieval of many inaccurate items.[7] Second, the FAC stage attempts to match the input "query" with these items based on their structure and higher-level semantic content. Forbus and Gentner assert that this model strikes a compromise between "seemingly incompatible intuitions about memory: Access tends to be governed by surface properties, while inference tends to be governed by relational matches." [page 68] These and other researchers, including Gentner [1983] and Burstein [1989], have long been investigating the nature of the structural and relational mapping necessary for the second stage of such a bicameral process.

*Retrieval based on Abstract Features* Despite these claims that readily extractable low-level features serve as the keys to human memory retrieval, other research has shown that

---

6 For "Many are called, but few are chosen."

7 By erring on the side of imprecision, this type of search tends to have very high recall. However, as Martin [1989] points out, even such a broad retrieval strategy cannot be guaranteed to retrieve all relevant items.

retrieval is also strongly influenced by pragmatic features, such as the processing task at the time of storage. Seifert [1988] has conducted a long line of experiments whose results support this view. These experiments show that the retrieval of stories from memory is facilitated by attending to the processing goals present when the story was initially encountered. Motivated in part by Schank's view that "remindings"[8] reveal the way in which an event was understood, Seifert has investigated the conditions under which such remindings occur.

Her experiments indicate that encoding a new story does not always activate thematically similar stories. Rather, similar stories are activated only when the encoding context presents a functional or strategic purpose for doing so. Cognitive goals thus form a critical part of the context in which encoding and retrieval operate. Seifert's studies indicate that such goals are particularly active in problem solving tasks such as planning and argumentation [page 365]. In some sense, these results countermand Schank's emphasis on automatic remindings, especially if one defines the notion of intentional reminding broadly, to include not only the consciously intended but also retrievals that occur due to cognitive *biases* in processing.

Seifert [1988] and Gentner [1989] argue that processing goals play a role in retrieval by affecting the features that are most attended to in the process of determining the similarity between items in memory. On this view, the mind attends to particular features of a new story based on the cognitive goals active in working memory at the time of encoding. Stories encoded under similar conditions are likely to be activated in the process. Interestingly, many past psychological experiments may have obscured this phenomenon by setting up simplistic contexts in which only one processing goal is

---

8 Schank [1982] coined the term "reminding" to refer to the spontaneous remembering of past experiences based on a current situation. He also uses the term to denote the result of such remembering, a past episode.

present.[9] This kind of experiment allows the goal to be *implicitly* encoded as part of the story's representation in memory. (On the other hand, AI studies of problem solving generally involve tasks with many different processing goals, thus making memory access based on goals a natural consequence of system building.)

That psychological models of memory organization should account for high-level features as indices remains a controversial yet active empirical issue. Researchers who adopt this view believe that it can be reconciled with simple feature-based approaches by developing theories that explain how and when high-level features participate in remembering. Such theories typically focus on how strategic purpose affects what is considered "relevant" in a given situation. Burstein [1989] investigates how queries of different forms affect the mapping techniques involved in arriving at a suitable analogy. (For example, he has compared queries that involve determining the functional organization of a device given its behavior to queries that involve finding a component capable of performing a particular behavior given a functional specification.) Viewing analogy as a matter of inducing structure ultimately leads to the idea that such structures, and the features that may play a role in mapping structural similarities, can serve as useful components in a theory that explains the phenomena of memory retrieval.

## 2.5 Case-Based Reasoning

Case-based reasoning (CBR) refers to a general methodology in which previously-encountered situations are used as aids in interpreting or solving new problem cases. The motivation for this methodology is to avoid the computational cost of re-solving problems from scratch each time they arise by caching solutions to each problem confronted; if a new situation closely resembles a past situation, the problem solving from the past situation may

---

9 Seifert points to the work of Gick and Holyoak [1983], who found in later studies that controlling for problem-solving context had a material effect on the performance of memory retrieval. See Seifert [1988] for a detailed discussion of this point.

provide a "short cut" in addressing the new situation. Fundamentally, CBR views problem solving as *remembering*, as a memory process rather than a deliberative one: "Finding the right [story of past experience] ... is what we mean by understanding." [Schank 1982] Remembering a similar prior case can not only save recomputation of a right answer but can also help avoid past failures by recognizing the conditions that led to a failure in an earlier case.

Though consistent with much that cognitive psychologists have observed regarding human problem solving in natural settings, CBR research typically places less emphasis on exact replication of human memory behavior. Rather, CBR emphasizes discovery of the principles that underlie reasoning with cases and building useful computer programs to perform tasks in real-world domains. However, in two general senses, CBR reflects the nature of how people solve problems. First, case-based systems can typically find a good but not optimal solution to a problem rather quickly, with less risk of a wildly wrong answer than systems that do not use past cases.[10] Tasks that lend themselves to a case-based approach, then, are typically characterized by multiple solutions in which trade-offs and the ability to compare solutions are more important than optimization and correctness. Second, each new problem solved increases the knowledge of the system, either by noting the success of a solution or by recognizing a failure in the presence of particular domain conditions.

### 2.5.1 Issues in Case-Based Reasoning

The process of case-based reasoning can be portrayed as consisting of the following steps [Riesbeck and Schank 1989]:

1. Characterize the current situation.
2. Retrieve previous cases similar to the situation.

---

10 The truth of this claim critically depends on the ability of a case-based system to recognize similar past cases effectively. Producing this ability is one of the central research aims of CBR.

3. Determine the best match to the situation among the retrieved cases.
4. Adapt the selected case(s) into a solution for the current situation.
5. Apply the solution. If it is satisfactory, assign indices to the new case based on the goals it achieves; if it fails, use the reasons for failure to identify indices of the situation predictive of the failure.
6. Store the situation and solution as a case in memory.

Case retrieval and storage constitute the memory processes of CBR, the reactive interface between case memory and the system. These steps rely specifically on the generation of indices provided by situation characterization and solution application. At the other extreme is case adaptation, a deliberative process that employs task- and domain-specific knowledge to modify a past solution for use in a new situation. The character of the other three steps in the methodology remains a part of active research. Some view these steps as extensions of the memory (for example, Waltz [1989] and Ashley [1990]), while others recognize that substantive problem solving may be required to perform these tasks (most notably, Hammond [1986], Koton [1988], and Kolodner [1991]). This dispute, as it relates to indexing, is discussed further below.

Researchers have generally applied this methodology in a number of ways. Some use the process as a "weak method,"[11] seeking to discover knowledge structures appropriate for applying its steps to various problem-solving tasks [Hammond 1986, Kolodner 1991, Birnbaum *et al.* 1991]. This style emphasizes case adaptation, as prior cases are used to derive and apply a solution to a new problem instance. In another branch of work, often called precedent-based CBR, the method uses cases to classify or evaluate a new situation based on the treatment of past similar situations. This approach, common especially in legal and diagnostic domains, de-emphasizes adaptation and stresses the

---

11 A weak method is a search technique that employs no domain-specific knowledge to guide the search.

ability to characterize and select cases based on similarity [Ashley 1990, Branting 1991, Bareiss 1989].

Much of the focus of CBR research has been placed on issues of adaptation. Since differences between the past and current cases may have significant effects on how the problem should be solved, adapting a case requires complex task- and domain-specific knowledge to ensure that the new solution will adequately account for the differences. Generally, this knowledge has been represented in the form of causal rules that define how to achieve particular goals and how changes to the solution will affect other parts of the solution [Hammond 1986]. One notable approach at developing a theory of how case adaptation can be performed involves the use of model-based reasoning. Koton [1988, 1990] and Goel and Chandrasekaran [1989] have described approaches in which device models, which capture causal understanding of how a device's structure contributes to its behavior, provide a principled means for adapting a solution to a new situation. This approach also assists in the evaluation of case differences by offering recourse to the model in deciding the effects that particular features have on the functioning of the device.

The other primary focus of case-based reasoning research is *indexing*. Whatever can be made of the differences between problem-solving CBR and precedent-based CBR, or of the differences among CBR approaches to various problem-solving tasks, they all share an essential dependence on case retrieval and storage [Kolodner 1991]. One of the goals of CBR is to avoid the need to do deliberative problem solving by remembering and employing past solutions. For this reason, the design of a case-based system should minimize the extent to which the system must rely on "compiled knowledge" to adapt and repair its solutions. The way to minimize deliberative problem solving in such matters is to retrieve the best possible past case — where "best" means the one that is most useful for dealing with the new situation. As the reasoner experiences more and more situations, it becomes increasingly better equipped to use past solutions with little or no modification.

How the memory of cases is organized and what features the reasoner can use to access the memory are critical issues both in building useful systems and in understanding the phenomenon of reasoning with cases.

### 2.5.2 Case Memory and the Indexing Problem

Most researchers agree on the basic issues underlying the indexing problem [Kolodner 1991]. Indices should be predictive of a case's utility in achieving an outcome. They should be abstract enough to indicate useful cases that are related to but different from one another, and yet concrete enough to be recognizable without extensive and counterproductive computation. Each case may have several indices associated with it, depending on the various roles the case might play in future problem solving (for example, combinations of factors responsible for or descriptive of its outcome, or combinations of factors that describe intermediate steps of note in the problem-solving process).

In addition, many researchers (in particular, Hammond [1989] and Birnbaum [1989]) have noted that an important trade-off exists between the cost and utility of different forms of index. That is, indices that can be derived from standard input at a low cost will tend to describe only a very narrow set of cases, while indices that maximize descriptiveness are likely to be computationally expensive to generate. Given this trade-off, an indexing vocabulary must strike a practical and cost-beneficial compromise.

With respect to case retrieval, the issue of indexing introduces a set of related issues [Domeshek 1991a]:

- determining index content and structure,
- matching indices,
- organizing and searching case memory, and
- generating probes to query the memory.

Any proposed indexing methodology must ultimately account for how to do each of these tasks. Not all of these issues have been formally examined in the literature, but the issue of index content has been the subject of considerable theoretical discussion. Such research has focused on this question: should abstract, high-level indices be used and, if so, what sorts of abstractions will offer the greatest power and utility for retrieval? Some researchers advocate the use of only simple, low-level features taken directly from system input. However, tracing back to CBR's roots in the work of Schank, the dominant view of indices has stressed the use of functional indices, ones defined somehow in terms of the system's problem-solving goals. The next two sections discuss these two primary approaches to the issue of index content.

*The Use of Low-Level Features* A small but active group of researchers continues to investigate the utility of surface features as indices for case memory. Waltz [1989] has argued that "indexing" is the wrong way to think about memory retrieval. His claim has two major points. One, indices are not needed for tractability, since memory retrieval is done in parallel. Two, most memory processes use only surface features and basic experiential context for retrieval. Thus, complex indices may be useful for describing problem solving, but they are not necessary to a theory of memory retrieval. He cites the traditional results from cognitive psychology mentioned above as evidence for his theory of "memory-based" reasoning.[12]

Using the Connection Machine, Waltz and his colleagues at Thinking Machines Corporation have demonstrated the utility of this approach for tasks such as learning to read text [Stanfill 1988] and predicting protein structure [Zhang and Waltz 1989]. Others have employed the approach of retrieval based on surface features, albeit on serial machines, for similar sorts of tasks. Especially interesting work has been done by Lehnert [1987] and

---

12 However, as noted above, these results may have been flawed in that they failed to control for problem-solving context [Seifert 1988].

Golding and Rosenbloom [1991] on word and surname pronunciation, respectively. This use of surface features not only eliminates the need to compute indices for queries and storage but also enables the application of a case-based approach to large, existing data bases.

But the approach also lacks a degree of complexity often required in performing and explaining problem solving. Ashley [1990] offers a partial compromise in the form of *dimensions*, relatively low-level factors that affect a legal claim. Dimensions are not surface features of a situation but rather are simple fact abstractions. However, the value of each dimension can be computed from system input data using a single production rule. Each case in memory is then indexed by the dimensions that characterize it. Thus, dimensions preserve the spirit of retrieval based on surface features, while also enabling the use of simple domain abstractions in organizing cases meaningfully.[13]

*The Use of Abstract Indices* The foundation of case-based reasoning research has been organization of memory according to the *functional features* of stored cases [Schank 1982]. These functional features denote the utility of a case in future situations. Schank discussed several types of functional feature, including expectation failures, processing goals, and abstractions (such as the morals of stories). Hammond [1986] describes a theory of memory closely tied to the processing goals of a planner. In CHEF, Hammond's system, past planning episodes are indexed by the goals they satisfied and by explanations of failures. Failure explanations consist of features of the situation causally responsible for the plan's inadequacy. Such features may be direct input or abstractions computed in the course of the planning process.

---

13 Ashley asserts that the nature of legal domains precludes the use of abstractions at a higher level than dimensions, due to the open-ended evolutionary character of legal concepts. Chapter 3 discusses Ashley's model of legal argument and the role of dimensions play in greater detail .

Hammond [1989] argues that the computational cost of using such complex features is, in fact, nil, since they must be generated in the course of problem solving anyway. Going further, Birnbaum [1989] contends that, even if a complex index must be computed outside the course of problem solving, the cost of this computation must be offset against the computational savings realized in using the index. If use of the index results in better case selection, and hence in less computation being required to adapt retrieved cases, then derivation of the index may prove to be cost-beneficial. This situation can occur only when there exists an abstract index vocabulary that is likely to result in the retrieval of more appropriate cases. Research aimed at identifying efficacious task-specific vocabularies (such as that described by Hammond [1989] and Birnbaum *et al.* [1991] for the task of planning) becomes crucial when considered from this perspective.

An even stronger claim is made by Martin [1989], who studied natural language analogies. Martin considered stories that share a common theme and found that, while the relational structures of the stories could be mapped effectively [Gentner 1983], the stories shared no common surface features. These examples demonstrate that "indexing on the basis of simple features ... in the hope that a subsequent structural match will have less work to do *cannot* handle examples that rely on purely structural constraints." [page 295, emphasis added] Thus, on this view, abstract index vocabularies are necessary to the recognition of some relevantly similar cases.

In general, one can view the kind of abstract index vocabulary typically used in CBR as being based on the goals satisfied by the generated solution. For example, in CHEF, indices are computed from the functioning of the plan, either from goals that the plan can achieve or from features predictive of goal failures. More recently, researchers have moved a step further, with the use of explicit *models* of the solution serving as a source of the index vocabulary. Goel [1989, also with Chandrasekaran 1989] has described an approach in which designs are described by the functions they realize. These

functions then serve as indices for design cases stored in memory. Use of such an abstract index not only facilitates efficient retrieval but also expedites identifying the part of a retrieved case that will be most useful in the current situation, since the functions can also index the behaviors that allow the device to perform each of its functions. Following a similar strategy, Sycara and Navinchandra [1991] use behavioral abstractions to organize memory according to the utility of designed components.

## 2.6 Major Themes in Memory Organization

Drawing on research in information retrieval, cognitive psychology, and case-based reasoning, one recognizes that each field is experimenting with the same continuum of approaches to memory organization. This continuum ranges along the dimension of index complexity. One end of the spectrum consists of approaches that advocate the use of little or no indexing at all, relying instead on a content-addressable memory to match purely syntactic features over massively parallel hardware. At the other end of the spectrum, complex indexing schemes are based on problem-solving goals and relations. Given this range of approaches, one can also recognize common themes that pervade all the disciplines and indicate promising directions for further work toward the goal of conceptual retrieval systems:

- the need to provide case retrieval support to human reasoners,
- the need for both simple and abstract features as indices, and
- the utility of problem-solving goals as abstract indices.

### 2.6.1 Case Retrieval for Human Reasoners

Each of these disciplines is addressing the need to provide useful analogs to human problem solvers. This research interest results from an understanding that people are often competent at reasoning with precedents but are not always good at remembering

appropriate ones in the first place. Legal information retrieval research is driven by the fact that the volume of legal cases far outstrips the capabilities of legal researchers to remain abreast of case law development in all domains [Hafner 1987, Bing 1987, Zarri 1985]. Cognitive psychologists have ample empirical evidence that this characteristic holds even in the presence of far fewer episodes [Sternberg 1977, Greeno 1978, Gick and Holyoak 1980]. In the line of case-based reasoning research exemplified by Goel *et al.* [1991] and described more generally by Kolodner [1991], the notion of augmenting human memory by providing cases at appropriate times for human use in a broader problem-solving context has become a primary goal.

Case-based reasoning offers a pragmatic suggestion in the attempt to solve the conceptual retrieval problem: index cases first by their utility, and only then by their surface similarity. Systems that retrieve the most useful cases will presumably be more useful both as assistants to human problem solving and as components of larger computer systems capable of performing difficult information-processing tasks.

### 2.6.2 Simple and Abstract Features as Indices

In the debate over low-level and high-level features as indices, each discipline seems to reach the conclusion that both are necessary in any realistic model of memory. The results of Seifert [1986, 1988], McDougal, Hammond, and Seifert [1991], and Gentner [1983, also with Toupin 1986] demonstrate that human memory incorporates elements of both surface and abstract features in the course of storing and retrieving episodes. Indeed, Ashley's [1990] use of dimensions could be considered an implementation of Forbus and Gentner's [1991] MAC/FAC cognitive architecture. Cases are indexed by relatively low-level domain features and then retrieved *en masse* based on direct feature matching. The retrieved cases are then sifted and selected according to the implicit roles they can play in the assembly of arguments. Simoudis and Miller [1991] also

employ such a strategy but use a form of model-based validation to select the causally relevant cases from among those retrieved.

Hafner's [1987] issue/case discrimination tree uses abstract factors to structure a collection of legal cases according to important domain concepts and their relationships. This structure provides access to cases at varying levels of conceptual and relational abstraction. However, Hafner also notes that access to cases via purely syntactic surface features — the text in the stored documents — will always be needed. As legal theory evolves, researchers will require the ability to retrieve cases dealing with new concepts and with new senses of existing concepts. These concepts, at least initially, will not be in the issue/case discrimination tree. Thus, the user will need access to documents containing particular text phrases that are known to characterize the new concepts.

### 2.6.3 Problem-Solving Goals as Abstract Indices

Finally, a principal theme running through studies of memory organization holds that problem solving context, especially goals, in large part determines which features are salient in any given situation [Seifert and Hammond 1989]. In cognitive psychology, research offers only a vague idea of the nature and source of these goals, and consequently the theme is still interpreted rather broadly [Gentner 1989, Seifert 1988]. Experience derived in building case-based reasoning, however, has provided clearer indications of the character of these goals. In particular, understanding the structure and behavior of the intended solution seems to provide valuable information for retrieving useful cases later. This has been the focus of several different lines of research, including Hammond's [1986] theory of planning, Domeshek's [1991a, 1991b] theory of social advice, and the use of model-based reasoning by Goel [1989] and Koton [1988, 1990]. Ultimately, these approaches all support the general notion that problem-solving goals serve as an important key to understanding and remembering cases.

Goel *et al.* [1991] present a thorough analysis of the sorts of indices that should be used to organize a conceptual memory for the task of conceptual design in architecture. This kind of analysis could be applied to any task and domain. Indeed, their short discussion of memory organization offers the seeds of a methodology for designing conceptual retrieval tools in different domains and for different tasks. Goel *et al.* address the question of how the subtasks that a conceptual designer performs — such as design adaptation and evaluation — affect and mutually interact with case memory structures. Unfortunately, they do not generalize their approach to other tasks or domains. A need for such a generalization clearly exists in regard to conceptual retrieval tools.

## 2.7 Conclusion

Researchers in several disciplines have investigated issues relevant to the notion of conceptual retrieval, and several common themes characterize their results. These themes suggest that one of the vital open issues for further research is to identify appropriate index vocabularies for describing cases in memory. However, Domeshek [1991a] reminds researchers that such a theory must address not only index content but also the organization and search of the index space. Developing abstract index vocabularies in isolation leaves system builders to identify the principles of data base organization and search that typify a vocabulary each time it is applied in a new domain. *Ad hoc* methodologies also offer little justification for believing that constructed systems will be useful or robust.

Among the most important of these themes is the idea that problem-solving goals can serve as especially useful indices for organizing a conceptual memory. Many researchers have focused their efforts on the development of index vocabularies for particular problem-solving tasks, but few have taken advantage of the contributions made by knowledge-based systems research into the nature of problem solving. The rest of this dissertation outlines a principled theory of indexing — vocabulary, organization, and

search — for the task of legal analysis. This research is done within the context of the Generic Task Approach to knowledge-based systems, which offers a set of languages that describe particularly useful problem-solving tasks and methods. By basing the theory of indexing on such a task-specific approach, case memory can be organized and searched according to the specific needs of a problem solver performing a particular task.

# CHAPTER 3

# JUSTIFICATORY REASONING IN TAX LAW: A REVIEW OF PAST WORK

## 3.1 Introduction

Conceptual retrieval does not take place in a vacuum, but rather in the context of some other mental activity. Such activity, whether story understanding or directed problem solving, establishes the goals that retrieval must satisfy. As discussed in Chapter 2, research in cognitive psychology and case-based reasoning indicates that this phenomenon is especially true for complex problem-solving tasks: Problem-solving goals play an important role in how problem solvers understand problems and solutions and in how they organize their memories. This dissertation examines conceptual retrieval in the context of a particular problem-solving task, the task of justificatory reasoning in the domain of tax law. In particular, the taxation of captive insurance arrangements is considered.

The remainder of this chapter consists of four main parts. Section 3.2 examines several AI techniques that have been proposed for the task of legal analysis, which involves the justification of a legal classification. In section 3.3, several approaches to representing justifications are reviewed. Section 3.4 presents a summary of a particular taxation issue, that of captive insurance corporations, and its place in the tax law domain. This domain serves as the test bed for developing a theory of conceptual retrieval based on problem-solving roles. The purpose of these reviews is to establish the task and domain knowledge with respect to which conceptual retrieval is considered in later chapters. Finally, section

3.5 discusses the implications for the idea of conceptual retrieval to be found in this analysis of a specific task and a specific domain.

## 3.2 AI Approaches to Legal Analysis

Legal reasoning consists, in large part, of two tasks, analysis and planning. Analysis involves the classification of a situation[1] as an instance of one or more legal concepts. This classification must be then be justified by identifying relevant statutes and case precedent that support the proposed categorization. In planning, the lawyer views the client's situation prospectively and recommends actions that satisfy the client's goals while minimizing the risk of unfavorable legal consequences. According to Buchanan and Headrick [1970], the analysis task is essentially one of argument formation, and planning consists of argument formation followed by risk assessment. In this sense, the two tasks of legal reasoning interrelate in a fundamental way and share many of the same subtasks.

Researchers in AI have long recognized the law as an attractive domain for study. The legal profession offers a tradition of examining its own reasoning processes, in the specialty of *jurisprudence*, as well as a stylized method for problem solving, in the form of *stare decisis*. Furthermore, the law offers a large volume of accessible and well-structured text to serve as a source of experimental data. AI research in the law (AIL) has typically addressed analytic problems — those of classification and justification — in an effort to identify useful representation and problem-solving techniques that can later be applied to more difficult planning problems. This body of research has evolved through several stages of techniques, from rule-based approaches, through exemplar-based approaches, to case-based approaches that more accurately reflect the legal enterprise.

---

1 The situation is often termed a *fact situation*, to emphasize that the situation is defined by the basic facts that characterize it.

### 3.2.1 Rule-Based Approaches

The earliest computer programs aimed at automating parts of the legal analysis task were systems that represented legal knowledge primarily as rules. Popp and Schlink [1975] described a prototypical system of this generation, Judith, which assisted in the decomposition of legal issues. Given a legal issue, Judith provided a set of propositions that define the issue. For each such subissue, the lawyer could either assert a truth value or ask Judith to decompose the subissue recursively into its constituent propositions. If the analysis "bottomed out" by reaching a subissue for which the lawyer desired more information but the system had no decomposition knowledge, Judith generated a simple query to a legal information retrieval system based on the propositions considered during the current session. Other systems of the day employed similar approaches and embodied similar motivations. For example, CORPTAX [Hellawell 1980] determined whether a stock redemption qualified for favorable tax treatment under a particular provision in the tax law, and Sprowl's [1979] program used a decomposition of a statute to create a simple legal form generator.

Though interesting as initial research vehicles, these early projects suffered from the serious limitations of their basic approach. Responsibility for all of the significant legal analysis lay with the system's programmer and user. Hellawell's work required an exhaustive decision-tree analysis of stock redemption and attribution provisions in the Internal Revenue Code, and Sprowl's document drafting system required a similar level of analysis by the user to decompose a statute and to produce "boilerplate" text. Judith did offer a more flexible goal/subgoal representation of legal rules but was limited by its inability to provide any assistance in searching the goal hierarchy. The only problem-solving method it possessed was the "weak method"[2] of issue decomposition (a user-directed mixture of depth- and breadth-first search).

---

2 A weak method is a search technique that employs no domain-specific knowledge to guide the search.

Attempts to overcome these limitations within the rule-based framework produced two notable systems, McCarty's TAXMAN [1977] and Meldman's legal analysis system [1977]. TAXMAN analyzed corporate stock reorganization transactions. Given a set of high-level assertions about a corporation, its stock ownership, and stock transfers, TAXMAN generated as output the proper tax classification for the transaction.[3] McCarty wrote TAXMAN in MicroPLANNER, a predicate logic-based language with general procedural programming power. By encoding heuristic search rules in MicroPLANNER, McCarty was able to give TAXMAN the ability to perform efficient conceptual abstractions based on experiential knowledge from taxation law. In a similar fashion, Meldman implemented a network of assault-and-battery concepts in a predicate logic language, creating a system that was able to classify fact situations as one or more types of intentional tort. Meldman sought to explicitly represent different forms of legal knowledge — concepts, rules, and generalized cases — using predicate logic rules as the epistemic primitive.

From a Generic Task viewpoint, both McCarty's and Meldman's work can be described as simple forms of hierarchical classification. TAXMAN consisted of a set of small classification hierarchies, where each class was represented as a sub-class of its parent type with additional constraints. These additional constraints constituted the category's establishment knowledge and was implemented as a set of structured pattern-matching rules for abstracting input data into appropriate taxation concepts. Meldman's system also employed a classification hierarchy of tort types and sets of ordered pattern-matching rules for determining the applicability of the type to the current facts at hand. In both systems, the pattern-matching rules closely resemble the generic task of structured matching, providing for a "bottom up" matching of fact patterns to the specifications of more abstract concepts.

---

3 According to Chapter I, Subchapter C, of the Internal Revenue Code of 1954.

The notion of encoding legal concepts and heuristic search knowledge as rules for classificatory analysis of a situation has been adopted by a variety of researchers and practitioners in a variety of tasks and domains.[4] The idea of matching fact situations against templates of event and state classes, with instantiation of variables across relations, permits these systems to address concepts beyond the scope of earlier propositional systems. Yet McCarty [1983, 1987] argued that TAXMAN, the most highly developed example of this approach, could not deal with basic concepts in particular domains of the law because it lacked recourse to a deeper "understanding" of the domain. This limitation follows from the fact that general legal rules abstract *too much* detail away from concept instances, with a resulting inability to recognize fundamental differences between two fact situations. This criticism led to a second generation of legal analysis systems stressing the representation of legal concepts at a more basic level.

### 3.2.2 Exemplar-Based Approaches

Through case analysis, builders of early rule-based systems realized that the rule-based approach would be insufficient for representing amorphous legal concepts. These systems relied on logical templates for expressing concepts, but the domains in which they operated were generally characterized by well-structured semantic relations. This feature was especially present in the tax domains that were chosen. Areas of developing law, though, resist this sort of formulation. Instead, they require an approach based on reasoning from examples.

Levi [1949] presents the classical statement of a legal concept as a set of mappings

---

4 In the general law, these include two systems for determining defendant liability and litigation value, LDS [Peterson and Waterman 1985] and SAL [Waterman, Paul, and Peterson 1986], and a system for the assembly of routine wills and revocable trusts, EPS [Schlobohm and Waterman 1987]. In the tax law, the approach has been applied to the identification of compliance issues with respect to the inclusion of prizes and awards in gross income [MacRae 1985], the determination of constructive ownership of stocks [Schlobohm 1985], and a system for the analysis of prospective pension plans [Grady and Patil 1987]. The most publicized system implemented for use in actual tax practice was ExperTAX, a program that identifies tax accrual, compliance, and planning issues described by Shpilberg, Graham and Schatz [1986].

between fact patterns and a prototypical instance.[5] On an exemplar-based theory, one seeks to show that a situation is an instance of a particular concept by demonstrating a set of value-preserving mappings from the instance to a known instance of the concept. In order to accomplish this kind of reasoning in a computational manner, one requires a scheme for efficiently and adequately representing concept exemplars and the mappings that can be applied to them. This kind of approach holds great promise for the non-computational legal theorist as well, since it enables the development of formal models for the evolution of legal concepts and for the process of analogical reasoning in the law.

Work on the exemplar-based approach quickly focused on the development of languages for representing primitive legal concepts. McCarty's [1983] TAXMAN II project aimed to create a language for describing legal fact patterns in terms of the *rights* and *obligations* of the parties involved. One result of this project has been LLD, a deontic logic that supports the representation and manipulation of mappings between fact patterns [McCarty 1985]. Using LLD, situations can be represented as collections of assertions about the rights and obligations of involved parties, and mappings can be represented as transformations that change certain features of a situation while maintaining other critical features. Mappings can be decomposed into more finely-grained mappings along relevant dimensions in the domain.[6] The conceptual distance between two situations then depends on the number and type of mappings (or "deformations") needed to transform one into the other. Finally, important concepts in the domain are represented as prototypical fact patterns.[7]

---

5 Such a theory of legal construction was offered at least as early as the turn of the century by both Oliver Wendell Holmes and Benjamin Cardozo, eminent legal scholars of the time.

6 In taxation domains, these dimensions might include concepts such as liquidity, risk, and control.

7 McCarty terms his approach "prototype and deformation." Along similar lines, Cross and deBessonet [1985, also deBessonet 1983] have developed a language called CCLIPS for representing mappings that involve the primitive legal notion of *causality*.

Exemplar-based approaches have been tried in a variety of legal domains. Schlobohm [with McCarty 1989] has developed a program, EPS II, that creates estate plans tailored to a client's goals. EPS II selects a prototype plan and then effects a series of mappings from the prototype to a fully-instantiated plan, with each mapping preserving features important to the client. These mappings are expressed in LLD using the primitive relations of *right* and *obligation*. Gardner [1985, 1987] describes a legal analysis system that uses generalized fact patterns to assist in classifying situations from contract law. In essence, these generalized fact patterns correspond to exemplars of particular concepts, such as offer, acceptance, and counteroffer, which can be instantiated based on the facts of an input situation. This system has been used to classify typical situations from a first-year law school contracts course.

The use of exemplars in legal analysis overcomes much of the brittleness found in a purely rule-based approach. Mapping fact situations to prototypical instances of a concept enables a more flexible consideration of a situation's facts. However, some problems remain. Exemplars, while possessing more detail than a single generalized rule, still abstract away details from each instance of the concept. Levi's [1949] discussion of reasoning from examples in the law actually deals with particular cases that have (or have not) been adjudged to be instances of a concept. As a concept evolves, different sets of facts may become relevant to determining whether a situation is an instance of the concept. Abstraction, however painstakingly done, may omit facts of a situation that could later be relevant in classifying an instance.

### 3.2.3 Case-Based Approaches

Motivated by this concern, the current generation of AI research modeling the legal analysis task has focused on case-based reasoning. Ironically, this move to the representation of low-level instances has also enabled legal analysis systems to augment

their classifications with justifications. These justifications conform to legal standards by citing relevant case precedent. This line of research is best characterized by the work of Ashley [1990] and Branting [1989, 1991].

*Ashley and Hypo* Like other legal analysis systems, Hypo takes as input a set of facts that define a particular legal situation (in this case, in trade secret law). As output, Hypo generates one or more "three-ply arguments" to support different classifications of the situation. These arguments are based on the favorable and unfavorable precedents that most closely match the input situation. To assist the user with other aspects of legal analysis, Hypo's output also includes a complete list of all precedent cases that are relevant to the situation at hand. These cases are stored in a library according to the facts that characterize them.

The input situation is analyzed in terms of *dimensions*, abstract legal concepts relevant to arguing the legal merits of a claim. Each dimension contains knowledge of a particular domain factor, the values it can take, and rules that test whether the factor applies to a situation. These dimensions play a variety of roles in the conduct of argument:

- they characterize situations,
- they index the memory of cases according to their presence or absence from the case,
- they differentiate among cases in determining the most relevant of the cases retrieved from memory, and
- they serve as the keys in constructing arguments and in generating useful hypothetical situations.

For Hypo, arguments consist, quite simply, of citations to relevant cases. The first ply of an argument involves citing the case that shares the greatest number of dimensions

with the input situation. The second ply of the argument consists of a response, made on behalf of the opposing party, that takes advantage of some weakness in the mapping between the case cited and the input situation.[8] Finally, the third ply offers a rebuttal on behalf of the original party, countering the response made by the opponent in ply two. Hypo generates such a three-ply argument for each case in memory that shares a maximal subset of dimensions with the input situation.[9]

By maintaining a full set of cases from the domain, Hypo can classify a situation by reference to specific precedent. This approach conforms to *stare decisis* and permits analysis in domains having few or no generalized classification rules. However, Hypo does all classification by case precedent. All the classificatory knowledge in the system resides in the dimensions — they ultimately determine what features of a situation are considered relevant in the domain. To the extent that Hypo is given a complete set of dimensions and a complete set of cases, the system can provide adequate classification by analogy. But Ashley offers no recourse to other classification knowledge.

*Branting and GREBE* Branting [1989, 1991] attempted to flexibly integrate rule-based and case-based knowledge in a single legal analysis system. His program, GREBE, takes an input fact situation and generates as output a single classification, justified by reference to appropriate statutory (or common law) rules and precedent cases. A justification generated by GREBE consists of a chain of reasoning from (some subset of) the input facts to a legal classification of the situation. Each inference in the chain is annotated by the rule or case that justifies the concluding assertion.

---

8 For example, by citing a case that shares a larger number of dimensions with the input situation but that was decided for the opposing party.

9 That is, no other case has a set of facts that subsumes the set of facts shared by the input situation and the case in question.

Rules and cases complement each other in the justification process. For example, the use of a rule to justify a particular inference may be impossible if one of the rule's preconditions is a generalized predicate. Such a generalized predicate will likely not appear as one of the base facts describing the situation. But if a precedent case exists in which the input situation is classified as an instance of the generalized predicate, then the case may be used to show the applicability of the rule to the situation. Thus cases may *operationalize* abstract predicates from legal rules.

Conversely, a precedent case might be useful for supporting a particular inference, except that the facts of the precedent do not match the facts of the current situation closely enough. GREBE could use a legal rule to derive a required fact from the known facts of the situation, thus facilitating a case analogy. Or there may exist a rule that shows one or more of the situation's facts to be a manifestation of a missing abstract term. This term could then be used as a fact about the situation, again facilitating application of case precedent. Branting terms this use of rules *case elaboration.* (He also describes the use of rules for *term reformulation*, another means by which the use of cases is made possible through use of a rule application.)

### 3.2.4 Status of AI in Legal Analysis

Case-based approaches such as Ashley's [1990] and Branting's [1989, 1991] enable legal analysis that relies on specific facts of past classifications. The idea of using multiple forms of knowledge (rules, exemplars, and cases) in legal classification is an old one, tracing back to Meldman [1977] and McCarty [1977]. But only with the advent of case-based analysis systems has the use of particular instances of past classifications been implemented. The move from purely abstract knowledge to a mix of abstract and concrete classification knowledge is essential for doing legal analysis that meets domain standards. This move has also enabled these systems to provide more adequate justification of their

analyses by referring to case precedent when appropriate. As a consequence, interest in the representation and manipulation of justifications has attracted more attention in the AIL community.

Of course, even a case-based approach cannot circumvent the ultimate problem of abstraction. Each case corresponds to specific classification instance and thus contains every fact used to describe its legal situation. However, every case representation abstracts from the "real" situation by including only a subset of the facts that characterize the situation (that subset considered relevant by the encoder). This is true even of full-text opinions generated by the courts. One can take the greatest care in describing a case within some formalism, but there can be no guarantee that the description will be considered complete in some later context. This problem is not specific to legal analysis, though, since it applies to all representation. All other things being equal, cases provide an extra level of detail over more general classificatory knowledge in the law.

## 3.3 Representing Justifications

With the development of more sophisticated techniques for automated legal analysis has come a corresponding growth in interest in the generation of justifications for legal classifications. Legal analysis must ultimately be supported by citation of relevant statute or case precedent. In a very important sense, these justifications — more so than the classifications they support — are the key products of analytic problem solving in the law. Several researchers have proposed techniques for representing justifications or justificatory arguments. Though these researchers do not always offer precise methods for manipulating justifications, their investigations provide insight into how different representations promote the understanding and use of justifications.

These approaches fall into two natural classes. In the first, research is aimed at capturing the *content* and *structure* of justifications — what kinds of assertions compose

justifications, and how these assertions are related. Such work generally seeks to broaden understanding of human argumentation in various domains. In the second class of approaches, justifications are represented so as to facilitate a particular computational method in which justifications play a role. This class can be further subdivided according to the type of argument being investigated:

- tactical argument, concerned with moves that achieve specific argument goals, and
- strategic argument, concerned with moves that establish goals and direct the flow of the argument.

### 3.3.1 The Structure and Content of Justifications

One of the most thorough and influential analyses of justifications was performed by the philosopher of science Toulmin [1958].[10] Toulmin wished to model scientific arguments from a variety of disciplines, but he found the traditional tools of logical analysis inadequate for the task. To remedy this problem, Toulmin devised a model of justificatory argument in which the *roles* played by particular assertions could be made explicit. Furthermore, this model also made explicit the nature of the relationships among these roles. Figure 4 depicts Toulmin's structural model of justifications.

This model defines specific roles that assertions may play in a justification:

- the data that characterize a state of the world,
- the claim to be concluded,
- the warrant that justifies the inference,
- the backing that supports applicability of the warrant,
- a qualification on the inference, and

---

10 See Chapter 5 for another discussion of Toulmin's work, in the context of a computational representation of justifications.

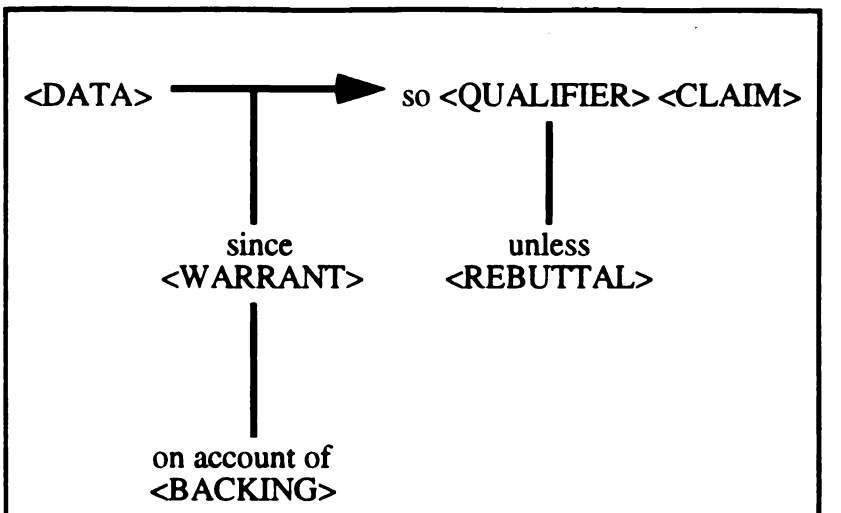


**Figure 4.** Toulmin's Model of Argument

- a rebuttal that contradicts the claim, under the conditions of the qualifier.

These roles either explicate the classical syllogism (by requiring that the nature of the premises be stated explicitly as data, warrant, or backing) or extend it (by providing slots for roles not directly filled by the syllogism's propositions). Such a structure allows one to capture a greater amount of the semantic content carried in a justification.

Several researchers interested in characterization of and computational support for legal analysis or argumentation have adopted Toulmin's rich model as the basis of their approach. Marshall [1989] proposes a general framework that integrates Toulmin's representation of the logical structure of justification with a *pragmatic* structure of argument goals and methods for achieving them. Within such a framework, justifications can be represented and manipulated in terms of case facts and argument goals.[11] Lutomski [1989]

---

11 Unfortunately, Marshall only outlines the integration of logical and pragmatic structures within her framework. Her paper instead offers an excellent description of the use of Toulmin structures for representing chains of inference and for annotating arguments with explanatory comments. This paper provided one of the initial motivations for the work described in this dissertation.

argues that the use of Toulmin's structure, with its focus on the functional roles of assertions, provides a principled method for assimilating statistical evidence into complex abductive arguments in the law. This approach also facilitates the teaching of the use of statistical evidence to novice lawyers. Others who have employed Toulmin structures include Lowe [1985] and Dick [1987]. The generality and completeness of Toulmin's model makes it widely applicable.

### 3.3.2 Strategic Argument Representation

Birnbaum's [1982] *argument molecules* were an early attempt to model strategic relationships among argument assertions. This approach arose out of research in natural language processing aimed at building a computer program that could understand and engage in arguments. In this approach, arguments consisted of networks of propositions connected by *support* and *attack* relations. The goal of the approach was to identify commonly- occurring patterns of support/attack relations and to investigate their use in planning justificatory discourse. Once useful argument molecules were cataloged, the program could use them to guide its selection of processing strategies. When the program recognized a particular molecule in the course of understanding an argument, it could generate an appropriate response based on its knowledge of the structure.

An example of an argument molecule was the *contrastive positions* structure. This relation centers on a mutual attack relation between the two main propositions that sum up alternative viewpoints of the arguers [Flowers, McGuire, and Birnbaum 1982]. Upon recognizing a contrastive positions structure, the program could focus the selection of its strategy based on knowledge of the structure. One of two goals would be established: strengthening the program's main assertion or undermining the opponent's.[12] In this way, the molecule plays a role in understanding the arguments in which it appears.

---

12 In either case, the "net strength" of the mutual attack relations would be shifted in favor of the program's key assertion, either by addition or by subtraction.

Argument molecules reflect the strategic roles played by assertions in justifications. By adding strategic content to Toulmin-like structures, each molecule can initiate expectations regarding the meaning of prior assertions and the utility of possible future assertions. Additionally, Birnbaum, Flowers, and McGuire [1980] maintain that these structures, and the expectations they engender, provide a useful way to organize case memory in support of argumentation. Indexing cases by the expectations that they fulfill or disappoint provides the arguer with a way to retrieve cases at times most advantageous to their use in an argument. This enables cases to assist the user in achieving argument goals indicated by the expectations and in avoiding potential pitfalls based on the opponent's opportunities.

### 3.3.3 Tactical Argument Representation

The work of both Ashley [1989, 1990] and Branting [1988, 1991], discussed above, involves tactical arguments — the assembly of a justification that achieves a given goal, to classify a situation. Rissland [1985] and Ashley [1985, 1990] describe a set of *moves* that an arguer can make in the course of tactical argument:

- citing the most similar favorable precedent case,
- citing a counter-example,
- distinguishing a case, by claiming that a difference between a fact situation and a precedent case justifies a different classification for the situation, or
- shifting the focus of argument, by presenting a related hypothetical situation whose classification depends on a factor more favorable to the arguer.

These moves, like Birnbaum's argument molecules, provide knowledge about arguments that is required for constructing justifications in a domain.

*Ashley and Hypo* In Ashley's model, justifications consist merely of citations to cases that share one or more dimensions with the input situation. Each case consists only of a set of dimensions to which an outcome (a classification) has been assigned. Thus, Hypo employs a trivial Toulmin structure, with base data connected to a conclusion via a case warrant. This lack of structure suffices for two reasons. First, Hypo is intended only as a first-level research assistant; all reasoning needed to flesh out the argument will be done by the system's attorney-user. Second, Hypo incorporates significant domain-specific knowledge for comparing fact situations and for selecting the best cases to cite based on a given fact situation. Most of this knowledge resides in the dimensions themselves, in the form of rules for evaluating the magnitude and value of relevant factors.

However, this approach limits Hypo to drawing simple factual analogies between two cases, since arguments carry no structural information as to *why* a particular outcome follows from a particular set of facts. Hypo's arguments are "flat" — they do not explicitly reflect the inferential relationships among constituent assertions. While Hypo can generate argument moves that achieve particular goals, it cannot represent or manipulate goals explicitly. Nor can Hypo delineate the warrants that authorize specific conclusions in the domain.

*Branting and GREBE* Branting's approach more closely parallels that of Toulmin than Birnbaum's or Ashley's. In GREBE, justifications are networks of assertions in which links carry citations to rules or cases that warrant the corresponding inference. As GREBE attempts to construct a justification, rules or cases may be used to decompose a target inference into a chain of simpler inferences, or they may be used to facilitate the inference by mediating levels of abstraction. A case in memory consists, in part, of the justification for the case situation's classification. These justification structures can be used explicitly in generating new classifications.

In many ways, GREBE represents the most advanced research system for legal analysis. It incorporates multiple forms of knowledge and uses them all in each step of the classification process. By explicitly representing the justifications, Branting is able to gain maximum leverage from prior classifications in addressing new fact situations. The use of Toulmin-like graphs facilitates the use of methods for justification construction that most accurately reflect reasoning in the domain.

## 3.4 The Taxation of Captive Insurance Arrangements

While the law has provided a fruitful domain for AI research, a particular area of the law — taxation — has proven especially attractive and fertile. Tax laws permeate all business dealings. The financial effects of taxation play a such a large role in the long-term profitability of businesses that many accounting and legal resources are devoted to tax compliance and planning. In addition, the complexity and abstractness of taxation law make it an important topic of legal research. McCarty [1983] argues that taxation is a suitable initial target for AI precisely because of its complexity and abstractness. Its artificiality allows AI researchers to investigate important issues germane to legal reasoning without having to tackle the full breadth of human experience. Other areas of the law, such as torts and contracts, rely almost wholly on the understanding of human relationships.[13]

As in other areas of the law, claims for a particular tax classification must be justified by citation of relevant statute and case law. The need to justify tax classifications becomes most acute for unsettled concepts, since each new case offers a potentially new circumstance to be classified. One area of the tax law actively undergoing definition within the courts at this time concerns the issue of *captive insurance arrangements*, in which subsidiaries purport to provide insurance coverage to companies that hold substantial

---

13 Conversely, McCarty notes, law students are taught topics such as torts and contracts first, saving more complex areas such as commercial and corporate law for later. This approach allows new students to rely on their common sense understanding of the world as they obtain a solid foundation in legal principles.

ownership interest in them. Investigation of the issues raised by such corporations and their tax classification has also led the courts, more generally, to refine many of their conclusions regarding what does and does not constitute "insurance." The work presented here, in considering conceptual retrieval for the task of justificatory reasoning in tax law, focuses further on the taxation issue of captive insurance arrangements and the body of case law that is currently evolving around it.

### 3.4.1 Insurance

Insurance is a social device for accumulating reserves to meet uncertain losses. The insured party (the "insured") shifts the risk of a loss to the insurer by paying a fixed *premium* in exchange for coverage in the event of a loss. The insurer assumes the risk of many independent individuals under the assumption that the total losses suffered by the body of insureds will be less than the total premiums paid. To this end, a vast and diverse insurance industry has developed. The concerns of insurers are like those of any businessmen: to sell their products (assumption of casualty risks) at a price (premium) that enables the insurer to make a reasonable profit (the difference between total premiums and total losses).

Because of the potential negative impact of catastrophic loss, the United States tax code allows for the deduction of insurance premiums as a business expense [Duer, Horvitz, and Coberly 1988]. Section 162(a) of the Internal Revenue Code offers the only statutory provision regarding insurance. In part, it states:

> "[T]here shall be allowed as a deduction all the ordinary and necessary expenses paid or incurred during the taxable year in carrying on any trade or business...."

Tax Regulations Section 1.162-1(a) then states that included among these ordinary and necessary business expenses are "...insurance premiums against fire, storm, theft,

accident, or other similar losses in the case of a business...." While these pieces of statute make clear that insurance premiums are deductible as business expenses, the Code goes no further in defining what counts as an "insurance premium."

A body of case law has evolved to better define what does and does not count as insurance and insurance premiums. For example, it is an established matter of law that payments made into a reserve against contingent losses constitute *self-insurance* but do not constitute deductible insurance premiums [Spring Canyon Coal Co. versus the Commissioner of Internal Revenue 1930].[14] The notion of self-insurance contravenes the tax accounting principle that the liability underlying a deductible expense must be both fixed and ascertainable [Tax Regulations Section 1.461-1(a)(2)]. In the case of an insurance contract, while the expense is taken in anticipation of an expected loss, the insurance contract fixes the liability to the amount of the premium[15] and eliminates the possibility of a greater loss. Since self-insurance lacks this important feature, the courts have held that such payments are not deductible.

Other court cases have provided positive criteria for the presence of insurance. In Le Gierse versus Helvering [1941], the Supreme Court held that insurance exists when premiums are paid pursuant to the exchange of an "insurance risk":

> "Historically and commonly insurance involves **risk-shifting and risk-distributing**.... That these elements of risk-shifting and risk-distributing are essential to [an] insurance contract is agreed by courts and commentators." (Emphasis added.)

Thus, whenever one can ascertain that the risk of loss has been shifted and distributed, one can claim that insurance exists. This two-prong test leaves open the question of what constitutes risk shifting and risk distribution.

---

14 All case references appear as a separate bibliographic listing, Appendix A.

15 And perhaps a deductible of a fixed amount.

*Risk Shifting* Again, subsequent court decisions have refined the definition of these open terms (for example, Steere Tank Lines versus the United States [1978] and Commissioner of Internal Revenue versus Treganowan [1950]). Risk shifting typically involves at least three elements:

- the risk of loss is transferred from the insured to the insurer,
- a premium is paid from insured to insurer that is less than the amount of the prospective loss, and
- the insurer possesses the financial capacity to bear the risk and indemnify the insured from loss.

In the analysis of risk shifting, the focus lies on the insured party and its contract with a legitimate insurer. Factors indicating that the insured party no longer bears the risk of a prospective loss provide positive evidence of risk shifting.

*Risk Distribution* The criterion of risk distribution focuses instead on the insurer and its ability to share risk across a large number of parties (that is, to fulfill the social function of insurance). Generally, risk distribution involves three features [O'Brien and Tung 1973]:

- *mass* — The group of insured parties has sufficient exposure to risk for the "law of large numbers" to take effect.
- *homogeneity* — The exposures are sufficiently similar that the risk of loss for each is approximately equal.
- *independence* — Each exposure is distinct enough from the others that the probability of a single event resulting in losses for many insureds is small.

Under these conditions, the law of large numbers holds that the risk borne by the insurer is less than the sum of the risks borne by the insured parties. This property guarantees that

the individual risks of the insureds will be shared among the pool of insureds through the medium of the insurer.

### 3.4.2 Captive Insurance Arrangements

For a variety of reasons, the cost and availability of adequate business insurance coverage have become more unfavorable over the last twenty years. "[I]nsurance costs now rival payroll and occupancy as the largest operating expenses for any business." [Duer, Horvitz, and Coberly 1988] One potential solution to this problem is to establish a *captive insurance* arrangement. In this strategy, a business either purchases an existing insurance company or incorporates a new insurance company for the purpose of obtaining insurance coverage from the "captive" insurer. Such a captive insurance company may also handle part or all of the insurance needs of other related companies, such as other subsidiaries of the captive's parent.

The range of possible captive arrangements is quite broad. The captive may be owned by a single parent corporation or by a group of corporations, or a large portion of its stock may be owned by a single parent with the rest distributed among many third-party shareholders. The captive may insure only the parent, the parent and other related companies, or both related companies and third-party customers. The captive may be incorporated as a U.S. corporation, perhaps in a state with special captive insurance laws[16], or as a foreign corporation. Behind all such arrangements, the unifying theme is the same: the parent company is now able to obtain (reasonably priced) insurance coverage that would otherwise be unavailable.

The advantages of captive insurance can be significant. First and foremost, the availability of insurance preserves the deductibility of premiums and provides a tax benefit. If the captive is incorporated in a foreign jurisdiction, the captive may also pay little or no

---

16 Such as the Colorado Captive Insurance Act, C.R.S. Section 10-6-101-130, 1973.

tax on premium revenue.[17] The non-tax benefits of such arrangements are also attractive. The company can obtain insurance that is not available in commercial markets, whether due to excessive or unusual risks or just to an unfavorable past claims history. By preserving the deductibility of premiums, the company can stabilize its reported earnings over the long term.[18] Furthermore, the captive's status as a *bona fide* insurance company may allow it access to reinsurance markets in which coverage is less expensive than direct insurance. Access to lower-cost coverage can result in lower total insurance costs for the parent and its related companies. Finally, a captive arrangement enables the parent corporation to retain control over the defense of claims made against the firms. Otherwise, independent counsel hired by the insurer decides how and when to defend claims of damage brought by outside parties.

Historically, the Internal Revenue Service (IRS) has viewed captive insurance arrangements with a high degree of skepticism. Its central concern involves the similarity between captive arrangements and self-insurance in which the reserve against losses has been incorporated as a separate entity. If the substance of a captive arrangement does not differ materially from that of self-insurance, then the IRS will eliminate the tax advantages of the arrangement. These advantages are significant enough that corporations have been eager to defend their captive arrangements in the courts. Their main goal is to distinguish the arrangements from self-insurance.

The criteria for the existence of insurance outlined in *Le Gierse* stand as the basic test of captive insurance. New factors come into play, though, in determining whether risk has been shifted or distributed. For many years, the IRS argued that a captive insurer was

---

17 Indeed, many countries, such as Barbados and Bermuda, have created lax tax laws for the express purpose of attracting foreign capital in the form of insurance premiums.

18 Without insurance, the (potentially large) expense of a casualty loss will fall in a single taxable year, or perhaps in a small number of years. With insurance, the cost of such risks is spread over all taxable years in the form of a fixed insurance premium. The result is that the company's net profit will be stable with respect to the cost of such losses.

part of the same "economic family" as its parent and sibling corporations. Under this economic family concept, all risk of loss remains within the family and so, by *definition*, risk could not be shifted. Thus, payments made to a captive insurer, whether by parent or sibling, were not deductible as insurance expense. The courts were sympathetic to this argument for some time. The decisions in Beach Aircraft Corporation versus the United States [1986] and Stearns-Roger Corporation versus the United States [1985] were based largely on acceptance of the economic family argument.

More recently, the courts have begun to recognize that this argument in effect violates the Corporate Entity Doctrine, first enumerated in Moline Properties versus the Commissioner of Internal Revenue [1943]. This doctrine holds that corporations having a legitimate business purpose are to be treated as separate entities for the purposes of taxation. Following this doctrine, the determination of a captive's validity must be based on a consideration of the captive as a separate corporate entity; the existence of an ownership relationship between a captive and a related company should not be used, *a priori*, to deny the existence of insurance between the companies. A recent spate of cases has seen the courts broaden their definition of insurance along these lines.

In Humana versus the Commissioner of Internal Revenue [1989], the court ruled that premiums paid by a sibling corporation to a captive are deductible since, relative to these two entities, risk can be shifted and distributed. This decision still disallowed the deductibility of premiums paid by the parent to the captive, on the grounds that the parent still bore the ultimate financial risk of a loss. Thus, risk was not shifted from parent to captive. However, in Sears, Roebuck, and Co. versus the Commissioner of Internal Revenue [1991] the court acknowledged that in practice risk could be shifted from parent to captive under the right conditions. These conditions require attention to the basic elements of risk shifting and risk distribution. If the elements for risk distribution are satisfied, then

the captive's status as a *bona fide* insurer can play a role in determining whether risk of loss has shifted from the parent insured to the pool of companies insured by the captive.

On this new line of reasoning, the determination of a captive's validity must be based on substantive issues regarding the existence of risk shifting and distribution. These issues are numerous, but the courts have pointed to several factors of particular interest. Among these are:

- the nature of the insurance contract between the insured party and the captive insurer,
- the degree of ownership interest held by the parent in the captive,
- the adequacy of the captive's capitalization for meeting potential losses,
- the captive's history as a bona fide insurance provider,
- the volume of unrelated company risks that are insured by the captive.

These factors have been identified through the consideration of new fact situations that have incrementally extended those considered in precedent cases. As the court argued in Clougherty Packing Company versus the Commissioner of Internal Revenue [1987], at some point along the dimensions that define these factors, the term "captive" may no longer be appropriate to describe these arrangements. Thus, decisions regarding the validity of captive insurance arrangements ultimately depend on the law's standing view of insurance itself.

## 3.5 Implications for Conceptual Retrieval

Chapter 2 describes the importance of understanding task and domain features for conceptual retrieval. As Hammond [1989] argues, "Memory retrieval is a subtask of other methods. First, we analyze the task and its functional needs, and then we determine the

role of memory in these terms." [page 52] The preceding examination of the task of legal analysis and the domain of captive insurance taxation suggests three key requirements on conceptual retrieval for this task and domain: (1) integration of problem solving and retrieval, (2) representation of justification structures, and (3) organization of case memory according to the roles cases play in problem solving.

### 3.5.1 Integration of Problem Solving and Retrieval

With great foresight, Popp and Schlink [1975] declared that "the future of information retrieval systems for the legal profession seems to require that they form an important part of a more complex legal information system rather than being a research tool in their own right." [page 305] Their system, Judith, integrated legal analysis with information retrieval by allowing a user to generate a data base query from the context of an analysis session. Not only did this approach take advantage of the focus provided by the analysis tool, but it also released the attorney-user from having to generate a complex key-word query. Judith could generate a query that was directly "on point" with the analysis done by the attorney thus far, while successfully managing the complexity that leads to the pitfalls described by Bing [1978].[19]

McCarty [1983, 1987] has also argued forcefully for the idea that knowledge-based systems in the law should ideally be hybrids that incorporate analysis, planning, and information retrieval components. He offers two reasons for this integration. First, the information retrieval component will have access to problem-solving knowledge and patterns of inference in the domain, thus permitting more concise and robust retrieval. Second, the problem-solving components will have direct access to case materials that are necessary to justify their conclusions. Given the evolutionary nature of the law, the conceptual retrieval system should ultimately be integrated with both a problem-solving

---

19 See Section 2.2 for a discussion of Bing's findings.

system and with a full-text system, since the latter permits access to the most detailed representation of the facts and reasoning that characterize past cases.

### 3.5.2 Representation of Domain Concepts and Justifications

Yet integration alone cannot insure adequate problem solving and retrieval skills. Many of the rule-based legal analysis systems would still prove to be inadequate because of limitations in the types of knowledge that they possess. Truly robust legal systems will require, in some sense, an *explicit* representation of legal concepts and relationships among them [Cross and deBessonet 1985]. This realization was one of the impetuses for the move to exemplar-based systems, with their notion of prototype and content-preserving mappings to other fact situations. A legal analysis or retrieval system in the domain of captive insurance taxation would require some representation of concepts such as *insurance*, *risk distribution*, and *unrelated risks* in order to efficiently reason about and retrieve situations in which the concepts were relevant.

Due to the critical standards of problem solving in the law, legal analysis systems must justify their classifications. And, as Branting [1988, 1991] showed, performing the task of legal classification *requires* access to the justifications of past classifications. Thus, conceptual retrieval systems must provide access to representations of the justifications underlying precedent cases. Furthermore, such systems will almost certainly need to use such representations in order to perform efficient, focused retrieval. This lesson led to the recent generation of case-based approaches to legal analysis and has engendered a more useful kind of analysis system — one that uses and produces justifications as a part of its problem solving.

### 3.5.3 Organization of Case Memory by Problem-Solving Roles

The desire for a case memory of past justifications leads to a final implication: the need to organize that memory according to the problem-solving roles that the justifications can play in later justifications. This realization has arisen from work on argument both in the law and in other domains. Research into argument molecules [Birnbaum, Flowers, and McGuire 1980; Birnbaum 1982] was directed at natural language understanding and processing of arguments in various domains. One of its key results was that assertions and groups of assertions play particular roles in the conduct of argument and that these roles are useful for indexing memory in a way that enables arguments to be retrieved at the most opportune times — when they are relevant to understanding or generating a later argument.

The work of Ashley [1990] and Branting [1989, 1991] supports this notion as well in the legal domain. Ashley argues that propositions and their *meanings* should be interpreted in the context of their force in justifying claims. Branting's GREBE system relies directly on the ability to retrieve cases when they will be most useful in constructing a new justification. However, Hypo and GREBE employ only feature-based indices. The next step in this progression is to take advantage of such roles in indexing memory. This approach would facilitate direct retrieval of cases according to their utility in problem solving and make for a more coherent organization of memory. These roles are already part of the vocabulary of the problem solver and thus offer a natural source of indices into memory. In the end, this kind of memory organization offers increased support for realistic justificatory problem solving in many domains.

## 3.6 Conclusion

In order to establish a problem-solving context for conceptual retrieval, this chapter examines three distinct but related topics:

- AI techniques for the task of legal analysis,

- the representation of justifications, and
- the domain of captive insurance taxation.

The first of these three topics introduces many of the task features that characterize legal analysis and justification. The second describes approaches for representing the content and structure of justifications that can be used to support problem solving and information retrieval. Finally, the domain review delineates important concepts and lines of reasoning that typify reasoning about captive insurance cases.

The chapters that follow draw from these reviews in order to offer a theory of conceptual retrieval for justificatory reasoning. Chapter 4 expands on the discussion of legal classification, offering a full task analysis of legal justification and describing a problem-solving architecture for the task. Chapter 5 then expands on the discussion of representing justifications, offering a formalism for representing justificatory arguments that directly supports conceptual retrieval in the context of the problem-solving architecture from Chapter 4. Finally, Chapter 6 describes a case memory based on the formalism from Chapter 5, using the domain of captive insurance taxation as a testbed for the case memory methodology.

# CHAPTER 4

# A PROBLEM SOLVING ARCHITECTURE FOR LEGAL ANALYSIS

## 4.1 Introduction

The previous two chapters establish a context in which to consider the conceptual retrieval problem. Chapter 2 defines the problem and considers work in three disciplines aimed at solving it. From this work arises the unifying theme that memory can fruitfully be organized according to the roles that stored items might fill in problem solving. Chapter 3 then analyzes past work on the task of legal justification, in which conceptual retrieval plays an integral part. Also described in Chapter 3 is the taxation issue of captive insurance arrangements, a developing area of the tax law that serves as an interesting experimental testbed for the conceptual retrieval problem.

This chapter proposes a problem solving architecture, in the technical sense of Sticklen [1990], for the task of legal justification. In this architecture, Generic Task problem solvers are integrated with a case memory organized by the roles that past justifications can play in assembling future justifications. The remainder of the chapter consists of two parts:

- a task analysis [Chandrasekaran 1990] of legal justification, and
- a problem solving architecture derived from this task analysis.

Discussion in this chapter focuses on components of the architecture other than the case memory, which will be described more completely in following chapters. The focus here is on the task of legal justification and, in particular, the functional requirements that this process places on case memory.

## 4.2 A Task Analysis of Legal Justification

The task of justification poses a straightforward information processing problem: given the facts of a situation and an assertion about the situation, produce a justification for believing the assertion[1] (Figure 5). Ordinarily, one expects the justification to establish an inferential chain leading from (some subset of) the facts to the assertion. Such an inferential chain consists of a sequence of true assertions, each linked to its antecedents in the chain by a *warrant* [Toulmin 1958] that justifies its inference. In different domains of discourse, different types of warrant and backing are considered "valid", that is, as leading to acceptable conclusion. Methods of generating and evaluating justifications depend largely on the nature of warrants and their backing.

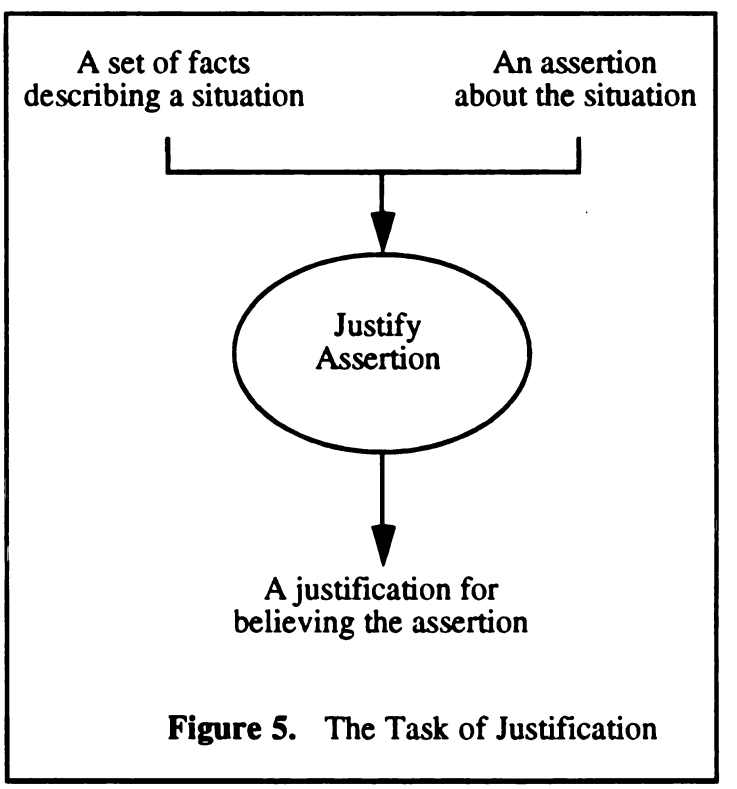


**Figure 5.** The Task of Justification

1 In practice, this task becomes one of creating multiple (and competing) justifications for the assertion [Ashley 1990, Branting 1991]. These justifications can then be evaluated and ranked by their persuasiveness. This project considers the specific task of generating and evaluating a single justification. If desired, knowledge of the type described by Ashley and Branting could be added to this analysis for consideration of the more general problem.

In the law, past cases serve as the primary warrants and backing for new justifications, and so the standard method for justification is based on analogy [Levi 1949, Ashley 1990]. This method sets up three subtasks: abstract the facts into appropriate legal terms, retrieve relevant cases from memory, and use the precedents to justify the target assertion (Figure 6). Ashley describes a control regime for this method that is essentially linear — abstract, retrieve, and apply — but typically these subgoals can be interleaved in complex ways. For example, in applying a precedent, the problem solver may find that a portion of the precedent's justification fails for the current situation. This difficulty can be addressed by retrieving another past case and using its justification.

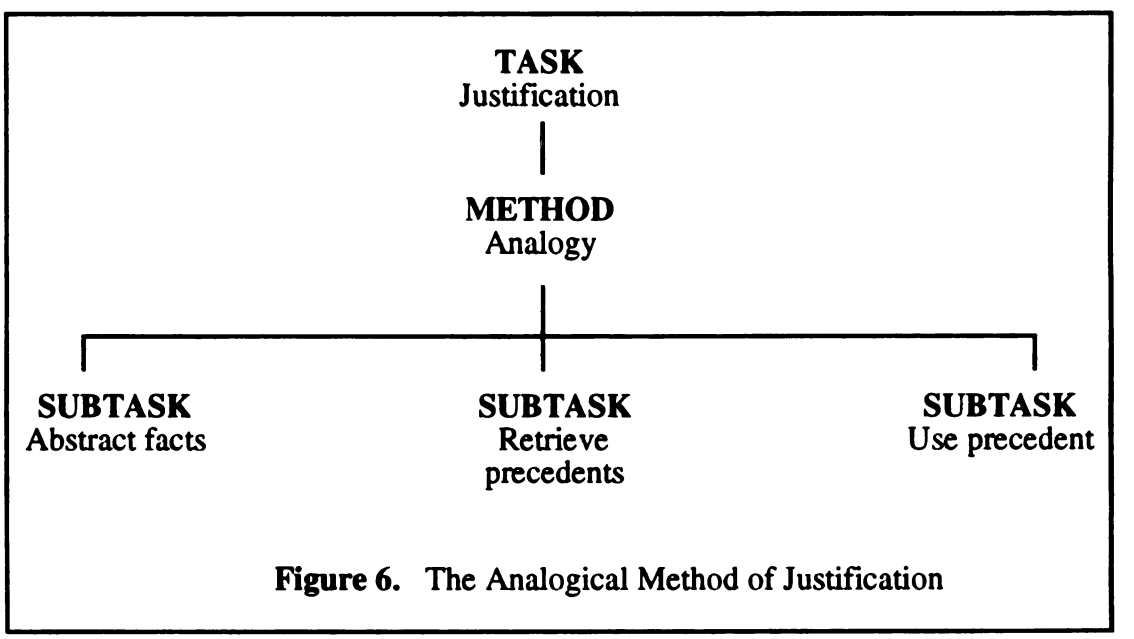


**Figure 6.** The Analogical Method of Justification

### 4.2.1 Fact Abstraction

The subtask of fact abstraction takes as input a set of data and gives as output one or more abstractions of the data. For the domain of tax law, these data correspond to facts describing a situation. Abstraction of facts is critical to the justification task. The power of

a justification lies in its ability to persuade. The appropriate use of a domain's technical vocabulary increases the precision and conciseness of the argument, thus making the justification more persuasive. Additionally, the retrieval of relevant cases may depend on the availability of important abstractions (Chapter 2).

One method for this task often used in the law involves matching patterns of facts to abstractions that characterize the patterns [Meldman 1977, McCarty 1977]. To employ this method, a problem solver must possess knowledge in the form of $D_1$, $D_2$, ... $D_n$ —> A rules, where the $D_i$s are data and A represents an abstract term. A variation on the method uses precedent cases in place of pattern match rules [Branting 1991]. In this variation, the problem solver retrieves past situations that share features with the current situation and notes the abstractions that characterized these precedents. Branting refers to this variation as "operationalizing" abstractions. This variation proves especially useful when a problem solver requires an abstraction in order to use some other knowledge but lacks the compiled pattern match rules needed to infer the abstraction.

### 4.2.2 The Subtask of Case Retrieval

The subtask of case retrieval can be characterized as follows: given a set of data, produce a set of relevant cases. In Chapter 2, several different views of this task are described and methods for performing it are presented. The input data can include facts, abstractions of the facts, or goals of the problem solver (such as an assertion to be justified). One family of methods for performing case retrieval relies on an associative memory that requires no search knowledge. However, the methods most studied in AI use explicit indices to organize memory. To use these methods, a problem solver requires knowledge for differentiating among (classes of) cases based on the features that characterize them.

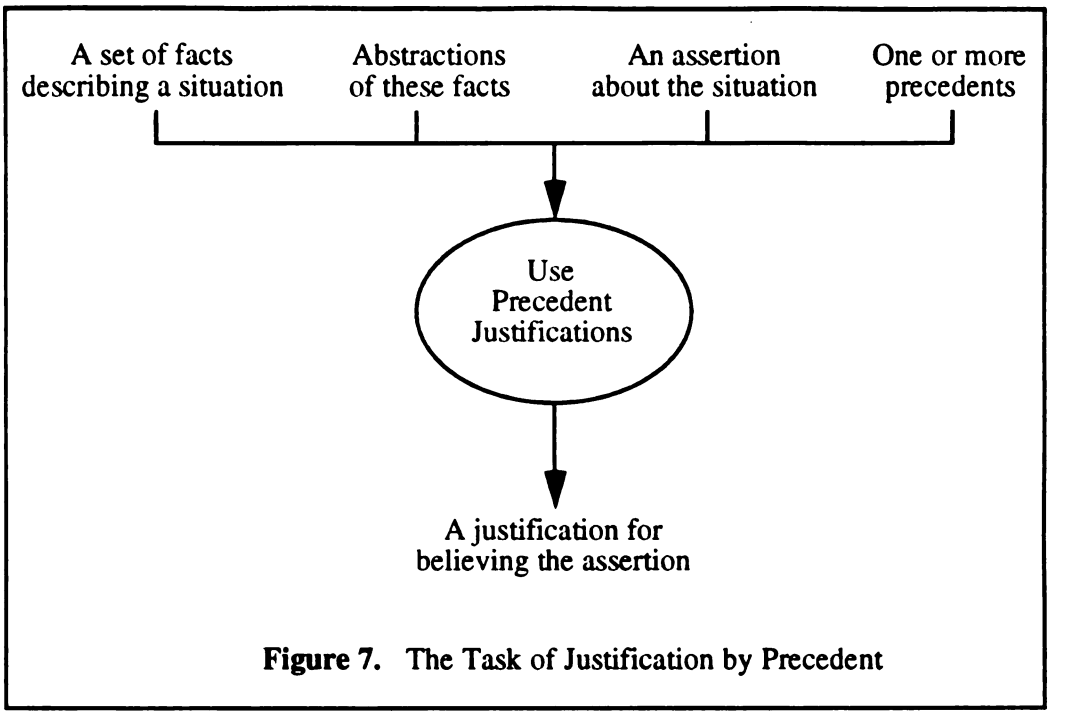


**Figure 7.** The Task of Justification by Precedent

### 4.2.3 The Subtask of Precedent Application

Finally, the subtask of using precedent justifications produces a justification for the target assertion (Figure 7). This subtask can be performed by adapting an input justification so that it applies to the current situation. Such a case adaptation method requires knowledge for evaluating justifications, determining the reason(s) for a justification's inadequacy, and modifying a justification to overcome its weaknesses. In the law, one might accomplish this modification subtask by applying another past justification as a "patch" to the inadequate portion of the justification being modified. Figure 8 depicts the full task analysis for legal justification, including possible methods and their subtasks.

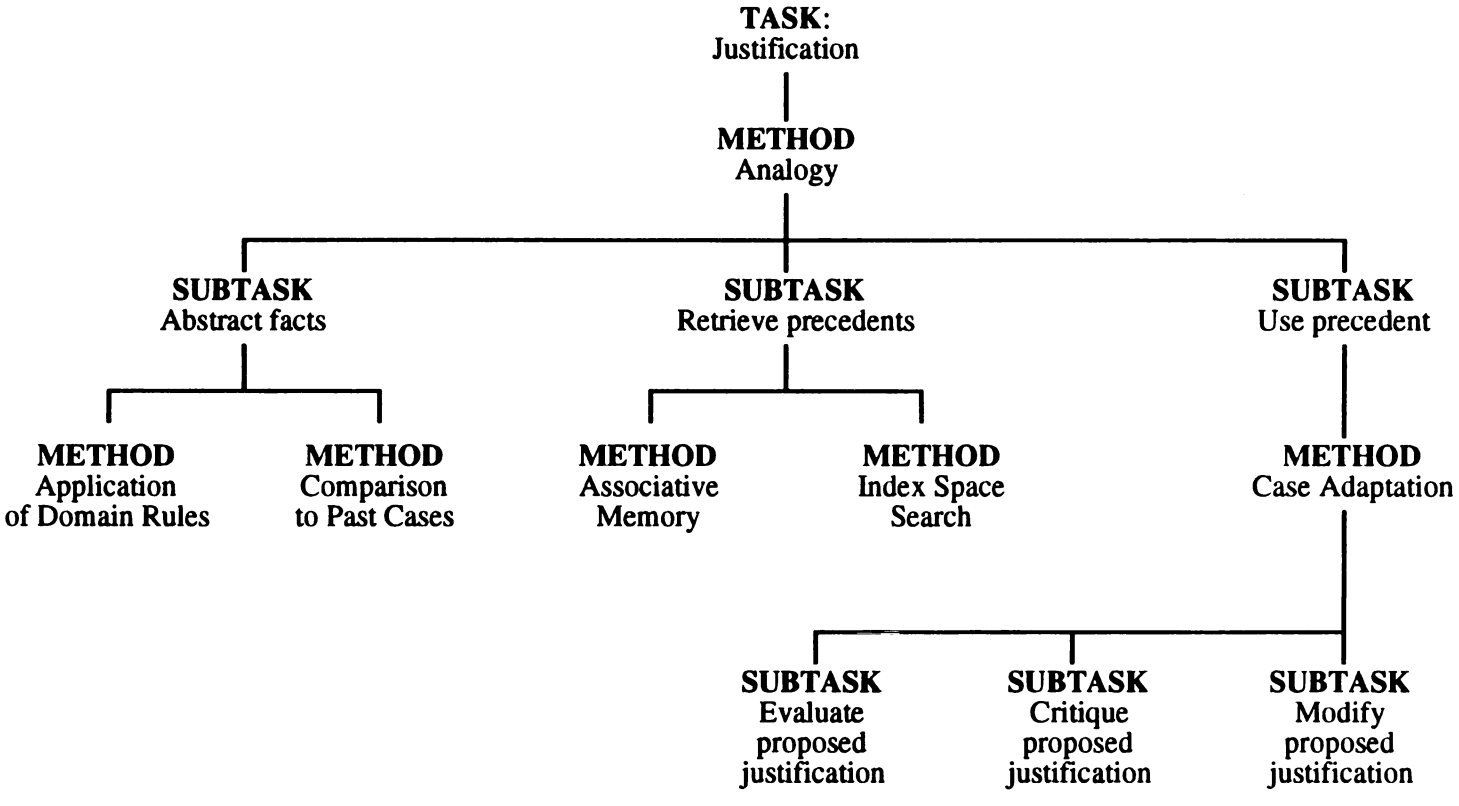


**Figure 8.** A Task Analysis of Legal Justification

### 4.2.4 A Control Strategy for Legal Justification

Given a task/method/subtask analysis of the justification task, one can develop a control strategy for generating justifications. Table 2 displays one such strategy, which generalizes the method proposed by Ashley [1990]. This control strategy does not give a complete picture of the problem-solving process, since each step involves further problem-solving. But specification of the high-level strategy provides some guidance in understanding legal justification, as well as guidance for building a knowledge-based system to perform the task.

**Table 2.** A Control Strategy for Legal Justification

1. Request assertion to be justified and initial data describing the situation.
2. Identify relevant legal abstractions of the data.
3. Identify relevant past justifications (cases).
4. Select the best match among the available cases, and propose it as a solution.
5. Evaluate the proposed solution. If it suffices to justify the assertion, then output an appropriate citation and halt.
6. Critique the proposed solution to identify individual inferences responsible for the justification's insufficiency.
7. For each inference identified, modify the inference chain so that it holds for the current situation.
8. Go to Step 5.

## 4.3 The Problem Solving Architecture

Following this task analysis, one can develop a problem solving architecture (PSA) for an agent that performs the task of legal justification. The description of the PSA given here follows the treatment of Sticklen [1990] and as such specifies three necessary features of the agent:

- the component problem solvers (sub-agents) which comprise the agent,
- the channels of communication among the sub-agents, and
- the problem-solving method employed by each.

These features are determined in large part by the decomposition of the analogy method presented in the preceding task analysis. They depend directly on its subtasks and the domain knowledge available for solving each.

Figure 9 presents a PSA for legal justification. The following sections describe the components of this architecture, the communication channels among them, and the problem-solving methods they employ. These task-specific agents are implemented as instances of Generic Task problem solvers. Detailed discussion of how the Case Memory performs its task appears in Chapters 6 and 7. For now, it is important to describe only the functional requirements that the legal justification architecture places on the conceptual memory component.

### 4.3.1 Components

The architecture consists of four sub-agents. The Justification Generator directs the activities of all the other agents and embodies the *use precedent to justify* subtask. The Fact Abstractor and Case Memory correspond to the two subtasks *abstract facts* and *retrieve precedents*, respectively. Finally, the Situation Data Base serves as a shared blackboard for the other agents. All facts and fact abstractions for the current situation reside in this database, which facilitates efficient communication among system components. Furthermore, the Situation Data Base manages all interaction between the system and the user.[2]

---

2 More generally, the Situation Data Base could manage interaction with other external sources as well, such as external data bases. See Chapter 9 for a brief discussion.

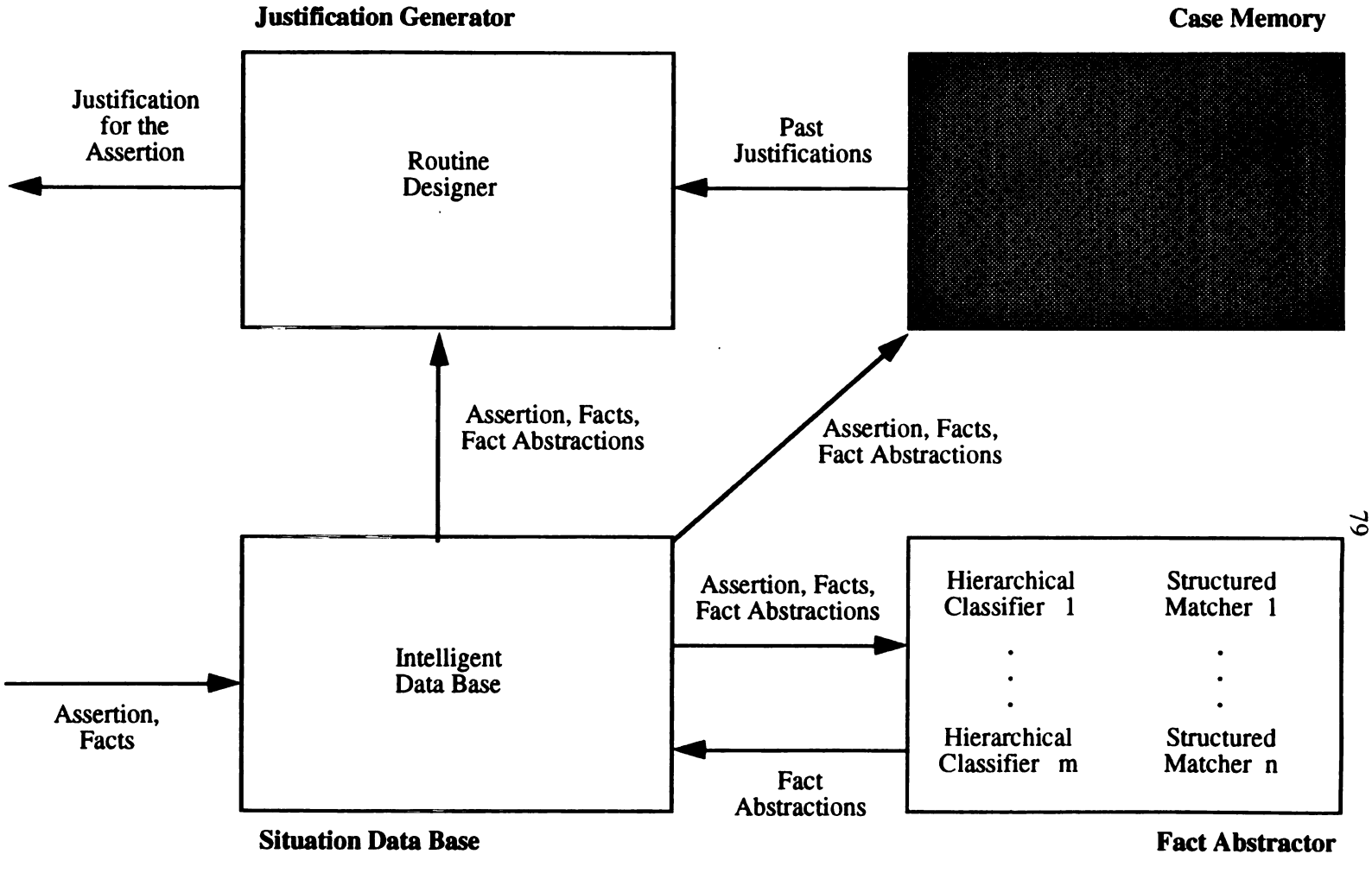


**Figure 9.** A Problem Solving Architecture for Legal Justification

### 4.3.2 Communication

Agents in the architecture communicate based on the input-output services each provides. Communication follows a message passing model: when an agent needs a piece of external data, it sends a message to the agent that can provide the data. Interactions among the agents are limited to the repertoire of messages each can handle, along fixed communication paths.

Table 3 depicts the communication channels present in the proposed PSA. Of special interest are the paths to and from the Case Memory. The Justification Generator sends requests to the Memory in the course of its problem solving. These requests may be for cases similar to a particular situation, or they may be for cases that can play a certain role in assembling a justification. In order to fill these requests, the Memory may need to refer to the facts of the current situation. A request is then sent to the Situation Data Base, which provides this information.

### 4.3.3 Problem Solving Methods

From the task analysis above, one can identify individual *generic tasks* [Chandrasekaran 1983, 1987] that play a role in legal justification. Each agent in the problem-solving architecture (Figure 9) can be viewed as composed of one or more Generic Task problem solvers. This section describes each PSA agent in terms of the generic method it employs and the knowledge it possesses.

Analysis of a variety of "real world" tasks and domains has led to the identification of what Chandrasekaran has termed generic tasks (GTs). Each GT is a problem-solving method that has been found useful for solving a particular task in a variety of domains. GTs have proven especially beneficial as models of problem-solving types because they explicitly delineate:

**Table 3.** Channels of Communication in the PSA

| | |
|---|---|
| **Justification Generator** | |
| **Input:** | Assertion, Facts, Fact Abstractions |
| **Requested From:** | Situation Data Base |
| **Input:** | Past Justifications |
| **Requested From:** | Case Memory |
| **Output:** | Justification for the Assertion |
| **Requested By:** | < System User > |
| **Fact Abstractor** | |
| **Input:** | Assertion, Facts, Fact Abstractions |
| **Requested From:** | Situation Data Base |
| **Output:** | Fact Abstractions |
| **Requested By:** | Situation Data Base |
| **Case Memory** | |
| **Input:** | Assertion, Facts, Fact Abstractions |
| **Requested From:** | Situation Data Base |
| **Output:** | Past Justifications |
| **Requested By:** | Justification Generator |
| **Situation Data Base** | |
| **Input:** | Assertion, Facts |
| **Requested From:** | < System User > |
| **Input:** | Fact Abstractions |
| **Requested From:** | Fact Abstractor |
| **Output:** | Assertion, Facts, Fact Abstractions |
| **Requested By:** | < All Three Problem Solvers > |

- the types of domain knowledge required to apply the method, and
- the nature of the control strategy used to perform the task.

These elements of a GT's definition constitutes an epistemic vocabulary for describing problem solvers independent of their implementation technology.

**Justification Generator** The top-level agent for generating new justifications performs the generic task of a *routine design* [Brown 1987, also with Chandrasekaran 1986]. This kind of problem solving develops when a designer works on a problem many times, each time with different but substantially similar requirements. In routine design, the designer decomposes the task into a number of subproblems and then solves each in turn. For each subproblem there exists a relatively small number of well-understood operators to apply.

A routine design problem solver consists of a hierarchy of design *specialists*, each concerned with a particular feature of the problem solver's output. The lowest-level specialist is the design *step*, which makes one design decision (such as selecting the value of a parameter). Aggregations of steps compose a design *task*, and a number of tasks constitute a design *plan*. A plan specialist embodies a method for producing a design or a part thereof. At any point in a plan or task, a *constraint* specialist may test the design to insure that it satisfies a necessary condition. If the constraint specialist identifies a problem, a *failure handler* is invoked to repair the design.

Viewing a justification as an artifact to be designed, the Justification Generator can be viewed as a routine designer.[3] The designer's top-level plan corresponds to the control strategy given in Table 2. Each operation in the strategy represents a task to be performed in generating a new justification. The first three tasks require no design knowledge *per se*; rather, they involve invocation of the Situation Data Base and Case Memory. Tasks 4

---

3 This is not to say that case-based justification is necessarily a routine process. The tasks of justification and argumentation may be quite non-routine — as in the creation of a novel analogy. In domains with some settled rules of law and large bodies of precedent, though, many new cases can be handled using routine knowledge. The approach outlined here addresses this more restricted situation.

through 7, though, embody the Generator's knowledge for creating new justifications from existing ones. This process may entail further requests to the Case Memory. When a new inference must be made, or when an existing inference in the proposed justification must be supported, the Generator asks the memory for another precedent that can fill this role.

Methods for selecting, evaluating, critiquing, and modifying justifications employ similar forms of knowledge across most legal domains. Chapter 3 reviews several computational methods for these tasks, in particular the work of Ashley [1990] and Branting [1989, 1991]. One common approach that can span the four subtasks involves the generation of *hypothetical* examples. A useful hypothetical might share all of the facts with some base situation except for one, or the value of a fact may differ slightly. Such hypotheticals can be used to evaluate a proposed justification (e.g., by testing a "slippery slope") or to critique a proposed justification (e.g., by pointing to a weak inference therein). Ashley explicates a set of rules for generating a hypothetical and another set for using them in justification.

**Fact Abstractor** A set of structured matchers and hierarchical classifiers constitutes the fact abstraction component of the architecture. The generic task of *structured matching* [Bylander, Johnson, and Goel 1991] addresses the problem of selecting one item from a small number of distinct alternatives. In this component, this task may involve selecting one abstraction from a set of abstractions or recognizing the presence of a single abstraction. In the course of making such a decision, the problem solver may partition its task into simpler decisions, make the simpler decisions, and then combine these "subdecisions" to make its assigned decision. This is the central motivation for structured matching.

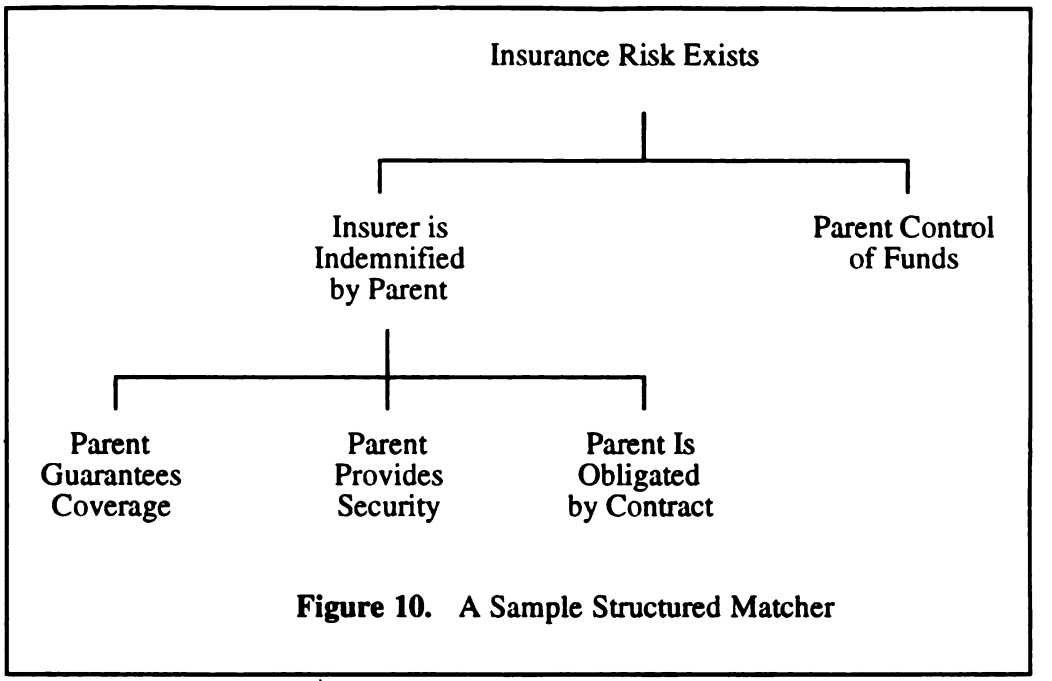


**Figure 10.** A Sample Structured Matcher

Figure 10 depicts a simple example from the domain of captive insurance law. This structured matcher determines whether the abstraction of insurance risk is present in the current situation. To make this decision, the matcher first makes decisions regarding whether the insurer is indemnified by the parent and whether the Parent retains control of the premium funds. It then uses these abstractions in deciding whether insurance risk exists. Each node in this hierarchy contains an ordered set of rules mapping patterns of data to decision alternatives. For leaf nodes, these patterns refer only to facts of the situation; for internal nodes, patterns may also refer to the decisions made by its subnodes. Partitioning patterns in this manner provides both computational efficiency and an explicit identification of subdecisions that are meaningful in the domain.[4]

---

4 Note also that such subdecisions may themselves represent meaningful fact abstractions in the domain. If so, their results can be made available to problem solvers other than the structured matcher (for instance, to the routine designer).

Some fact abstractions can be organized as a generalization hierarchy. In these cases, the Fact Abstractor employs the generic task *hierarchical classification* [Sticklen, Chandrasekaran, and Josephson 1987]. This GT selects one or more abstractions from the hierarchy of abstraction specialists using an *establish-refine* control strategy. When a specialist determines that its abstraction accurately describes the situation (that is, the specialist "establishes"), it attempts to refine the classification by allowing its subordinates in the hierarchy to attempt to establish themselves. Classification typically begins at the top of the hierarchy, with the most general abstraction, and then proceeds until one or more leaf-level abstractions have established.

Figure 11 depicts a hierarchical classifier from the captive insurance domain. The knowledge that each abstraction specialist uses in its attempt to establish can take a variety of forms. In the Fact Abstractor, though, the establishment task is achieved by structured matching over sets of pattern-mapping rules. For each specialist, there is a dedicated structured matching problem solver to recognize the presence of the specialist's abstraction. Such structured matchers and hierarchical classifiers allow the Fact Abstractor to identify abstractions along multiple dimensions of domain importance. They may be invoked either individually or *en masse* by request from the Situation Data Base. On an *en masse* request, the Fact Abstractor initiates the matching and classification agents in a pre-defined order.

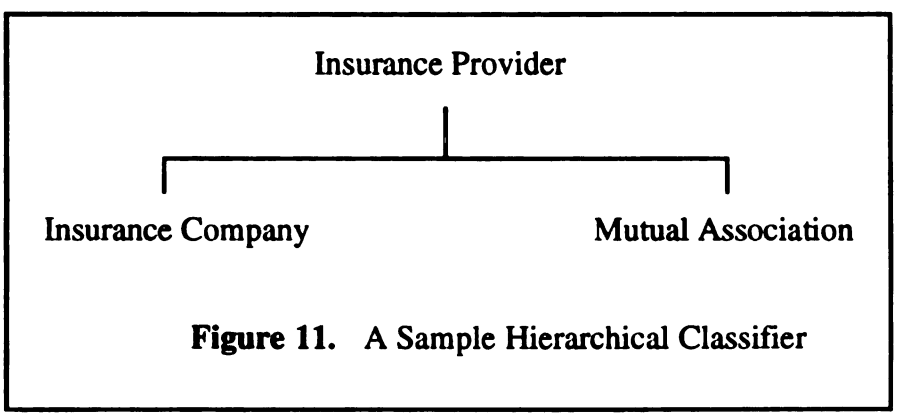


**Figure 11.** A Sample Hierarchical Classifier

**Situation Data Base** The Situation Data Base is an instance of the intelligent database generic task [Mittal, Chandrasekaran, and Sticklen 1984]. An intelligent data base manages situational, case-specific data using knowledge about data objects and relationships among them. The behaviors that an intelligent data base can be expected to produce include:

- the ability to obtain the value of a data object, whether from the system user or from some other source, and
- the ability to answer queries about data objects, retrieving either data values or abstractions on data values.

As conceived in the context of the problem solving architecture, the Situation Data Base provides only a subset of these capabilities. The Data Base maintains two types of data objects, facts and fact abstractions. For each fact, it has a slot for the fact's value and a method for obtaining a value from the user. For each fact abstraction, it has a slot for the abstraction's value and a pointer to the Fact Abstractor agent capable of determining its value. Thus, all abstraction behavior resides outside the data base, as does all representation of data relationships.

**Case Memory** A detailed description of the organization and problem-solving methods used by the Case Memory appears in Chapter 6.

## 4.4 Conclusion

A task analysis of legal justification leads to an appropriate problem solving architecture for generating justifications from precedent cases. Several of the subtasks established by this method, including data abstraction and routine design, are well understood in the law and in AI. Consequently, one can describe problem solvers that perform these subtasks using knowledge available in the domain.

Less well understood is the subtask of case retrieval. The PSA defines a task-specific context in which to perform retrieval, placing particular functional requirements on the case memory. Two open issues remain:

**How are justifications represented in memory?** This representation determines not only the content of the case memory but also the low-level detail of the routine designer that generates justifications.

**How is the case memory organized and searched?** In order to support conceptual retrieval, the case memory must be able to retrieve cases based not only on situation features but also on the role the case will play in problem solving.

These issues are closely related. Tax accountants and lawyers understand new justifications based largely on their relationship to past cases. Thus, a representation of justifications should take such relationships into account. These relationships also reflect the roles that past cases play in generating justifications, and so they should provide guidance for structuring case memory effectively. The next two chapters describe answers to these open questions that take advantage of the connection between case understanding and memory organization.

# CHAPTER 5

# A FUNCTIONAL REPRESENTATION OF JUSTIFICATORY ANALYSIS

## 5.1 Introduction

Tax accountants organize their understanding of legal justifications in a way that relates the justification to other justifications they already understand and to the accepted principles of tax accounting. This method of understanding results from the need to justify conclusions in terms of past decisions and existing statute. Thus, the use to which justifications are put affects not only how cases are organized in memory but also how cases are represented. One of the central contributions of this work is to show the intricate relationship between case representation and organization that follows from consideration of how justifications are used.

This chapter proposes a representation of justificatory analyses that is based on the Functional Representation (FR) of devices [Sembugamoorthy and Chandrasekaran 1986]. Extending the intuitions of Toulmin [1958], this representation captures the relationships among justifications used as warrants for drawing conclusions. The rest of the chapter includes:

- description of the motivations underlying this work, and
- specification of a functional representation for justificatory analyses.

The latter of these sections adapts and extends the vocabulary of the FR to the representation of legal cases that embody justifications. In the chapter's conclusion, this representation is related back to the issue of case organization.

## 5.2 Motivations

The initial motivation for this work came from the work of Toulmin [1958], a philosopher of science who questioned the utility of traditional logic as a tool for analyzing scientific argument. Toulmin developed a model of justificatory argument that explicitly identifies the roles that assertions can play in justifying a claim. This model is reminiscent of the FR, which expresses device behavior in terms of the mechanisms by which behavior is understood. The affinity between Toulmin's model and the FR has motivated this research aimed at understanding legal analyses as abstract devices.

### 5.2.1 Toulmin's Model of Argument

On Toulmin's view, logic deals not with techniques of inferring but rather with retrospective justification of claims. Traditional work in logic has focused on the types of proofs found in mathematics; Toulmin recognized that most arguments do not correspond to this notion of absolute proof. Instead, he set out to extend the classical syllogism to incorporate practical issues of justification. Abandoning mathematical proof, he adopted *jurisprudence* as the foundation of his model of logical argument. Jurisprudence stresses the persuasive nature of argument — "making the case" by providing appropriate evidence and citing the warrant that justifies inference.

This approach led Toulmin to greatly enrich the vocabulary available for describing arguments. Figure 12 reproduces the template for justifications he proposed. In this model, terms such as data, backing, and warrant characterize assertions as playing particular roles in the justification, in particular relationship to other assertions. The model extends the classical syllogism in at least two ways. First, its vocabulary more precisely explicates the roles filled by assertions (both explicitly stated and implicitly intended) in the justification. Second, this form of argument permits a homogeneous representation of different types of justifications, whether deductive, inductive, or abductive.

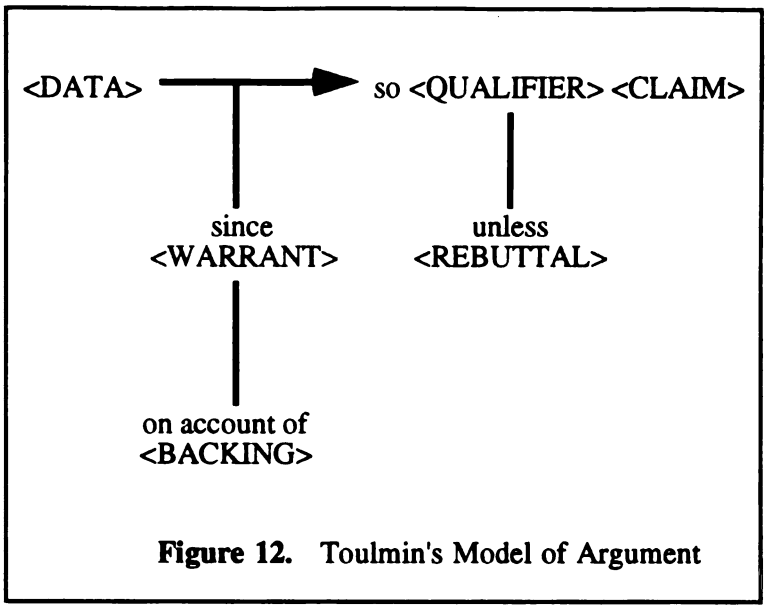


**Figure 12.** Toulmin's Model of Argument

Figure 13 depicts a typical argument in the classical syllogism and in Toulmin's representation. This example demonstrates how Toulmin's vocabulary enables one to state clearly the purpose of each assertion in the syllogism and the specific relationship among assertions. While the classical syllogism confounds the nature of the two premises, Toulmin's model shows that the former serves as an assumption and the latter as the warrant for the inference.

By representing justificatory roles unambiguously, the model facilitates development of criteria for judging the quality of justifications in different domains. For example, warrants express the force of a conclusion and are typically domain-independent. The warrant in Figure 13 could be used in any domain of discourse, from law to sociology. But the nature of the backing, which supports application of the warrant to the data, varies across domains. In the law, the backing must usually appeal to case precedent or statute, whereas in other domains category definition or statistical evidence are more appropriate.

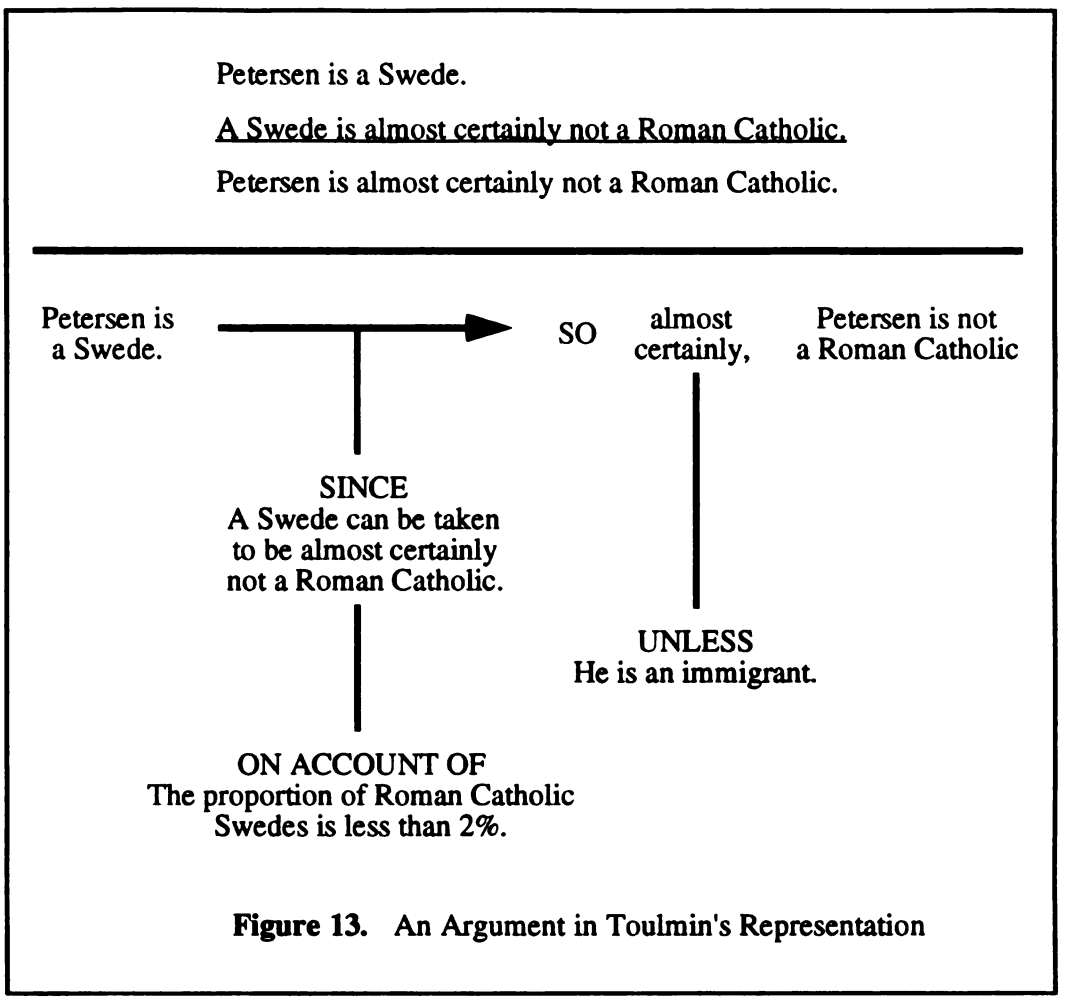


**Figure 13.** An Argument in Toulmin's Representation

### 5.2.2 Functional Device Understanding

Toulmin's model of justificatory argument calls to mind the Functional Representation of Sembugamoorthy and Chandrasekaran [1986]. The FR provides a language for describing devices based on their known functions or goals. A number of intuitions underlie the FR, including [Allemang 1990]:

- Devices have functions.[1]
- A device may consist of component subdevices, in which case the device achieves its functions by coordinating the functionality of its components.
- How a device achieves its functions is irrelevant to understanding its role as a component in another device.

In the FR, a device is decomposed into its components, whose own functions are then composed to achieve the functions of the device. The causal behaviors of each device or component are indexed according to the functions they realize. Since the functions of a device can be expressed in terms of component functions, the FR supports abstraction of behavioral detail across levels of the device decomposition. This feature allows the descriptions of a device and its components to be written in languages at different levels of abstraction.[2]

Figure 14 gives an example of the FR for a simple device, the ordinary household clothespin. The behavior **Open Arm** presents a sequence of causally related states that illustrates how the function **Open** is achieved. The label on each link refers to the mechanism understood to be responsible for the state transition, given that the preceding state was reached. These labels may point to basic knowledge of the world, such as one of Newton's laws, which lies outside the FR itself. However, they may also point to a function of some component of the device responsible for achieving the state change. To see a more detailed description of how the state change is achieved, one can refer to the behaviors implementing the component's function.

---

1 "This is taken as the defining characteristic of a device; a device is something which has a known function." [Allemang 1990, page 2]

2 For example, one could describe a transistor radio in terms of transistor circuits, radio signals, or tuning frequencies. Each level of description would be appropriate under different circumstances.

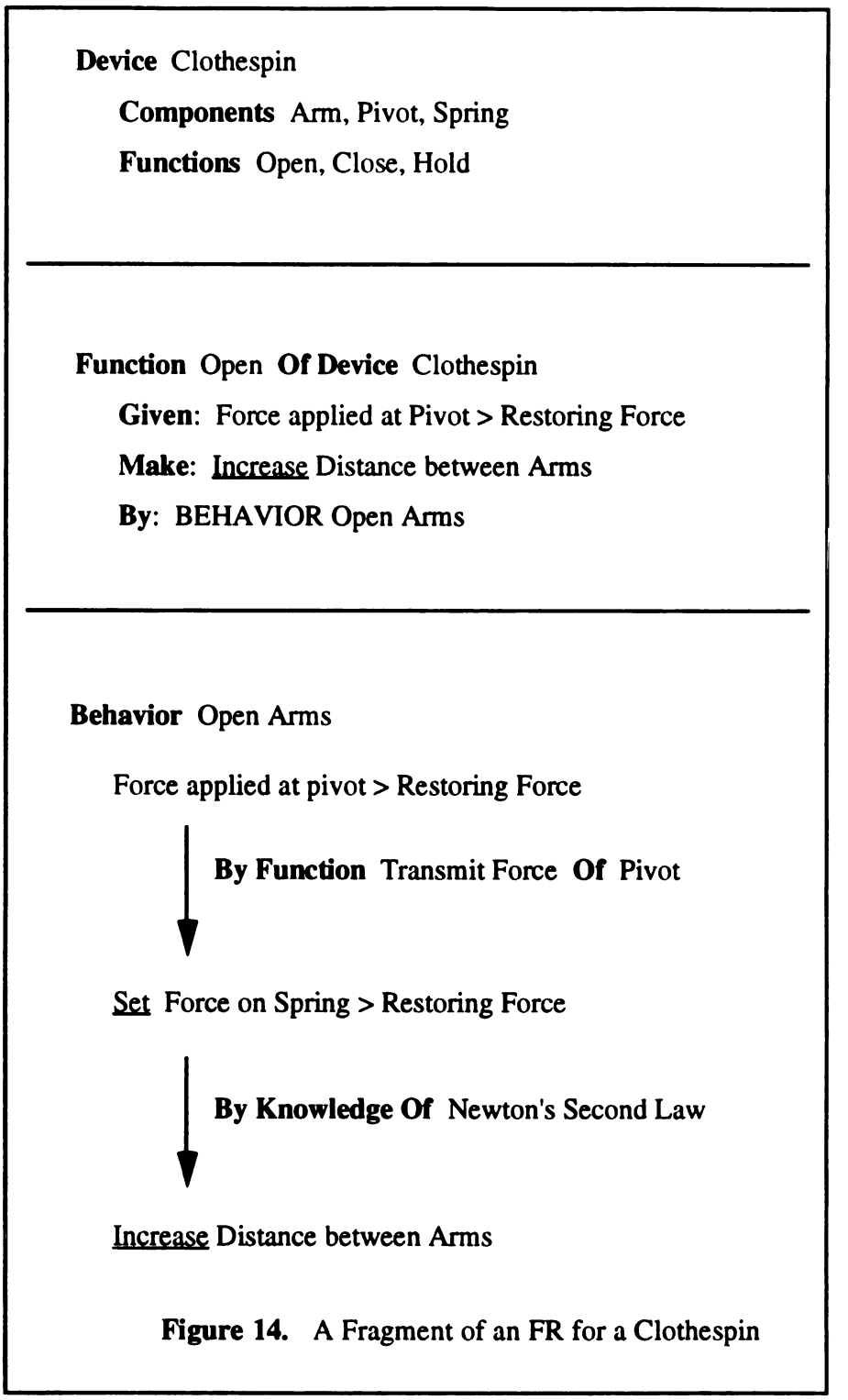


**Figure 14.** A Fragment of an FR for a Clothespin

The FR was originally intended for describing physical devices that had been designed with particular functions in mind [Sembugamoorthy and Chandrasekaran 1986]. Later work demonstrated its utility for describing other physical systems, such as human body physiology [Sticklen 1987], to which one can ascribe functional characteristics. These uses of the FR are teleological in this sense: functions correspond to behaviors that either

- support a designer's goals, or
- offer some analytic advantage in understanding a system.

Adopting this sense of teleology, researchers have extended the FR for use in describing abstract devices such as computer programs [Allemang 1990] and ecological systems [Sticklen and Tufankji 1992].

### 5.2.3 Viewing a Legal Case as a Device

One can view a legal case as an abstract device with the function of supporting a claim, given an initial set of assertions. Different parts of the case play specific roles, such as to rebut a counterargument or to propose a hypothetical situation, that in concert achieve the case's primary goal(s). Considered this way, a legal case can be modeled using the FR in a way strongly reminiscent of Toulmin's view of justification. The graph used by Toulmin to depict a justification corresponds to a behavior in the FR, a "causal" line of reasoning that delineates inference relationships among assertions.

Warrants correspond to link annotations in the FR. They may appeal to knowledge of the world or to empirical data, as Toulmin's backings do, or they may refer to the function of another case capable of supporting the inference link. Such layering of justifications, though not discussed explicitly by Toulmin, reflects the spirit of justification that he sought to capture: Claims that are challenged can be supported by appeal to more detailed arguments. This intuition of the affinity between Toulmin's ideas and the FR

motivates the representation proposed here. Table 4 shows the basis for a mapping between Toulmin's terminology and that of the FR.

**Table 4.** Viewing a Legal as a Device in the FR

| FR Terminology | Toulmin's Terminology |
|---|---|
| Device | ( Legal case ) |
| Function | ( Legal issue ) |
| Behavior | Justification |
| Preconditions | Data |
| Link annotation | Warrant |
| < None > | Backing |
| Postconditions | Claim |

## 5.3 Representing Legal Analysis in the FR

In the domain of taxation law, justifications reside within the bodies of legal cases. Each case includes not only the justifying chain of inference but also contextual information that defines the situation to which the justification applies. Thus, an adequate portrayal of legal justifications should account for both the case context and the chain of inference. This section describes a language for representing legal justifications, based on the Functional Representation, that accounts for both.

### 5.3.1 The Legal Case as a Device

A case corresponds to a device in the FR (Table 5). Case descriptions consist of three elements:

- an identifier,
- a case context description, and

- one or more issue descriptions.

The identifier serves as the name by which other devices may refer to the case. The case context description includes information necessary for understanding and reasoning about a case in the domain. Finally, the case is characterized by the issues that it addresses. These issues identify the functions that the case might play in analysis. Both the case context and issue descriptions are discussed in greater detail below. But first, several device-level issues must be addressed.

**Table 5.** Language Grammar: A Legal Case

<CASE> ::= **Case** *identifier*
<CASE CONTEXT>
<CASE ISSUES>

*Selection of Case Functions* Since the FR is a language for *describing*, not defining, devices, every possible function of the case need not be included in its description. One would initially include only those functions intended for the case by the justification's designer. In legal domains, the designer is typically the attorney who prepares an argument for the court or the judge who writes the court's opinion for a case. These functions will correspond to the significant legal issues raised by the case that the court must resolve. Later, if the body of case law evolves in such a way that the case is used for some other purpose, then other functions can be added to the case's description.[3]

3 This would almost certainly be done by a legally-trained system user, not by the system itself. At this point, no provision has been made for the memory to learn new concepts and classify instances of them.

Such a situation occurred in the case of Helvering versus Le Gierse [1941][4]. The Le Gierse case dealt with a claim that the proceeds from a life insurance policy should be excluded from the decedent's estate, as prescribed by law. On this issue, the court held that the proceeds must be included in the estate, because the insurance policy was accompanied by an annuity contract that essentially negated the risk inherent in insurance. The court's justification thus relied on an inference that, in Le Gierse's circumstances, true insurance did not exist. As the case law surrounding issues of insurance grew, this argument came to be used as a precedent in many cases involving the existence of insurance. This justification now constitutes one of the more important functions of Helvering versus Le Gierse in tax law.

*Treatment of Multiple Opinions* In some situations, different judges hearing a case will reach different conclusions regarding an issue. Similarly, two judges may reach the same conclusion but differ significantly in their lines of reasoning. The result is multiple opinions for the case[5]. In this representation, each opinion that offers a different line of reasoning constitutes a separate function of the case. This feature reflects the different roles a case may play as a precedent in later justifications, based on which opinion is cited. Only the majority opinion is controlling, in the sense that future decisions are bound by its reasoning, but the other opinions can often be used in particular contexts as persuasive evidence.

---

4 All case references appear as a separate bibliographic listing, Appendix A.

5 Cases may contain three types of opinion. The majority opinion presents the court's holding and its justification for this conclusion. One or more concurring opinions agree with the majority holding but present different lines of reasoning for reaching the conclusion. Finally, one or more dissenting opinions may disagree with the majority's holding and offer a line of reasoning to justify a different conclusion.

**Table 6.** Language Grammar: Case Context

<CASE CONTEXT> ::=

**Context**

**Legal Documentation**
**Plaintiff** *plaintiff name*
**Defendant** *defendant name*
**Citation** *official citation*
**Date** *date of decision*

**Procedural Context**
**Setting** *court hearing the case*
*location of decision in appellate chain*
**Outcome** *decision*
*relationship to prior/later cases*

**Facts**
<ASSERTION> {<ASSERTION>}*

### 5.3.2 Case Context as Device Annotation

The case context description includes background information that characterizes the situation at issue. This information consists in three parts (Table 6): legal documentation, the procedural context, and the facts of the situation. Legal documentation essentially defines the case for reference in the legal literature. Of greater semantic importance is the procedural context frame. The case setting determines the *pedigree* of the decision, its importance as cited evidence in different venues. Naturally, the outcome of the case and its relationship to other cases in the appellate chain will also have a major impact on when and how the case is useful as backing.

The facts of the case define the state of the world in which a justification is offered. These facts may include only those held to be relevant by the court, but more generally they will include other assertions as well. Facts stipulated by both parties and facts presented by

one party as relevant to the case can often provide a more complete context for creating and evaluating a justification. Table 7 presents a sample case context frame, for the case of Humana, Inc., versus the Commissioner of Internal Revenue [1989]. The language for representing facts is discussed in more detail in the following two sections.

### 5.3.3 Legal Issues as Function Identifiers

Just as devices in the FR are characterized by their functions, cases in the law are characterized by the issues they address. Each case addresses at least one legal issue (Table 8). The issue can be expressed as a legal question, raised either by a party to the case or by the court, to be resolved. For each issue, the court declares a holding, an answer to the question in the context of relevant case facts. Such a holding will generally be affirmative (for the plaintiff) or negative (for the defendant). The question and holding define the endpoint to which the court's analysis must aim — a justification of the holding with respect to the question. In this representation, this corresponds to a function of the case.

A function is specified as a precondition/postcondition pair (Table 9). One or more preconditions serve as the assumptions upon which the justification is based. The postcondition is the claim made by the court, composed from the legal question and holding that define the issue. In this sense, the issue identifies a function of the case and corresponds to the postcondition of the function. The **By** slot points to a justification (as described below) that conveys the chain of inference which Concludes the claim **Given** the assumptive facts of the case.[6]

---

6 Other expressions of the Functional Representation [Sembugamoorthy and Chandrasekaran 1986, Allemang 1990] have also allowed an optional **Provided** slot. This slot would indicate a state or predicate that must hold in order for the function to be "meaningful." Such a predicate can be thought of as a special precondition for the function, not directly involved in the causal behavior realizing the function, that defines a relevant context for the function. The **Provided** slot could fruitfully be added to the representation outlined here. (See the discussion of this topic in Chapter 8.)

**Table 7.** A Sample Case Context Frame — Humana [1989]

**Case** Humana-89

**Context**

**Legal Documentation**

| | |
|---|---|
| **Plaintiff** | Humana, Inc. |
| **Defendant** | Commissioner of Internal Revenue |
| **Citation** | 89-2 USTC ¶9453 |
| **Date** | July 7, 1989 |

**Procedural Context**

| | |
|---|---|
| **Setting** | U.S. Court of Appeals. Plaintiff appeals ruling of the Tax Court that none of its intra-family insurance premiums are tax-deductible as business expenses. |
| **Outcome** | Affirms, reverses, and remands 88 TC 197, in parts. |

**Facts**

Plaintiff unable to obtain liability insurance at fair price.
Plaintiff incorporated wholly-owned insurance subsidiary.
Subsidiary meets all statutory and regulatory requirements.
Subsidiary provides insurance to plaintiff and its affiliates.
Subsidiary was fully capitalized at creation.
Plaintiff deducted all intra-family premiums as ordinary and necessary business expense.

**Table 8.** Language Grammar: Case Issues

<CASE ISSUES> ::= **Issues** <ISSUE> { <ISSUE> }*

<ISSUE> ::=

**Question** *legal question raised by case*
**Holding** *court's answer to the question*

<FUNCTION>

**Table 9.** Language Grammar: Function

<FUNCTION> ::=

**Function** *identifier*
**Given** <ASSERTION> { <ASSERTION> }*
**Conclude** <ASSERTION>
**By** <JUSTIFICATION>

<ASSERTION> ::= *predicate from the domain*

**Table 10.** A Sample Issues Frame — Humana [1989]

**Question** Are payments made by an entity to its wholly-owned subsidiary deductible as insurance premiums?
**Holding** No.
**Function** Classify Parent Payments
**Given** Parent pays premium to a subsidiary.
**Conclude** Payment is not deductible.
**By** Conclude No Parent Insurance

**Question** Are payments made by an entity to a sibling deductible as insurance premiums?
**Holding** Yes.
**Function** Classify Subsidiary Payments
**Given** Sibling pays premium to a sibling.
**Conclude** Payment is deductible.
**By** Conclude Sibling Insurance

The assertions that fill the **Given** and **Conclude** slots are specified in a state language that is independent of the FR. Here, assertions consist of domain predicates taken from the semantics of tax accounting. These predicates may originate in descriptions of factual situations in the domain or as abstractions of fact patterns. As a result, the language for expressing assertions in the issue description of a case is also used to express the facts of the case in its context description. An example of an issue description, again from the Humana case, appears in Table 10.

### 5.3.4 Justifications as Behaviors

Each function points to the justification that supports its claim. The justification (Table 11) embodies the chain of inference connecting case facts to the holding of the court. That is, the function specifies a claim made by the case as an input/output relation, and the justification specifies the reasoning that justifies the claim. In this representation, the justification also carries a slot for additional commentary, to allow further textual explanation of the justification. This commentary may refer to case citations not directly part of the inference chain[7] or to comments explaining the justification.

The justification definition corresponds to a behavior in the FR. A behavior is described as an annotated directed graph with assertions serving as nodes. Each link in the graph carries an annotation that indicates the warrant for the inference. As described in Table 11, the graph must constitute a chain, an alternating sequence of assertions and warrants. In general, though, each step in the inference may draw on additional preconditions.[8]

---

7 Such as related or distinguished cases. See discussion of Ashley in Chapter 3.

8 Allemang [1990] has also extended the notion of a behavior to include simple cycles and multiple-link edges between nodes.

**Table 11.** Language Grammar: Justification

<JUSTIFICATION> ::=

**Justification** *identifier*
**Comment** *explanatory text*
**Definition**
<ASSERTION> { <WARRANT> <ASSERTION> }*

A warrant carries one of three possible annotations (Table 12). A **By Knowledge** warrant refers to knowledge outside of the Functional Representation. In the domain of taxation, such knowledge could be of an accepted economic definition or of a statute.[9] A **By Function** warrant cites the function of another case as the backing for an inference. This is the central feature of the representation for legal analyses described here: the ability to cite particular issues and justifications from other cases in the domain as backing for inferences in later cases. Finally, a **By Justification** warrant indicates that the detailed inference between two assertions is suppressed in the current graph. The inference is described in a separate justification. This annotation corresponds to a *By Behavior* link in the FR.

Figure 15 depicts a simple example of a justification represented as a behavior. One additional element of the representation remains, the notion of a hypothetical. A hypothetical is an argument or justification that invokes a conjecture: what if a certain circumstance arises? Hypotheticals comprise an important class of arguments in the law, since they enable exploration of the reasonableness of a claim in potential scenarios. In this

9 Of course, such knowledge may itself be describable using the FR. This is especially true of statutes in taxation law. One might represent a statute as a device, with the function of supporting a particular class of claims. The **BY KNOWLEDGE** link could then be replaced with a **BY FUNCTION ... OF STATUTE ...** link. This possibility is not pursued further here.

**Table 12.** Language Grammar: Warrants

<WARRANT> ::=

**By Knowledge Of** *identifier* |
**By Function** *identifier1* **Of Case** *identifier2* |
**By Justification** *identifier*

**Justification** Conclude No Parent Insurance

**Comment:** This justification relies on the hypothetical scenario proposed in Carnation to argue that no parent can insure with its own subsidiary.

**Definition**

Parent pays premium to a subsidiary.

**By Function** Conclude Risk Not Shifted
**Of Case** Clougherty-87

Risk is not shifted.

**By Justification** Define Insurance Standard

Payment is not deductible.

**Figure 15.** A Sample Justification — Humana [1989]

representation, hypothetical "facts" posed as preconditions in a justification are distinguished as hypothetical. This distinction alerts users of the justification that this fact need not hold in a situation in order for the justification as a whole to hold.[10]

### 5.3.5 A Complete Example of the Representation

Table 13 outlines the full language for representing a legal analysis. This language enables one to depict an analysis and its justification in a way that captures domain relationships among elements of the analysis and among different cases. Consider the case of Humana, Inc., versus the Commissioner of Internal Revenue [1989]. In *Humana*, the U.S. Court of Appeals had to resolve whether payments Humana and its subsidiaries made to a wholly-owned insurance subsidiary were legally deductible as insurance premiums. On the issue of payments made by Humana itself, embodied in the function Classify Parent Payments, the court justified its holding that the payments were not deductible by citing both the tax code and a precedent case. These citations provided evidence for particular steps in the court's reasoning. Table 14 shows the case and issue descriptions for *Humana*, while Figures 15 and 16 present the line of reasoning used in resolving the issue of parent payments.

---

10 The next section includes an example of a hypothetical.

**Table 13.** A Language for Representing Legal Analysis

<CASE> ::= **Case** *identifier* <CASE CONTEXT> <CASE ISSUES>

<CASE CONTEXT> ::=
**Context**
**Legal Documentation**
**Plaintiff** *plaintiff name*
**Defendant** *defendant name*
**Citation** *official citation*
**Date** *date of decision*
**Procedural Context**
**Setting** *court hearing the case*
*location in appellate chain*
**Outcome** *decision*
*relationship to prior/later cases*
**Facts** <ASSERTION> { <ASSERTION> }*

<CASE ISSUES> ::= **Issues** <ISSUE> { <ISSUE>* }

<ISSUE> ::= **Question** *legal question raised by case*
**Holding** *court's answer to the question*
<FUNCTION>

<FUNCTION> ::=
**Function** *identifier*
**Given** <ASSERTION> { <ASSERTION> }*
**Conclude** <ASSERTION>
**By** <JUSTIFICATION>

<ASSERTION> ::= *predicate from the domain*

<JUSTIFICATION> ::=
**Justification** *identifier*
**Comment** *explanatory text*
**Definition** <ASSERTION> {<WARRANT><ASSERTION>}*

<WARRANT> :=
**By Knowledge Of** *identifier* |
**By Function** *identifier1* **Of Case** *identifier2* |
**By Justification** *identifier*

**Justification** Define Insurance Standard

**Comment:** This justification adopts the definitional test of Helvering versus Le Gierse [1941] as the legal standard for the existence of insurance.

**Definition**

Risk is shifted & Risk is distributed

↓ **By Function** Define Insurance
**Of Case** LeGierse-41

Insurance exists.

↓ **By Knowledge Of**
Case Data (Payments in question)

Payment is an insurance premium.

↓ **By Knowledge Of**
Tax Regulation §1.162-1(a) — 1954

Payment is an ordinary and necessary business expense.

↓ **By Knowledge Of**
Internal Revenue Code §162(a) — 1954

Payment is deductible.

**Figure 16.** The Justification Define Insurance Standard — Humana [1989]

**Table 14.** A Complete Case Description — Humana [1989]

Case Humana-89

**Context**

**Legal Documentation**

| | |
|---|---|
| **Plaintiff** | Humana, Inc. |
| **Defendant** | Commissioner of Internal Revenue |
| **Citation** | 89-2 USTC ¶9453 |
| **Date** | July 7, 1989 |

**Procedural Context**

| | |
|---|---|
| **Setting** | U.S. Court of Appeals.<br>Plaintiff appeals ruling of the Tax Court that none of its intra-family insurance premiums are tax-deductible as business expenses. |
| **Outcome** | Affirms, reverses, and remands<br>88 TC 197, in parts. |

**Facts**

Plaintiff unable to obtain liability insurance at fair price.
Plaintiff incorporated wholly-owned insurance subsidiary.
Subsidiary meets all statutory and regulatory requirements.
Subsidiary provides insurance to plaintiff and its affiliates.
Subsidiary was fully capitalized at creation.
Plaintiff deducted all intra-family premiums as ordinary and necessary business expense.

**Issues**

| | |
|---|---|
| **Question** | Are payments made by an entity to its wholly-owned subsidiary deductible as insurance premiums? |
| **Holding** | No. |
| **Function** | Classify Parent Payments |
| **Given** | Parent pays premium to a subsidiary. |
| **Conclude** | Payment is not deductible. |
| **By** | Conclude No Parent Insurance |

| | |
|---|---|
| **Question** | Are payments made by an entity to a sibling deductible as insurance premiums? |
| **Holding** | Yes. |
| **Function** | Classify Subsidiary Payments |
| **Given** | Sibling pays premium to a sibling. |
| **Conclude** | Payment is deductible. |
| **By** | Conclude Sibling Insurance |

In order to examine in greater detail why the court concluded that no insurance existed, one can refer to the case cited in support of this conclusion, Clougherty versus the Commissioner of Internal Revenue [1981]. A portion of the FR for *Clougherty* appears in Figure 17. The Court applied a line of reasoning that proposed a hypothetical scenario: suppose that the parent suffered an insured loss. Analysis of the resulting reimbursement indicates that the risk of loss has not been shifted from the "insured" party. According to the legal precedent of Le Gierse versus Helvering [1941], risk shifting is a necessary condition for insurance to exist. Hence, the conclusion that true insurance does not exist is justified by precedent.

Again, if desired, one could examine the backing for this conclusion in greater detail by referring to the justification offered in *Le Gierse*. This justification "bottoms out" in the sense that all of its backings refer to working knowledge of the business domain and not to prior arguments. The function of *Le Gierse* cited in *Clougherty* is an example of a function ascribed retrospectively to a case based on its use in justification. This function would likely not have been part of an FR description of *Le Gierse* in 1941; only subsequent use of the case as a precedent in insurance cases would lead one to consider defining insurance to be one of *Le Gierse*'s functions.

## 5.4 Conclusion

This chapter presents a functional representation for justificatory analysis, motivated by the work of Toulmin. In doing so, new emphasis is placed on particular elements of the influencing representations.

*Functional Representation* The FR is applied to a new sort of abstract device, a justificatory analysis. Each such device can be considered as a component of a top-level device that corresponds to the body of case law as a whole. This corpus of cases consists

**Case** Clougherty-87

**Function** Conclude Risk Not Shifted
**Given:** Parent pays premium to a subsidiary.
**Conclude:** No insurance exists.
**By :** Hypothetical Loss Scenario

**Justification** Hypothetical Loss Scenario

**Comment:** This justification adopts the definitional test of Helvering versus Le Gierse [1941] as the legal standard for the existence of insurance.

**Definition**

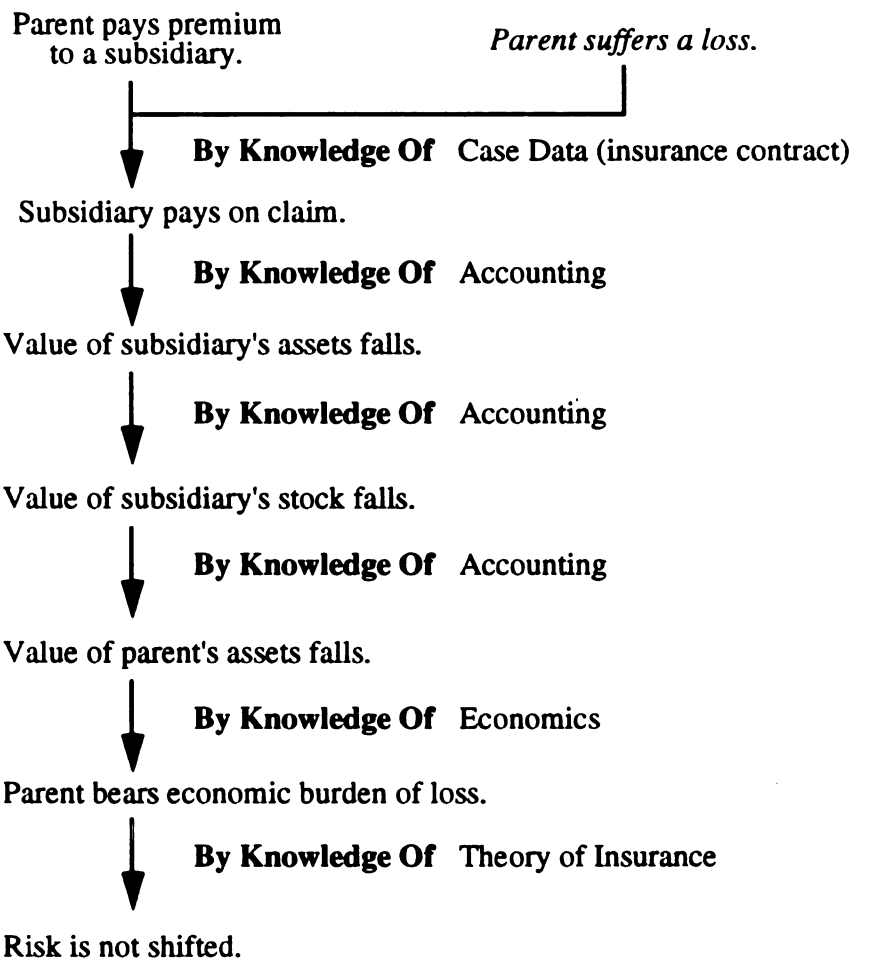


**Figure 17.** A Portion of the FR for the Clougherty Case

of individual analyses combined to support claims in the domain. The functions of the device comprise all the functions of its constituent devices. This type of analysis is possible precisely because cases have roles (functions) in justifying the reasoning in other cases.

Table 15 shows the mapping between the traditional FR and the FR for justifications described here. In this use of the FR, a behavior (justification) does not consist of a causal chain of device states but rather denotes an inferential chain of assertions about the situation being described. Each level of description — device, function, and behavior — carries additional commentary that explains the analysis and the context to which it applies.

*Toulmin's Model* The citation of past cases as backing for a new inference elaborates Toulmin's notion of a warrant. Each inference can invoke a prior justification as evidence to back its conclusion, until reaching the point at which a justification relies on assumed

**Table 15.** The FR for Legal Justifications

| The FR for Legal Justifications | Traditional FR |
|---|---|
| Case | Device |
| Issue | Component |
| Function | Function |
| Justification | Behavior |
| Assertion | State |
| Assumptions | Preconditions |
| Claim | Postcondition |
| Warrant | Link annotation |
| By Knowledge pointers | By Knowledge |
| By Function pointers | By Function |
| By Justification pointers | By Behavior |

domain knowledge. By casting the model in computational terms, the representation described here provides a meaningful way to integrate justifications — according to the functions they play. Use of the FR also explicates how assertions at different levels of abstraction relate to one another. Justifications at different levels can be related, again, according to the functions they play (that is, according to their pre-/post-condition specifications). Table 16 shows the mapping of Toulmin's terminology to elements of the new functional representation.

By indexing justifications according to the issues they address, this representation offers the starting point for a case indexing methodology. The use of case citation links in justifications comprises the most direct form of indexing, from case to case. The hope embodied in this representation, though, is that such indexing can be generalized to the body of case law as a whole, based on relationships among issues in the domain. This generalization constitutes the topic of the next chapter.

**Table 16.** The FR for Legal Justifications and Toulmin

| The FR for Legal Justifications | Toulmin's Terminology |
|---|---|
| Case | < Legal case > |
| Issue | < Legal issue > |
| Function | < None > |
| Justification | Argument or justification |
| Assertion | Assertion |
| Assumptions | Data |
| Claim | Claim |
| Warrant | Warrant |
| By Knowledge pointers | Backing |
| By Function pointers | < None > |
| By Justification pointers | < None > |

# CHAPTER 6

# A CONCEPTUAL MEMORY OF JUSTIFICATIONS

## 6.1 Introduction

In Chapter 4, a problem-solving architecture for justificatory legal analysis was presented. One component of this architecture, the case memory, provides access to past justifications for use in the course of constructing new classifications. The remaining elements of the architecture specify the functional requirements placed on case memory. Chapter 5 offers a functional representation for justifications that creates the possibility of conceptual retrieval based on the *roles* that justifications can play in problem solving. By applying the principles behind the Functional Representation, one can conceive of an organization of case memory that promotes focused access to cases based on the inferences that they make and support.

This chapter proposes a model of conceptual retrieval that is motivated by the functional representation of justifications. The model capitalizes on the idea of a functional decomposition to organize the memory of cases according to the issues for which they are relevant in justifying future cases. The remainder of the chapter describes:

- the index vocabulary for the model,
- a particular index organization, called an *issue composition hierarchy*, that relates functional indices and partitions case memory, and
- the retrieval algorithm by which the memory is searched for relevant cases.

**Table 17.** A Language for Representing Legal Analysis

<CASE> ::= **Case** *identifier* <CASE CONTEXT> <CASE ISSUES>

<CASE CONTEXT> ::=
**Context**
**Legal Documentation**
**Plaintiff** *plaintiff name*
**Defendant** *defendant name*
**Citation** *official citation*
**Date** *date of decision*
**Procedural Context**
**Setting** *court hearing the case*
*location in appellate chain*
**Outcome** *decision*
*relationship to prior/later cases*
**Facts** <ASSERTION> { <ASSERTION> }*

<CASE ISSUES> ::= **Issues** <ISSUE> { <ISSUE> }*

<ISSUE> ::= **Question** *legal question raised by case*
**Holding** *court's answer to the question*
<FUNCTION>

<FUNCTION> ::=
**Function** *identifier*
**Given** <ASSERTION> { <ASSERTION> }*
**Conclude** <ASSERTION>
**By** <JUSTIFICATION>

<ASSERTION> ::= *predicate from the domain*

<JUSTIFICATION> ::=
**Justification** *identifier*
**Comment** *explanatory text*
**Definition** <ASSERTION> { <WARRANT> <ASSERTION> }*

<WARRANT> := **By Knowledge Of** *identifier* |
**By Function** *identifier1* **Of Case** *identifier2* |
**By Justification** *identifier*

The last of these sections also addresses the notion of how the case retrieval algorithm can be adapted to provide automatic indexing of cases into the issue composition hierarchy as cases enter the conceptual memory.

## 6.2 Index Vocabulary

An index vocabulary consists of those terms from which the user may construct queries to the memory. In traditional key-word systems, this vocabulary is defined as the set of all significant words appearing in a stored document. These terms are then combined using Boolean and adjacency operators to identify syntactic patterns of words in documents. Thus, the nature of the index terms, and the queries formed from them, is based directly on how items in the data base are stored — as full-text documents in natural language.

In one sense, the approach to *conceptual* retrieval proposed herein applies a similar notion: the index vocabulary arises from case representation. Cases in memory are represented using the functional notation described in Chapter 5 (Table 17). This notation serves as the source of index terms by explicating particular conceptual roles for justification statements to fill — most notably, warrants and claims. In capturing these basic elements of justificatory reasoning, such a memory can provide responses to queries generated in the course of problem-solving activity, queries that deal with particular steps in the process of justification.

In this respect, however, the index vocabulary proposed here differs from the general-purpose lexicons such as KWIC. The problem-solving architecture in which the conceptual memory resides places particular functional requirements on the content of its performance (Figure 18). Queries are sent to the memory in the course of problem solving. These requests may be for specific cases, or they may be for any cases that can play a certain role in assembling a justification. In order to fill these requests, the memory may

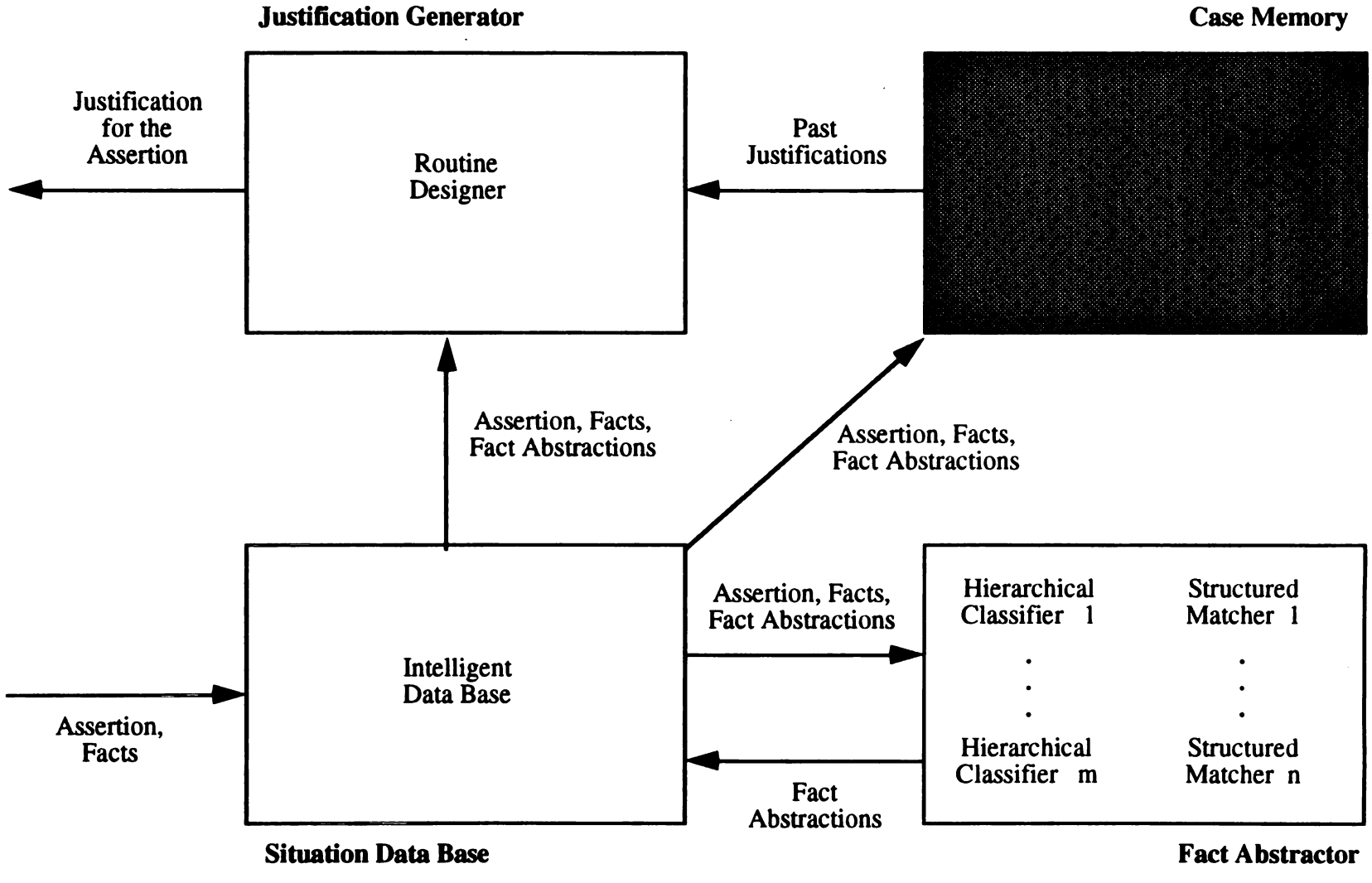


**Figure 18.** The Problem Solving Architecture for Legal Justification

refer to the facts of the current situation. This architecture, together with the functional representation for justifications, provides strong guidance in identifying an appropriate index vocabulary for the conceptual memory. The memory's task is focused, which focuses the types of queries to which it must respond.

This chapter describes the domain knowledge and problem-solving method that comprise the case memory black box in Figure 18. This description begins with a specification of the kinds of query that the memory can answer. Queries to the case memory can be of two types[1]:

- a request for a particular case and function, by citation, or
- a request for any cases that addresses a particular issue.

Together, these can be viewed as requests based on a specific sort of *abstract feature*: the ability to help justify a particular assertion. Each of these two types of query is characterized by a particular set of index terms that shapes the search space of cases. Furthermore, each type of index can be traced back to a particular element of the functional case representation. This synergy between representation and vocabulary is addressed further below.

### 6.2.1 Index Terms for Case Citation Queries

The simplest form of query requests a particular case. The case representation supports a direct form of case-to-case retrieval through the explicit use of case citations as warrants. In the course of problem solving, the justification generator may wish to obtain

---

1 Chapter 4 also mentions a third type of query to conceptual memory — a request for cases that share a set of surface features with the current fact situation. Such "surface features" consist of the facts and fact abstractions that characterize the situation. Research on retrieval of this type has been conducted in several disciplines (see Chapter 2). Meldman [1977], McCarty [1977], Hafner [1981, 1987], and Ashley [1990] have all offered potential approaches to this problem. Retrieval based on surface features is a general-purpose retrieval strategy, not tied in any direct way to a particular problem-solving task or method. Since this thesis is concerned with conceptual retrieval for a task-specific problem solver, retrieval based on surface features will be deferred until it can be discussed in the context of task-directed retrieval.

more detail about the justification for an assertion in an already-retrieved case. This need corresponds to the expansion of a **By Function ... of Case...** warrant. Queries of this type involve requesting the case cited in the warrant, and in particular the function responsible for the inference. The index vocabulary for such queries consists of the set of all case identifiers in the memory.[2]

### 6.2.2 Index Terms for Justification Queries

The second kind of query focuses on a specific assertion to be supported. In order to justify the assertion, the justification generator can request precedent cases that have justified the same assertion (or a similar one). This sort of retrieval comprises the main contribution of representing cases in the FR — the ability to retrieve cases based on the justifications they support. Assertions of this type appear as postconditions of functions, in the **By** slot of a case's **Function** frame. The index vocabulary for issue justification queries consists of assertions to be justified, in the form of domain predicates.

Additionally, the justification generator may desire precedents that justify the assertion from one or more specific facts as assumptions. In this situation, the generator may also include in its query the facts of the current case (or a subset of them). These facts may appear as preconditions of a function, in the **Given** slot of a case's **Function** frame, or as **Facts** in the **Context** frame. Like assertions to be justified, these facts are also predicates from the domain, either facts input to the system or fact abstractions derived in problem solving. Thus, the full index vocabulary for this class of queries consists of domain-predicate assertions.

---

2 Note that this is not the same as the full legal citation stored in the **Citation** slot of the **Legal Documentation** frame of the case, but rather is a system-dependent name assigned to the case for internal reference. In traditional KWIC systems, one can request individual cases by specifying the legal citation in a query. This is not because such citations are used as case identifiers but because the legal citation appears in the text of the stored document. Thus, KWIC systems maintain a consistency of index vocabulary by including in the free-text data base all information that a user is likely to request. However, care must be taken in specifying the query so as not to retrieve all cases in which the target case is merely mentioned, for instance, as a citation or sidebar.

### 6.2.3 Summary of Index Vocabulary

These query classes yield two distinct components of the index vocabulary: *case identifiers* and *assertions about a fact situation.* The case memory must be able to retrieve cases characterized by each of these classes of index terms. Furthermore, retrieval of cases based on assertions must handle two circumstances: assertions to be justified in the context of a given fact situation, and assertions that characterize the situation in which an assertion is to be justified.[3]

## 6.3 Index Organization

An index organization specifies the relationship among terms in an index vocabulary. Such an organization is closely tied to the development of efficient algorithms for searching the index space. This section describes index organizations of the vocabulary for case citation and justification queries. The research contribution made here is the introduction of the *issue composition hierarchy.* This hierarchy relates assertions based on their roles as preconditions for justifying other assertions in the domain, thus providing a mechanism for organizing indices for justification queries.

### 6.3.1 Index Organization for Case Citation Indices

The index vocabulary for case citation queries consists of case identifiers, specifically those that appear in the **By Function ... Of Case ...** slot of any justification warrant. These indices provide for a "direct look-up" of cases and functions. Cases can be organized in any suitable data base format (for example, a flat table or a B-tree), and the identifier provided in the query can be used as the key in a standard data base

---

3 One can also conceive of retrieval based on other attributes of the representation. For example, one may wish to retrieve all cases that cite a particular case. This type of query could be supported by the representation, through indexing of **By Function** warrants according the case cited. KWIC systems support such retrieval by searching for text references to the target case. This and other such queries are not considered further in this work.

search. In this situation, the case organization and the retrieval algorithm are trivially defined: given a case citation index, retrieve the case by look-up.[4]

Such a direct look-up is one of the central features of the functional representation for legal justifications: whenever a case is cited directly in support of an assertion, then that case may be relevant in justifying a similar assertion in a later situation. The FR for cases makes the retrieval of such a case a trivial matter of table access. Once the case has been retrieved, the target function can also be retrieved directly, using the function identifier appearing in the warrant. This kind of direct indexing of behaviors (here, justifications) according to the functions they denote was one of the central features of the original Functional Representation for devices.

### 6.3.2 Index Organization for Justification Indices

*Viewing a Body of Case Law as a Whole* The cases that make up a body of law — for example, the body of case law surrounding captive insurance arrangements — comprise a set of justifications. These cases classify fact patterns as positive or negative instances of important domain concepts, such as *insurance*. Each classification, an assertion regarding a particular fact situation, is supported by a justificatory line of reasoning. In justifying a classification, a case decision may refer to other issues in the domain as part of its justification. For example, the courts have found that the presence of *risk shifting* and *risk distribution* can be determinative of the presence of insurance. One can thus support a claim that insurance is present by showing that risk was both shifted and distributed. In

---

4 One can also conceive of a case citation query that requests a particular case and *all* cases cited therein. Such a query might be useful in a situation where the problem solver wishes to adapt all phases of the requested case's justifications to the situation at hand. The case memory would then have to recursively expand all case citation warrants in the requested case. More generally, though, the problem solver can request the specific case first and then request each individual expansion as desired. This approach leaves the specification of expansion knowledge to the external problem solver, making the case memory's interface with such problem solvers simpler and more general.

this sense, the issue of insurance has been "decomposed" into two subissues, risk shifting and risk distribution (Figure 19).

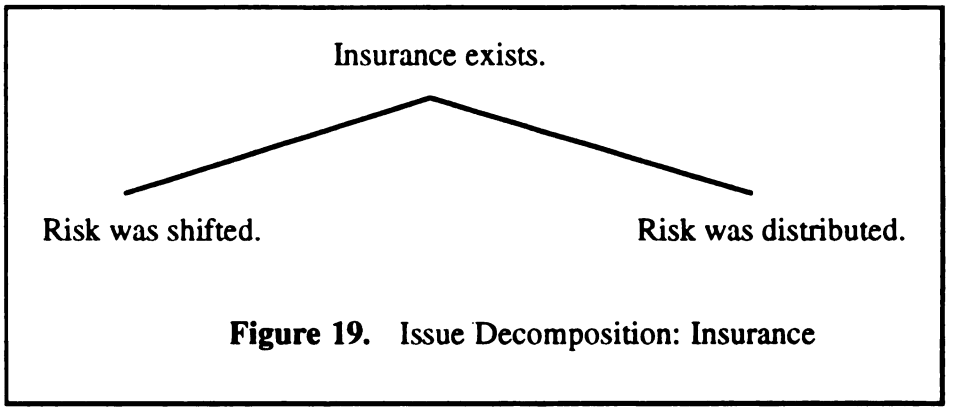


**Figure 19.** Issue Decomposition: Insurance

Cases are characterized by the issues they address. For each such issue, the case is assigned a function that denotes the line of reasoning that supports the case's conclusion regarding the issue. Thus, the issues of the domain can be characterized by the various case functions that address them. This connection between issues in the domain and functions in the case representation indicates an extra utility in considering issues as the defining features of the case law. Not only do issues capture the important classifications made in the domain, but they also provide direct links to the functions of individual cases.

The body of case law is characterized by the issues addressed in its cases. These issues are further related along the dimension of *composition*. If an assertion about $issue_2$ can be used as a precondition for justifying an assertion about $issue_1$, then a justification of the former assertion can be used as a component in justifying the latter assertion. In this case, $issue_1$ *is composed of* $issue_2$. The composition relation is not a necessary one; it denotes only that $issue_2$ *may* be used in support of $issue_1$. This way of relating assertions and the issues to which they refer facilitates an organization of the case law that directly supports justification of assertions, via reference to case functions.

*The Issue Composition Hierarchy* The index vocabulary of justification queries consists of domain assertions: the memory is given a query in the form of an assertion to be justified. These indices are related in a directed acyclic graph called an **issue composition hierarchy (ICH)**. This hierarchy consists of two elements:

- Nodes correspond to issues from the domain. In the domain of captive insurance, these might include insurance, risk shifting, and risk distribution.
- Edges reflect the composition relation. For nodes $N_1$ and $N_2$, there exists an edge $(N_1,N_2)$ if and only if justification of an assertion regarding $N_2$ can be used as a precondition for justifying an assertion regarding $N_1$.

The graph is directed and acyclic because legal issue composition is both antisymmetric and acyclic.[5] Such a hierarchy is rooted at the central concept of the domain. In the case of captive insurance arrangements, this is the concept of *insurance*. All other issues in the domain relate back to the concept of insurance via composition links. Figure 20 presents a partial issue composition hierarchy for the domain of captive insurance.

The ICH may not be a true hierarchy. For instance, in Figure 20, an assertion that the alleged insurer is a legitimate provider of insurance can be used as data in justifying that risk was both shifted and distributed.[6] Consequently, the node for *legitimate provider of insurance* is linked to two parent nodes, *risk sharing* and *risk distribution.* This feature of the ICH reflects the interrelated nature of legal concepts. The definition of the ICH's edges

---

5 That is, if classification as an instance of one issue is potentially a precondition for classification as an instance of another, the latter issue cannot also be a precondition of the former. This is true for a single edge (antisymmetric) or for multiple edges (acyclic).

6 See Sears, Roebuck, and Company versus Commissioner of Internal Revenue [1991].

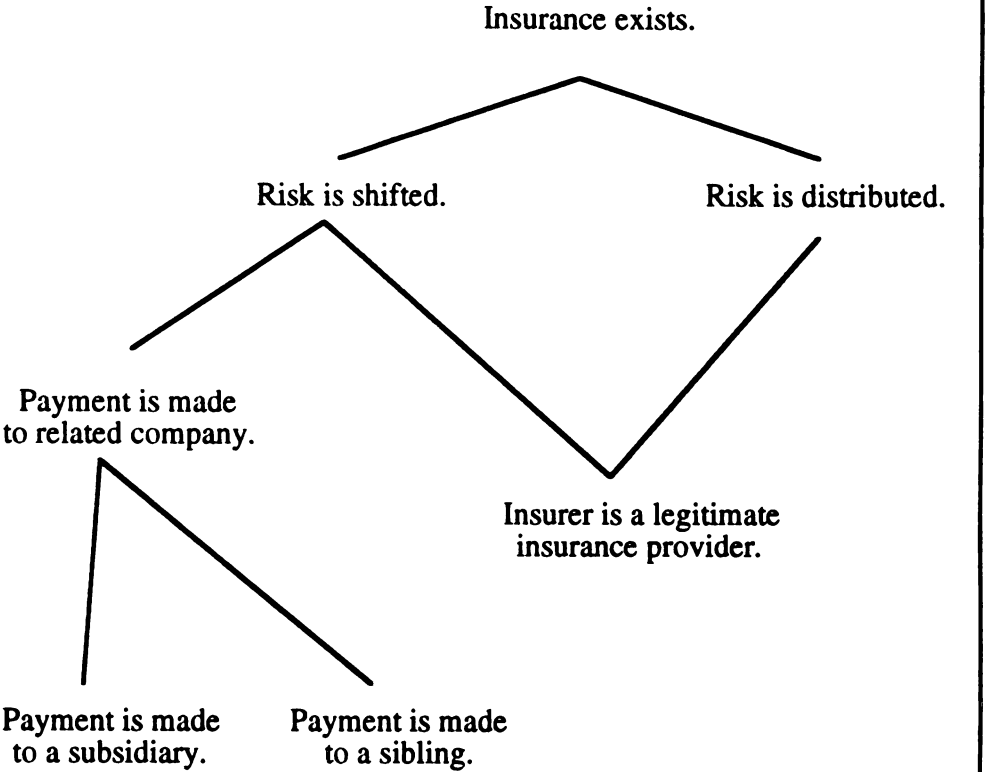


**Figure 20.** A Sample Issue Composition Hierarchy

also permits the notion of a negative correlation. A justification for the assertion that the payment is made to a related company will be useful for justifying the assertion that risk is *not* shifted. Conversely, a justification for the assertion that the payment is not made to a related company will be useful for justifying the assertion that risk is shifted. The critical relationship that defines the ICH is that assertions regarding the child issue may be useful in justifying assertions about the parent issue.

*The ICH as an Index into Case Memory* The connection between issues and case functions makes the ICH an effective and efficient index into case memory. Each issue

node contains a pointer to the cases that deal with the issue, in particular, to the case functions that justify the corresponding assertions. Thus, cases are organized, at the level of the body of case law, according to their defining features — the functions they play in justifying domain assertions. Just as individual cases are defined by their functions, the body of case law is organized according to these functions. In one sense, then, one can view the body of case law as a device, with its individual cases as component subdevices. The top-level device has no independent functions except those of its components. The issue composition hierarchy comprises a device-level index of these constituent functions (and the corresponding *justifications*).

However, this single stage of indexing will be insufficient for providing adequate retrieval of cases for justificatory reasoning. At most nodes in the hierarchy, there will be several cases that deal with the issue. When constructing a new justification, the problem solver will certainly desire a finer grain of distinction among the cases. In each case, the issue will have been decided positively or negatively.[7] Further, in each case, the decision favored either the related group of companies or the Internal Revenue Service. These features of the case play an important part in the problem solver determining whether or not the case will be useful for the purpose at hand. For example, a case in which the issue was decided negatively may be of little use if the problem solver needs to justify a positive classification.

At this point in the indexing process, the use of case features to guide retrieval also becomes critical. Retrieval of cases based solely on their feature similarity provides little direct support for the task of justificatory reasoning. However, *in the context of* justification roles, case features themselves can become useful in choosing the most relevant items from a set of cases judged to be useful for the task at hand. If two cases can provide assistance in justifying the claim that risk shifting has occurred, but the facts of one

---

7 That is, classification of the case's fact situation was either justified or unjustified.

of the cases more closely matches the current situation's facts than the other, then the more similar case will likely be more persuasive in a justification. Thus the salience of a case's features is determined only in the context of a particular task and its goals.

For these reasons, the conceptual memory requires a second stage of indexing. At each node, there exists a fact similarity matcher that maps the facts of the situation against those of the cases indexed by the node. This matcher segregates cases first according to issue resolution and case outcome and then according to facts and critical fact abstractions. The fact similarity matcher at each node provides a function similar to that of the compound pointer introduced by Hafner [1987].[8] This compound pointer consisted of four separate pointers, based on the cross product of two dimensions, issue resolution and case outcome. Figure 21 depicts such a pointer, for the issue of risk sharing. In effect, the structured matcher proposed here incorporates these two dimensions but also takes into account important fact patterns that characterize the cases. This second stage permits the case memory to provide even more focused retrieval, within the context of justifying a particular assertion.

Thus, the two stages of index organization provided by the ICH are:

- At the level of the body of case law, case functions are segregated by the issues they address. A case can be retrieved at any node corresponding to an issue with which the case deals. Nodes are linked by edges that reflect the composition relation among issues.
- At the level of the issue node, a fact similarity matcher segregates cases according (1) to issue and case outcome and (2) characteristic fact patterns. Each case indexed by the node will appear in exactly one of the equivalence classes defined by the similarity matcher.

---

8 See Chapter 2 for a discussion of Hafner and her model of legal information retrieval.

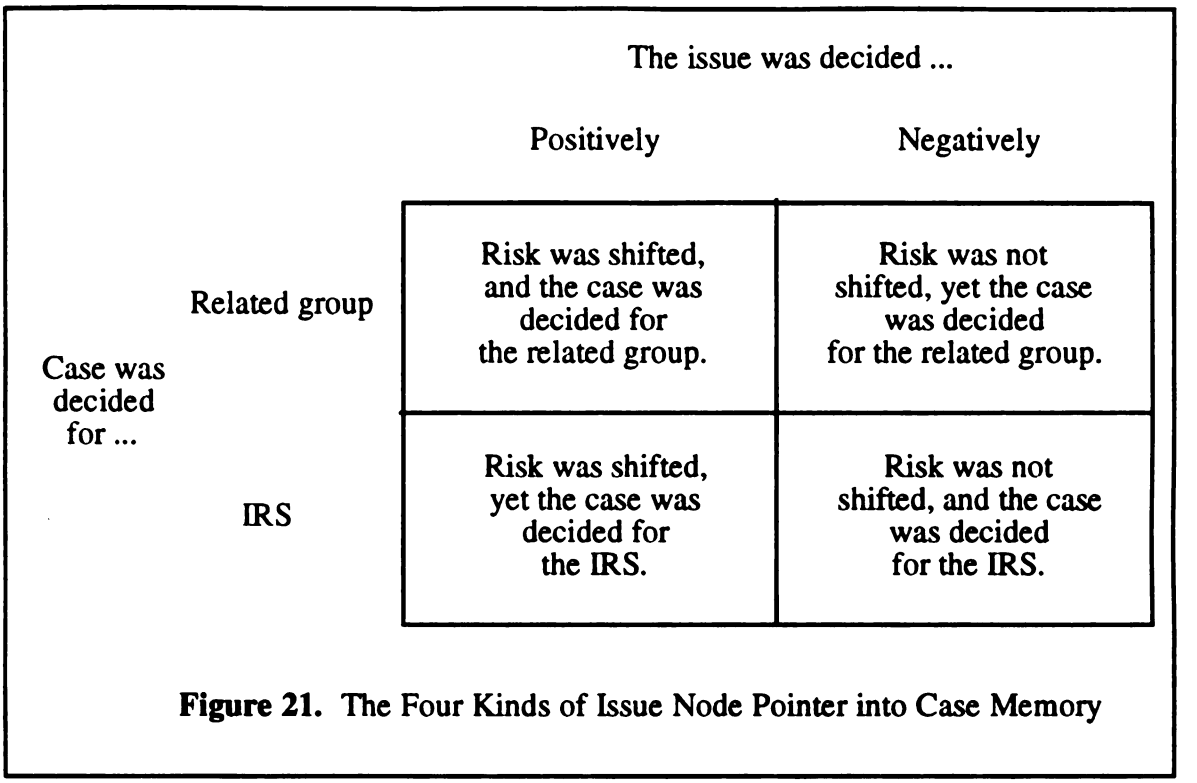


**Figure 21.** The Four Kinds of Issue Node Pointer into Case Memory

## 6.4 The Case Retrieval Algorithm

Case memory will be searched using one of two algorithms, depending on the nature of the query. Case citation queries provide case identifiers as indices into memory, and (as introduced above) a standard table look-up or B-tree traversal suffices as the retrieval algorithm. Justification queries provide assertions as indices into the memory, and thus they require search of the issue composition hierarchy. This algorithm is outlined in Table 18.

This "match and decompose" algorithm is a variation of the *establish-and-refine* method of hierarchical classification [Bylander and Mittal 1986; Sticklen, Chandrasekaran, and Josephson 1987]. At each node in the ICH, the algorithm attempts to match the assertion against the node's issue. Three possible values can be returned by the match process: exact match, subissue match, and no match. Any exact match indicates that the

**Table 18.** A Case Retrieval Algorithm for Searching the ICH

GIVEN: An assertion to justify
An issue node (initially, the root of the ICH)

1. Match the assertion against the issue at the current node.
2. If there is an exact match: Determine which of the cases at the current node is most similar to the situation at hand. Return the set of cases so indicated. Halt.
3. If there is no match: Return in failure.
{Otherwise, the match signifies a potential subissue match.}
4. For each subnode of the current node, call the case retrieval algorithm with the subnode.
5. If there are no subnodes, or if all subnodes return in failure: Return in failure.

assertion deals directly with issue at hand. A subissue match indicates that the assertion deals with a subissue of the current issue. Finally, no match indicates that the assertion has no semantic connection at all to the current issue and thus has no connection to any of its subissues.

Any time an assertion exactly matches an issue node, the set of cases pointed by the node are directly relevant to justifying the assertion. In this situation, the algorithm invokes the structured matcher that selects among the node's cases and returns the resulting case set. Any time there is no match at all, the subgraph of the ICH rooted at the current node can be ignored; none of the issues in that subgraph are related to the assertion. In the situation of a "partial" subissue match, the node has recognized that the assertion is not directly related to the issue at the current node but may be related to one of its subnodes. So the current issue is decomposed into its subissues, and the "match and decompose" algorithm is called recursively for each subissue node. This recursive decomposition continues until one of two conditions is met: If an exact match is found, the relevant cases pointed to by the matching node are returned as the result, and processing halts. If the

whole ICH has been explored and no exact matches have been found, then the algorithm returns no cases and halts.

The idea behind this algorithm is a simple one: the issue composition hierarchy reflects relationships among issues in the domain. By traversing the ICH in a match-and-decompose fashion, only those parts of the hierarchy that are potentially relevant to the target issue are explored. As soon as a node recognizes that the assertion is not relevant to its issue, exploration of that portion of the hierarchy can be terminated. Eventually, the assertion is matched against the issue with which it deals, and the cases pointed to by that issue's node can be sent to the problem solver.

Only two situations exist in which an assertion will find no match in the hierarchy. First, the assertion may be a new one in the domain, never having been addressed in a prior case. This situation, which calls for an extension of the hierarchy, is considered in the following section. Second, the knowledge necessary for recognizing an exact match is missing in the appropriate node. This situation is primarily a problem of language. If the input to the case memory were an assertion in natural language, it is quite plausible that a query having an exact match in the ICH could find no match, since the assertion could be phrased in a novel way. However, in the context of the problem-solving architecture presented in Figure 18, queries to the case memory consists of formalized domain predicates handled by the justification generator and situation data base. This limited representation enables a more complete enumeration of the (kinds of) assertions that are relevant to any given issue.

### 6.4.1 Match Knowledge in the ICH

The case retrieval algorithm does not specify the nature of the match knowledge possessed by the nodes in the hierarchy for determining the issue's relevance. In hierarchical classification, establishment knowledge typically consists of compiled fact

patterns that characterize the classification category. A structured matcher compares situation data with these fact patterns, returning a confidence value for the hypothesis that the situation is an instance of the category. A similar strategy suffices for the match of an assertion to an issue node in the ICH. Indeed, the ICH node matching problem will often be simpler than the hierarchical classification scenario. Match knowledge must identify an assertion as an exact or partial match. Since an assertion consists of a formal domain predicate, the match process can directly compare the predicate for an exact match. A subissue match will be indicated whenever the assertion matches an assertion known to relate to a particular subissue, or when the assertion is a specific instance of a more general predicate related to a subissue. This latter form of partial match can often be accomplished by sending a request to the situation data base regarding the predicate in question.

### 6.4.2 Use of the Retrieval Algorithm for Automatic Indexing

In order for the issue composition hierarchy to grow as new cases are decided, some mechanism must exist for indexing cases as they are added to the conceptual memory. The case retrieval algorithm offers a potential mechanism for *automatic* indexing of new cases. Table 19 offers an adaptation of the retrieval algorithm aimed at this goal. Given a case represented in the FR, the indexing algorithm adds the case to the case citation index (Step 2) and then attempts to add each function of the case to the issue composition hierarchy (Step 3). To achieve the last step, the match-and-decompose technique of the case retrieval algorithm is called in an attempt to find the proper node for indexing each function of the case.

However, this search may fail, for one of two reasons: the proper node does not have adequate match knowledge to recognize that the assertion justified by the function should be indexed there, or there is no matching node. In the former situation, some outside agent — perhaps a human user — must instruct the memory to index the case at the

**Table 19.** A Case Indexing Algorithm

GIVEN: A case represented in the FR

1. Add the case to memory.
2. Add the case identifier, and a pointer to the case, to the case citation index.
3. For each function of the case:
   a. Apply the match-and-decompose algorithm to find an exact match in the issue composition hierarchy.
   b. If an exact match is found, then add the case and function to the set of cases indicated by the matching node's fact similarity matcher.
   c. If no exact match is found, then ask the user whether (1) the function should be stored at an existing node or (2) thefunction represents a new issue.
      i. If the function should be stored at an existing node, store the case and function as in Step 3b.
      ii. If the function represents a new issue, create a new node as directed by the user, and store the case and function as in Step 3b.

appropriate node. (The agent should then, if possible, also augment that node's match knowledge so that the node will recognize the assertion on later attempts.) The latter situation signifies that the ICH is incomplete, that a new issue decomposition should be added to the hierarchy. At the node or nodes which were unable to decompose themselves further, the algorithm can add an appropriate subissue node and index the new case at that point.[9]

9 Another possibility exists: no exact or subissue match succeeds at any node. In this scenario, the algorithm requires outside assistance in indexing the new case. Furthermore, the outside agent should examine this case, the ICH, and the match knowledge in the hierarchy to determine the ultimate source of the algorithm's failure.

## 6.5 Conclusion

The functional representation for legal analysis makes possible a memory indexing strategy that reflects the utility of cases in justificatory reasoning. This chapter presents a model of conceptual retrieval, based on the FR for cases, for use in a problem solving architecture for this task. This model — which specifies how cases are requested, how they are indexed and organized, and how they are retrieved — employs the advantages inherent in a functional representation for indexing knowledge about abstract devices. In essence, the model consists of two interrelated indices:

- the *case citation index*, for when the problem solver knows of a particular case and function that can help to justify a particular assertion, and
- the *issue composition hierarchy*, for when the problem solver needs cases that can help to justify an assertion but does not already have a direct reference to a particular case.

This two-part model relies directly on features of the functional representation of cases for its indexing and organization scheme. The use of case citations as warrants for justifications and the indexing of justifications according to the role they play in the case are fundamental attributes of this FR, and they capture basic elements of legal reasoning. In this way, the representation of cases is closely tied to the organization of cases in memory, and both follow from domain-specific and task-specific understanding of how justifications are used.

# CHAPTER 7

# CRISTA: A COMPUTER PROGRAM FOR CONCEPTUAL RETRIEVAL

## 7.1 Introduction

The previous three chapters outline the elements of a legal analysis problem solver. Chapter 4 presents a problem solving architecture (PSA) for justificatory legal analysis in terms of Generic Task problem solvers. In Chapter 5, a functional representation for legal cases is developed. Finally, Chapter 6 proposes a model of conceptual memory that is based on the functional representation of cases. This model is designed to serve as the case memory component in the problem solving architecture of Chapter 4. These chapters provide an abstract description of a case-based justificatory reasoner, with examples taken from the tax law.

This chapter describes CRISTA, a computer program that implements a portion of the problem solving architecture for the domain of captive insurance taxation. In particular, CRISTA embodies the model of conceptual memory described in Chapter 6. Cases in CRISTA's data base are represented using the FR for cases and are retrieved via an issue composition hierarchy for the domain. The chapter consists of two main sections:

- discussion of the CRISTA program and its implementation, and
- presentation of two samples problems solved by CRISTA, focusing on its use of case memory.

## 7.2 The Implementation of CRISTA

The computer program CRISTA implements a portion of the problem solving architecture outlined in Chapter 4. In order to describe this implementation in more detail, three issues must be addressed:

- the software environment in which CRISTA was constructed,
- the top-level organization and algorithm of CRISTA, and
- the sub-agents of CRISTA and their implementation.

### 7.2.1 The Software Environment

CRISTA is specified as an architecture of Generic Task problem solvers. In addition to providing an analytic theory of problem-solving types, the Generic Task approach [Chandrasekaran 1983, 1987] also offers a basis for constructing programming languages that embody these types. Each generic task specifies a particular problem-solving method. Such a method delineates the types of domain knowledge it requires and the control strategy for applying this knowledge. As a result, one can construct for each generic task a programming language that incorporates the abstract method and its control strategy. This kind of language can be thought of as a programming "shell," in which the user enumerates domain knowledge of a particular type, for a particular task.

In the AI/KBS Laboratory at Michigan State University, a software environment consisting of the Generic Task languages has been developed. The foundation for this environment is ParcPlace's Objectworks™, a full-featured implementation of the object-oriented language Smalltalk. Smalltalk was selected as the infrastructure for the AI/KBS environment in large part because of its pure message-passing model of object interaction. Use of such a model follows naturally from the Generic Task notion of an intelligent agent as a community of cooperating knowledge specialists. Furthermore, Objectworks™ offers a number of other features important to a research software environment, including

complete compatibility of source code and compiled images across hardware platforms as well as compatible graphics support.

The basis of the AI/KBS Lab's Generic Task environment is **SNO**[1], a set of Smalltalk classes that provides basic support for named objects and object relations. The SNO language was implemented largely by Dr. Jon Sticklen, with assistance from the author and other graduate students in the Lab. SNO furnishes primitives for the assembly of individual Generic Task languages and for the combination of multiple agents (built from these languages) into a problem-solving architecture of the type described in Chapter 4. In essence, SNO extends Objectworks™ by adding primitives that support the Generic Task languages.

The Generic Task software environment consists, at this time, of four generic task languages[2]:

- **HC**, for the GT of hierarchical classification [Bylander and Mittal 1986; Sticklen, Chandrasekaran, and Josephson 1987],
- **SM**, for the GT of structured matching [Bylander, Johnson, and Goel 1991; Sticklen, Chandrasekaran, and Josephson 1987],
- **DSPL**, for the GT of routine design [Brown 1987, Brown and Chandrasekaran 1986], and
- **FM**, for the functional representation of devices [Sticklen 1987, Sembugamoorthy and Chandrasekaran 1986].

The author of this thesis wrote HC and SM. Other graduate students in the Lab, Ahmed Kamel and Kurt Patzer, were the principal builders of DSPL and FM, respectively. These languages now comprise a "tool bench" of GT shells from which one can construct Generic

---

1 For "Spartan Named Object."

2 Each of these languages was originally implemented in Lisp environments at the Laboratory for AI Research at the Ohio State University. The corresponding languages — CSRL, HYPER, DSPL, and FR — are described in the cited works.

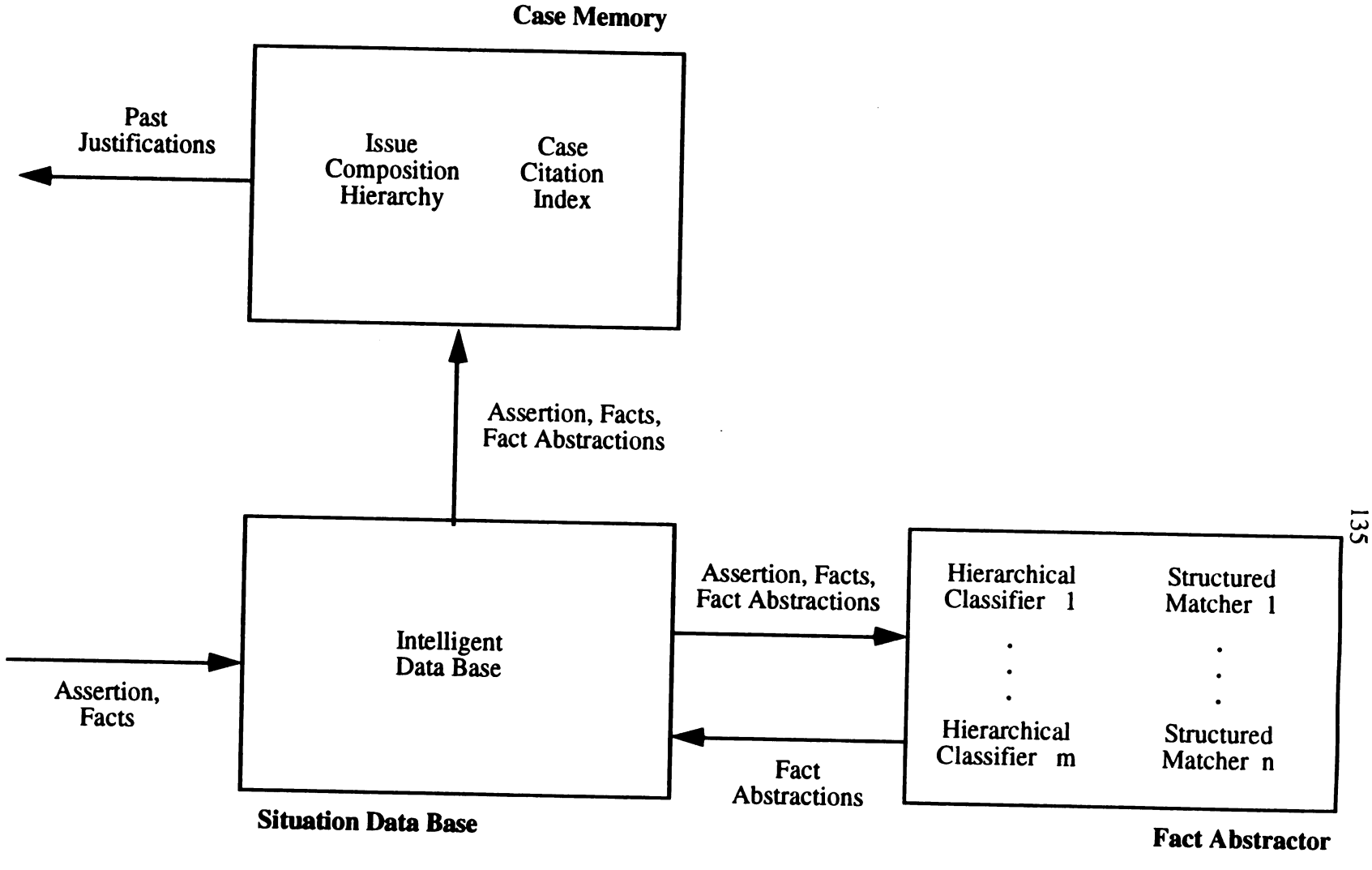


**Figure 22.** CRISTA's Problem Solving Architecture

Task problem solvers in an integrated environment. It is from this tool bench that CRISTA was built.[3] Any extensions to the basic languages are described when appropriate below.

### 7.2.2 CRISTA: The Top Level

CRISTA was designed and implemented to test the idea of functional case representation and the model of conceptual retrieval based on this representation. As such, it embodies a large portion of the problem-solving architecture for justificatory analysis outlined (Figure 22). CRISTA's architecture includes three of the agents from the legal analysis architecture: the Fact Abstractor, the Situation Data Base, and the Case Memory. These agents constitute the basic requirements for conceptual retrieval in the model detailed in Chapter 6. The Data Base manages all data and data acquisition for the Case Memory. The Fact Abstractor generates fact abstractions for use by the Case Memory, on request from the Data Base. Finally, the Case Memory retrieves cases based on their ability to assist in justifying assertions provided by the user.

**Table 20.** CRISTA's Control Strategy — Abstract Level

1. Request an assertion to be justified and the initial data that describe the legal situation.
2. Identify relevant legal abstractions of the data.
3. Identify relevant past justifications (cases).
4. Go to Step 1.

[3] SNO also provides a primitive form of the inferencing data base described by Mittal, Chandrasekaran, and Sticklen [1984]. This incomplete version of the intelligent data base generic task is sufficient to implement the Situation Data Base of the architecture discussed below.

The omission of the Justification Generator agent from the full justification architecture means that CRISTA must have a top-level control strategy for directing its components in the task of case retrieval. Indeed, CRISTA's control strategy (Table 20) follows that of the proposed Justification Generator with respect to the retrieving of past cases. This control strategy is implemented as a sequence of message exchanges among the agents and system user. Table 21 expands the control strategy to demonstrate the messages that are exchanged. The reader will note that, while the architecture itself is fixed[4], the number and sequence of requests made in retrieving cases is dynamic. Requests are made only when an agent requires particular facts or fact abstractions in the course of addressing a given legal situation. Thus, the control strategy for CRISTA as a whole is driven solely by interactions among its individual agents.

### 7.2.3 CRISTA: The Subagents

CRISTA consists of three subagents. Each of these subagents embodies a specific problem-solving method and possesses specific forms of domain knowledge. This section describes the domain knowledge and particular implementation details that characterize each of these subagents: the Situation Data Base, the Fact Abstractor, and the Case Memory.

**Situation Data Base** The Situation Data Base provides a subset of the capabilities found in an instance of the intelligent database generic task (Footnote 2). Its two primary responsibilities are to manage the situational data that describe a case and to manage all interaction with the system user. In CRISTA, the Data Base is an instance of the class AbstractionDataBase, which is a subclass of the standard SNO data base. This subclass extends the SNO data base to allow direction of data variables to fact abstraction agents.

---

4 That is, the paths of communication among the agents are fixed at program design time. See Table 3.

**Table 21.** CRISTA's Control Strategy — Message-Passing Level

0. System user initiates a session with CRISTA by requesting a legal justification from the Case Memory.

1. Case Memory requests an assertion from the Situation Data Base, which forwards the request to the user.

   On the first such request, Data Base also asks the user for the initial data that describe the legal situation.

   The assertion is returned to the Case Memory.

2. In the course of retrieving relevant justifications, the Case Memory may request relevant legal abstractions of the case data from the Situation Data Base.

   In the course of filling this request, the Situation Data Base may request (a) additional case data from the user and (b) new fact abstractions from the Fact Abstractor.

   In the course of identifying relevant legal abstractions, the Fact Abstractor may request additional case data from the Situation Data Base.

3. The Case Memory answers the initial request (Step 1) by sending relevant past justifications to the system user.

4. Go to Step 1.

**Table 22.** Channels of Communication in CRISTA

| | |
|---|---|
| **Fact Abstractor** | |
| **Input:** | Assertion, Facts, Fact Abstractions |
| **Requested From:** | Situation Data Base |
| **Output:** | Fact Abstractions |
| **Requested By:** | Situation Data Base |
| **Case Memory** | |
| **Input:** | Assertion, Facts, Fact Abstractions |
| **Requested From:** | Situation Data Base |
| **Output:** | Past Justifications |
| **Requested By:** | < System User > |
| **Situation Data Base** | |
| **Input:** | Assertion, Facts |
| **Requested From:** | < System User > |
| **Input:** | Fact Abstractions |
| **Requested From:** | Fact Abstractor |
| **Output:** | Assertion, Facts, Fact Abstractions |
| **Requested By:** | Fact Abstractor, Case Memory |

The Data Base maintains two types of data variables, facts and fact abstractions. For each fact, it has a slot for the fact's value and a method for obtaining a value from the user. For each fact abstraction, it has a slot for the abstraction's value and a pointer to the Fact Abstractor agent capable of determining its value. When the Data Base receives a request for a fact or abstraction, it first checks to see if the requested item has a value in the current case. If so, the Data Base returns that value to the requestor. If not, it sends a request to the appropriate source agent, either a specific fact abstractor or the system user, to determine a value. It then stores the value and sends it to the requestor.

The Situation Data Base has one variable for each fact and one variable for each fact abstraction. The fact abstraction agents are described below. For the domain of captive insurance taxation, CRISTA maintains approximately thirty fact variables. These are variables are of several types:

- numeric variables, such as *PercentOfRevenuesFromOutsideRisks* and *NumberOfIndependentBuyers*,
- binary variables, such as *IsInsurerFullyLicensed*, and
- qualitative variables, such as *NatureOfPremiumTerms*.

Several of these fact variables are listed in Table 23. The number of qualitative variables is smaller than that for the other two types. Typically, qualitative variables require some domain knowledge in order to abstract a qualitative value from a non-qualitative value. CRISTA's few qualitative variables require little judgment (leaving abstraction for the Fact Abstraction agents), instead calling for "multiple choice" answers from observables.

**Fact Abstractor** A set of structured matchers and hierarchical classifiers constitutes the fact abstraction agent of CRISTA. Each structured matcher generates an individual abstraction, and each hierarchical classifier generates an abstraction within a generalization hierarchy. In order to generate an abstraction, each fact abstractor may request the values

**Table 23.** A Sample of CRISTA's Fact Variables

**Numeric Variables**

PercentOfRevenuesFromOutsideRisks
NumberOfIndependentBuyers
PercentOfOwnershipInterest
PercentParentRisks
PercentSiblingRisks
PercentOfIndependentBuyers
PercentOfRiskReinsuredThroughThirdParty
NumberOfJurisdictionsLicensed
AmountOfInsurersCapital
AmountOfInsurersExposure

**Binary Variables**

IsInsurerFullyLicensed
IsInsurerRegulated
IsThereACapitalizationAgreement
AreThereOtherRelatedContracts
IsPaymentAFixedExpense
DoesParentDirectlyOwnSubsidiary
DoesSubsidiaryInsureSiblings
IsRiskReinsuredThroughThirdParty

**Qualitative Variables**

NatureOfPremiumTerms
JurisdictionOfInsurer

for data variables (either for facts or for other abstractions) from the Situation Data Base. The structured matchers are written in the language SM, and the hierarchical classifiers are written in the language HC. These languages directly implement the generic tasks of structured matching and hierarchical classification as described in Chapter 4.

For the domain of captive insurance taxation, CRISTA's fact abstractors supply two kinds of knowledge. First, matchers demonstrate hierarchical relationships among fact patterns for evaluating the value of a specific abstraction. Second, classifiers exhibit hierarchical relationships among *abstractions* in the domain. Each classification specialist possesses a structured matcher whose task is to determine the value of that specialist's abstraction. If this abstraction is present in the case data, then the specialist's children in the hierarchy — which represent more specific types of the abstraction — are explored.

CRISTA's fact abstractor consists of approximately five classifiers and twelve matchers. Together, these agents represent approximately fifteen classifier abstractions and thirty-five matcher abstractions. In total, these agents provide all the necessary fact abstractions for use in selecting relevant cases from the Case Memory. Figures 23 through 25 depict three of CRISTA's classifiers, and Figures 26 through 31 present six of CRISTA's matchers. These problem solvers are typical of the fact abstraction agents in CRISTA.

One may note that each of these agents is relatively small in comparison to classifiers and matchers in other domains (for example, in diagnostic medicine or biological taxonomy). In such domains, a single classification hierarchy might have dozens of classification specialists, with each specialist holding a structured matcher consisting of several simple matchers. The modest size of CRISTA's agents can be traced to two causes. Some agents are small because not all of the available domain knowledge has been

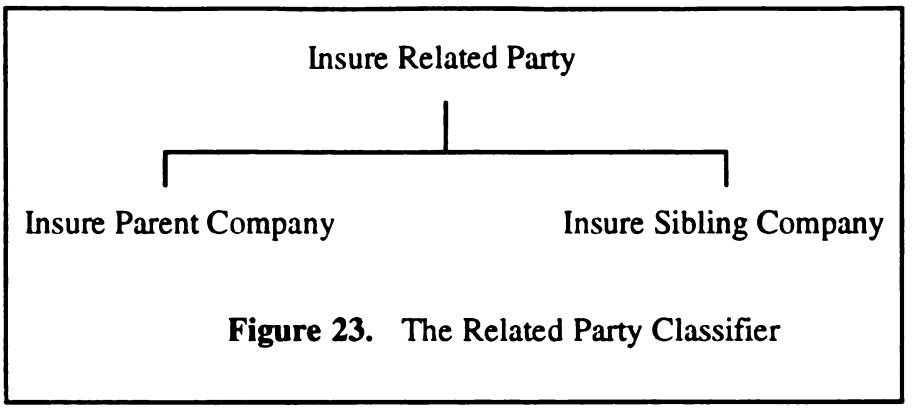


**Figure 23.** The Related Party Classifier

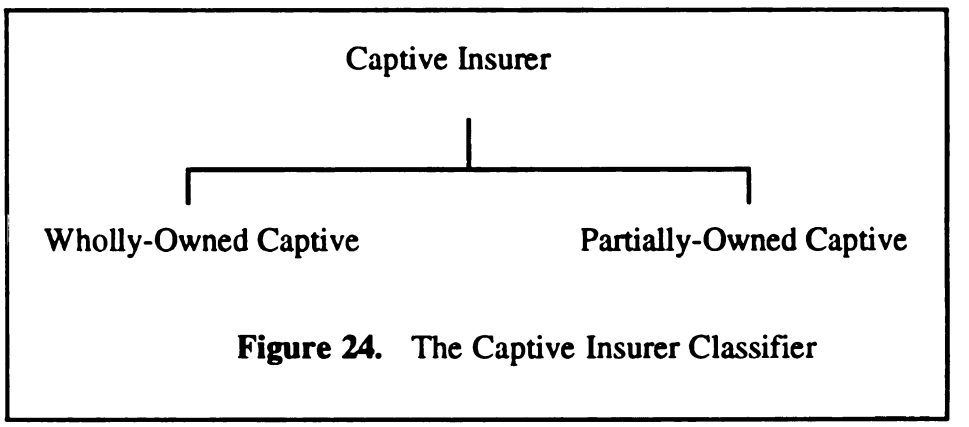


**Figure 24.** The Captive Insurer Classifier

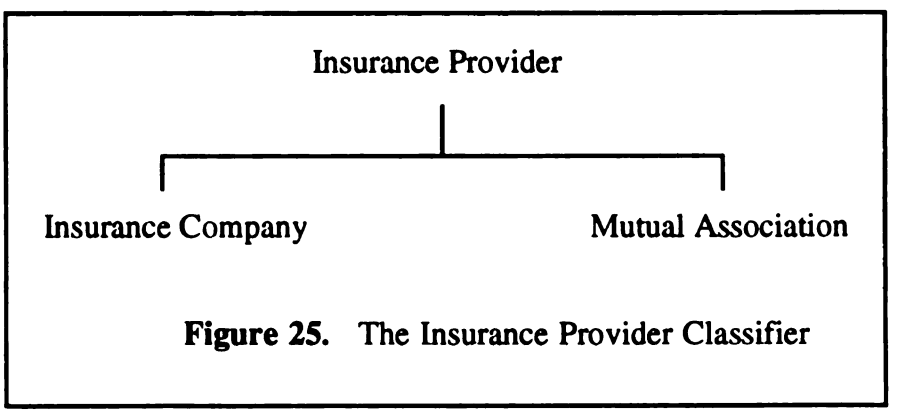


**Figure 25.** The Insurance Provider Classifier

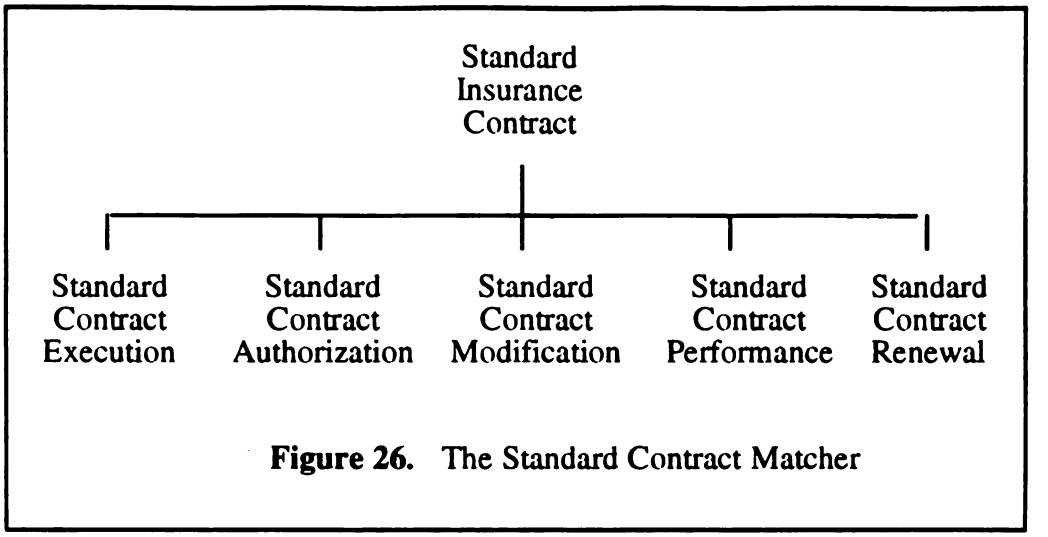


**Figure 26.** The Standard Contract Matcher

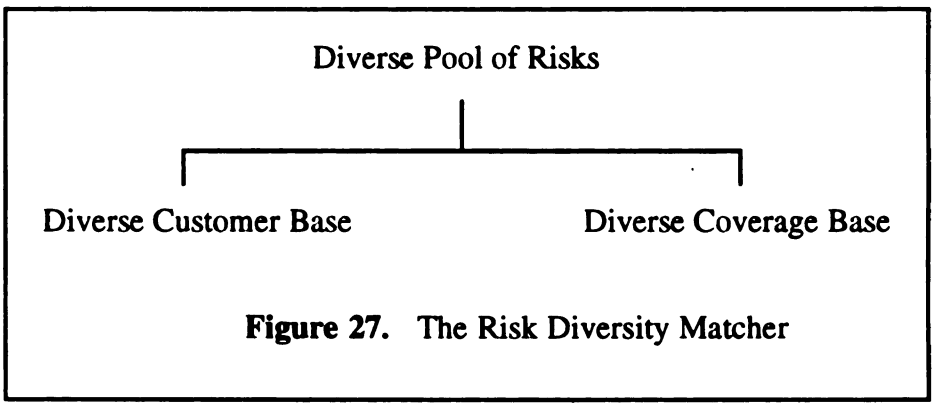


**Figure 27.** The Risk Diversity Matcher

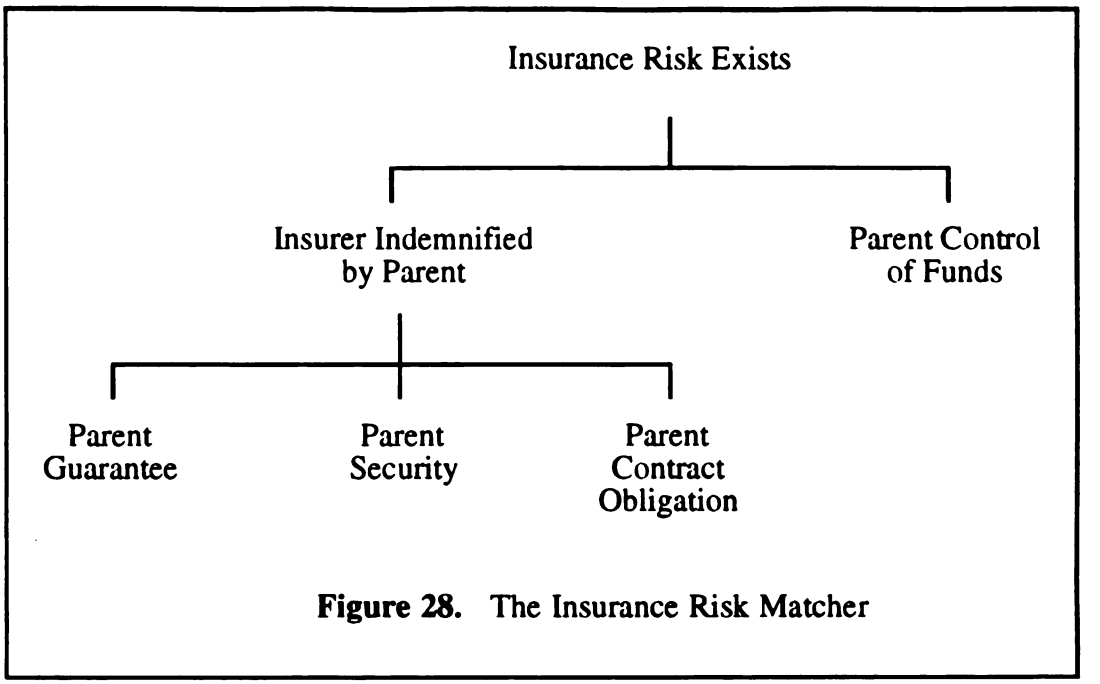


**Figure 28.** The Insurance Risk Matcher

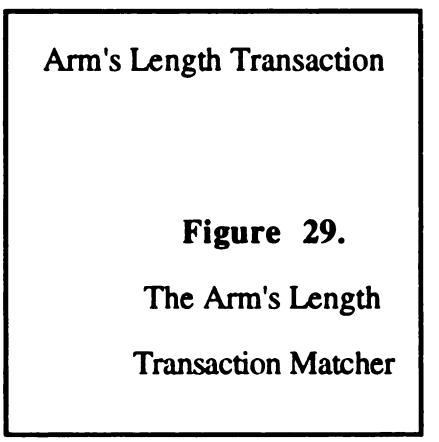


**Figure 29.**

The Arm's Length Transaction Matcher

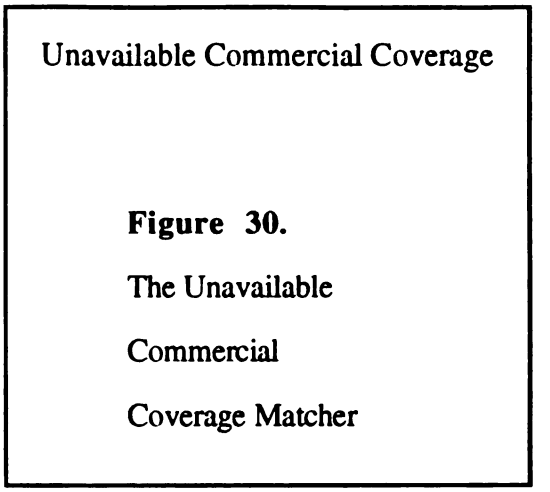


**Figure 30.**

The Unavailable

Commercial

Coverage Matcher

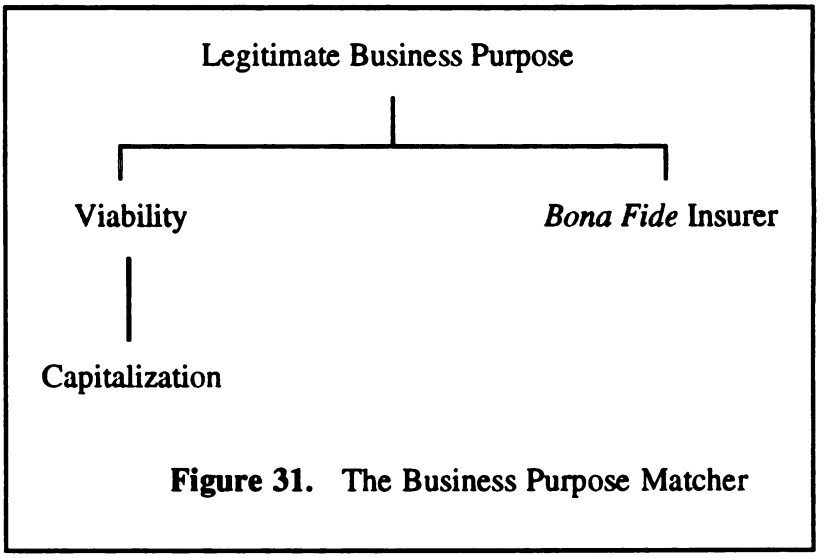


**Figure 31.** The Business Purpose Matcher

supplied to them. The Business Purpose matcher (Figure 31) could be decomposed to include other constituent abstractions from the domain, such as the role of current financial position in determining the insurer's viability. This kind of knowledge has been left out of CRISTA because it was not necessary to demonstrate the case memory facility of the program. More importantly, though, abstractions in this domain do not seem to consist of several levels of subdecisions. Adding more abstraction knowledge to CRISTA would more likely involve adding new, relatively independent matching and classification agents.

**Case Memory** CRISTA's Case Memory embodies the model of conceptual retrieval described in Chapter 6. The Memory accepts requests of two types. One can request the case indicated by a particular case identifier, or one can request cases that may help in justifying a particular assertion. The implemented Case Memory consists of three basic elements: the body of cases, the case citation index, and the issue composition hierarchy for the domain of captive insurance.

The cases themselves are implemented as devices in the language FM. This language was extended to allow two features of the FR for cases. First, the case representation supports legal documentation at the level of the case (device), issue (function), justification (behavior), and warrant (behavior links). Consequently, a subclass of each of these FM objects was created and augmented with slots to hold the appropriate documentation notes. Second, any precondition to a justification can be tagged as a "hypothetical" fact, which indicates that the fact need not be present in order for the justification to be applicable to a fact situation. An example of a case stored in CRISTA's memory appears in Table 24. In total, CRISTA contains 25 cases from the domain of captive insurance taxation.

The case citation index provides a direct mapping from case identifiers to cases. This index is implemented as a *dictionary* in Smalltalk, a data structure that consisting of

**Table 24.** A Sample Case from CRISTA — Humana [1989]

Case Humana-89

**Context**

**Legal Documentation**

| | |
|---|---|
| **Plaintiff** | Humana, Inc. |
| **Defendant** | Commissioner of Internal Revenue |
| **Citation** | 89-2 USTC ¶9453 |
| **Date** | July 7, 1989 |

**Procedural Context**

| | |
|---|---|
| **Setting** | U.S. Court of Appeals.<br>Plaintiff appeals ruling of the Tax Court that none of its intra-family insurance premiums are tax-deductible as business expenses. |
| **Outcome** | Affirms, reverses, and remands<br>88 TC 197, in parts. |

**Facts**

Plaintiff unable to obtain liability insurance at fair price.
Plaintiff incorporated wholly-owned insurance subsidiary.
Subsidiary meets all statutory and regulatory requirements.
Subsidiary provides insurance to plaintiff and its affiliates.
Subsidiary was fully capitalized at creation.
Plaintiff deducted all intra-family premiums as ordinary and necessary business expense.

**Issues**

| | |
|---|---|
| **Question** | Are payments made by an entity to its wholly-owned subsidiary deductible as insurance premiums? |
| **Holding** | No. |
| **Function** | Classify Parent Payments |
| **Given** | Parent pays premium to a subsidiary. |
| **Conclude** | Payment is not deductible. |
| **By** | Conclude No Parent Insurance |

| | |
|---|---|
| **Question** | Are payments made by an entity to a sibling deductible as insurance premiums? |
| **Holding** | Yes. |
| **Function** | Classify Subsidiary Payments |
| **Given** | Sibling pays premium to a sibling. |
| **Conclude** | Payment is deductible. |
| **By** | Conclude Sibling Insurance |

ordered key-value pairs. The case identifier serves as a key into the dictionary, which retrieves the associated value, which in CRISTA is the case itself. All the cases that appear in CRISTA's library are listed in Appendix A, along with the case identifiers that act as the keys in the case citation index.

The issue composition hierarchy organizes cases according to the roles they play in justifying assertions. In CRISTA, this hierarchy is implemented as a specialization of the hierarchical classifier provided by the language HC. This specialization embodies two special features of the ICH. First, HC is specialized to incorporate the match-and-decompose case retrieval method (in lieu of the standard establish-and-refine classification method) and the adapted case insertion method. Second, each node in the hierarchy requires two types of match knowledge. The issue specialist must have knowledge for determining whether an input assertion pertains to the issue. This knowledge corresponds to the "establish" knowledge of the standard classification specialist. Once the issue specialist has matched the assertion, it then needs additional knowledge for selecting relevant cases from the cases to which it points. This knowledge corresponds to the fact similarity matcher described in Chapter 6 — it selects cases based on the resolution of the issue and outcome of the case. Both of these forms of match knowledge are implemented in CRISTA as structured matchers, in the language SM.

Since CRISTA operates in the domain of captive insurance, its issue composition hierarchy captures relationships among assertions in the domain of insurance taxation. In Figures 32 and 33, the ICH that indexes CRISTA's Case Memory is given.[5] This index consists of eighteen nodes, which collectively point to the thirty-five functions that comprise the cases in CRISTA's memory. The nodes at the top of the hierarchy denote more general assertions about the domain, and so they point to a larger number of cases

---

5 The issue of risk shifting has been investigated in much greater detail by the courts, and as a result the portion of the ICH rooted at the node, "Risk is shifted," is considerably more developed than other parts of the hierarchy.

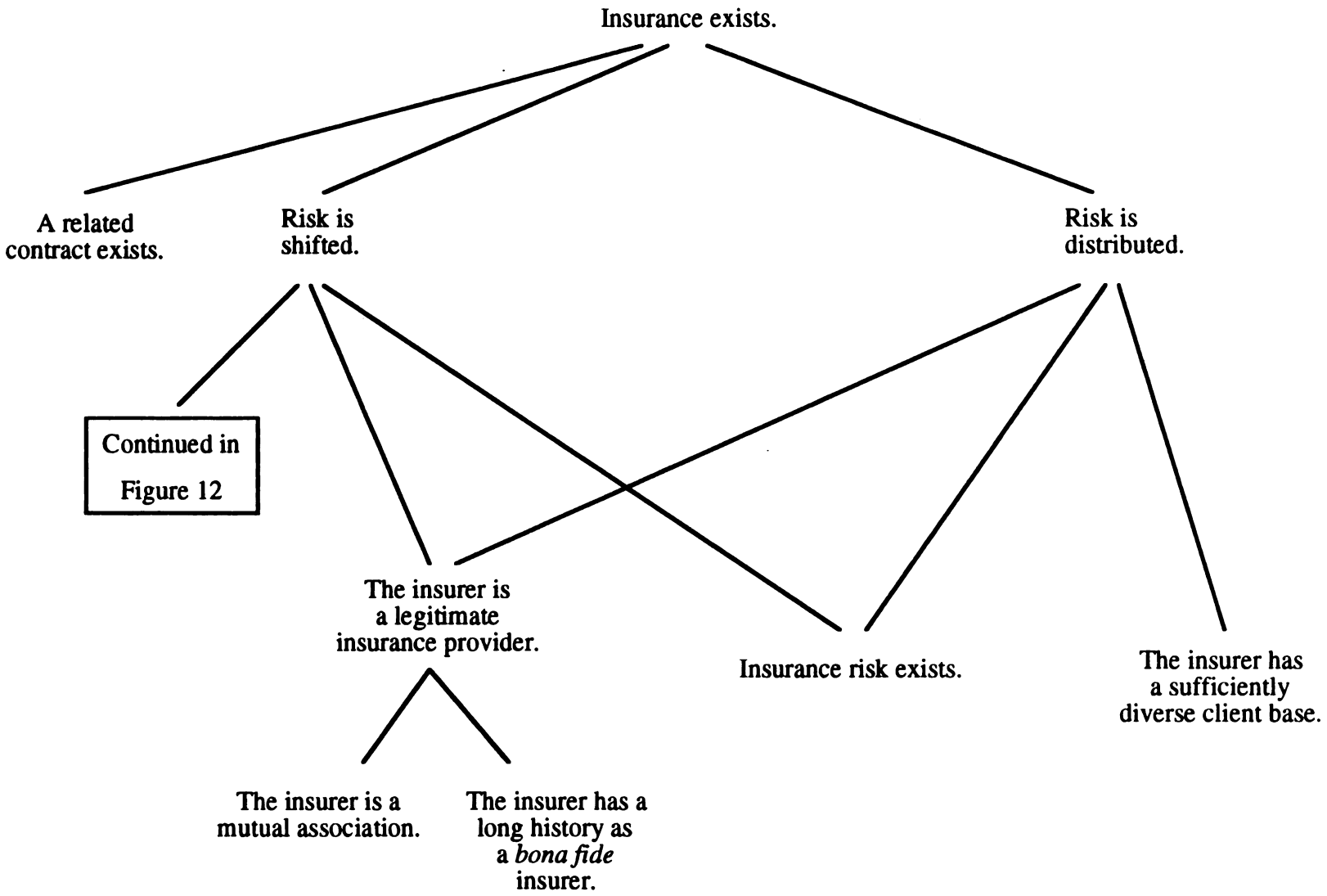


**Figure 32.** CRISTA's Issue Composition Hierarchy (Part 1)

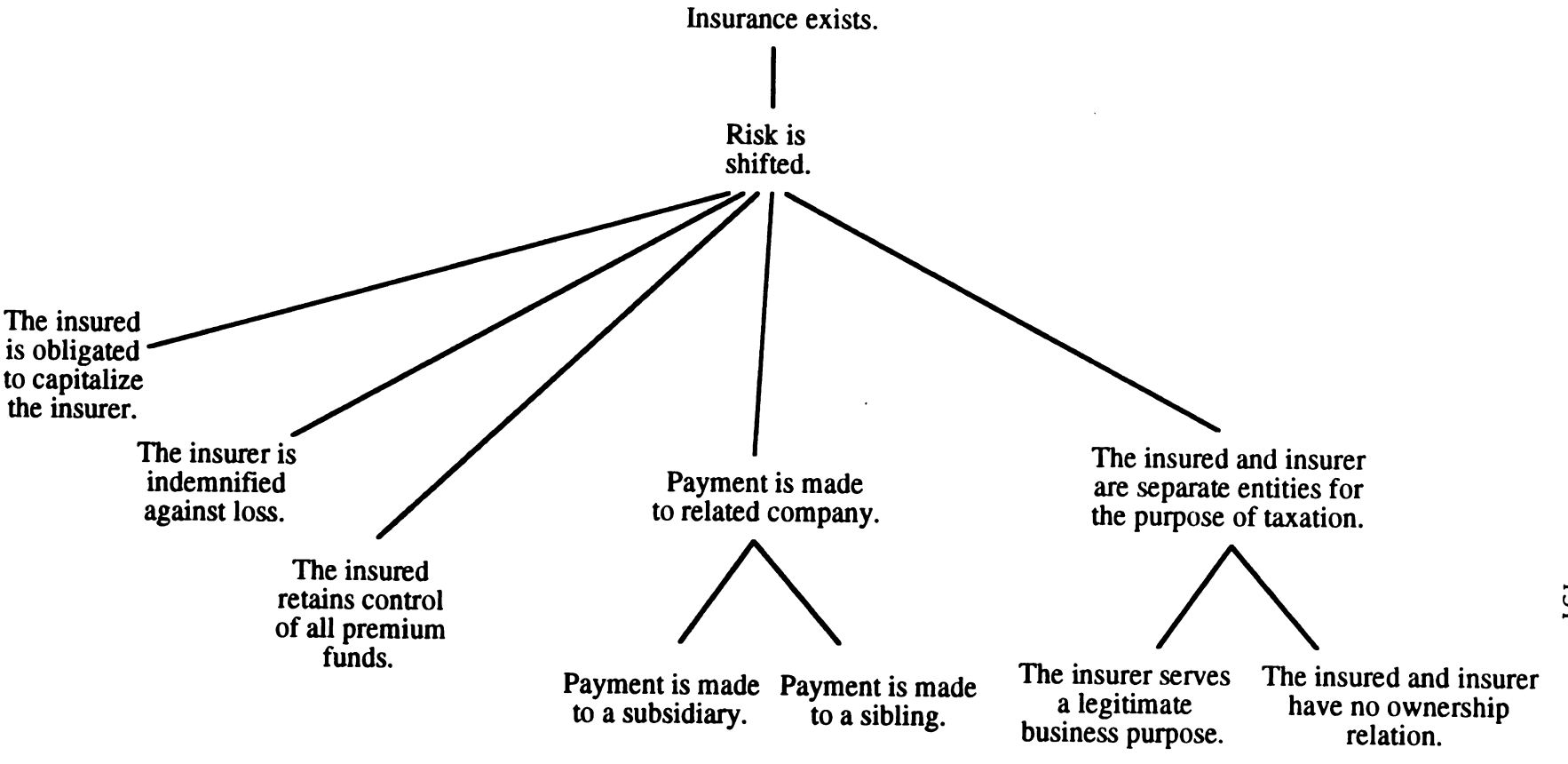


**Figure 33.** CRISTA's Issue Composition Hierarchy (Part 2)

and case functions. At this point, the ICH is relatively stable; addition of the four cases most recently decided in the domain to the Case Memory resulted in the addition of only one new node to the hierarchy. Unless the courts adopt a radically new course in deciding captive insurance issues, the existing ICH will likely remain as an accurate model of issues in the domain.

The division between assertions that need to be justified and assertions which can be abstracted using "compiled" domain knowledge is not always distinct. Several of the fact abstraction agents in CRISTA — for instance, the matchers for the existence of insurance risk and legitimate business purpose — are also still the subject of debate in the domain of captive insurance. For each of these issues, there exists some compiled knowledge for identifying the particular instances of the abstraction in case data. This knowledge is incomplete, in the sense that it is inconclusive in some fact situations and still unsettled for other classes of situations. Consequently, some cases justify assertions regarding these abstractions as if they were any other issue in the domain. CRISTA accounts for this duality by including fact abstractors for these concepts as well as nodes corresponding to the concepts in the ICH. Thus CRISTA is able to rely on compiled knowledge for recognizing the abstractions when appropriate while retaining recourse to case-based justifications when they are more appropriate.

## 7.3 Samples of CRISTA's Problem Solving

Given this summary description of CRISTA, its knowledge content, and the pertinent details of its implementation, one can now consider how the program functions in the context of specific problems. This section follows two traces of CRISTA in action. In the first, CRISTA interacts with a user who presents a case similar to Humana versus the Commissioner of Internal Revenue [1989]. This trace demonstrates CRISTA's behavior in

a relatively simple interaction. In the second, CRISTA adds a new case to its case memory, The Harper Group versus the Commissioner of Internal Revenue [1991].

Note that these samples are not intended to provide an all-inclusive view of CRISTA's capabilities. The full specification of these appears in Chapters 4 through 6, in terms of the Generic Task problem solvers from which CRISTA is built. Additionally, implementation-specific details for CRISTA appear in the preceding section. Rather, these samples aim to illustrate CRISTA's behavior in two typical circumstances involving conceptual retrieval.

### 7.3.1 Sample Problem #1

The first sample problem involves a case very similar to *Humana* (Table 24). It differs from *Humana* in the presence of two additional facts: the captive insurer is incorporated in Bermuda, and it is licensed as an insurer in only three states. These facts may weaken the plaintiff's case in that they shed doubt on the belief that the captive insurer has a legitimate business purpose outside of tax avoidance. The user seeks cases that will support the claim that insurance exists between the captive subsidiary and its sibling companies, the other subsidiaries of the captive's parent. This trace follows the steps of the control strategy outlined in Table 21.

The user initiates a session with CRISTA (Step 0) by requesting cases that will support an assertion. The Case Memory forwards this request to the Situation Data Base (Step 1), which is responsible for interaction with the user. The Data Base requests the specific assertion, InsuranceExists(Sibling, Captive), and the initial case data. The user enters the data listed in Table 25 by selecting the appropriate questions from a menu and then selecting or typing the answers. At this point, the desired assertion is returned to the Case Memory.

**Table 25.** The Facts of Sample Problem 1

| | |
|---|---|
| ClientRole | Plaintiff |
| DoesParentDirectlyOwnSubsidiary | Yes |
| PercentOfOwnershipInterest | 100 |
| DoesSubsidiaryInsureSiblings | Yes |
| PercentOfRevenuesFromOutsideRisks | 0 |
| IsInsurerFullyLicensed | Yes |
| IsInsurerRegulated | Yes |
| IsThereACapitalizationAgreement | No |
| NumberOfJurisdictionsLicensed | 3 |
| JurisdictionOfInsurer | Bermuda |

Since the query is for a justification, the Memory invokes its match-and-decompose method for the ICH (Step 2). The assertion, InsuranceExists(Sibling, Captive), matches at the root node of the hierarchy. The node's similarity matcher identifies six cases in the memory in which the decision favored the plaintiff and in which insurance was found to exist (that is, in which the issue was decided positively). These six cases, listed in Table 26, are made available to the user in a browser window, from which any of the cases' functional representation may be viewed (Step 3).

**Table 26.** Cases Returned by CRISTA on the First Pass of Sample 1

AMERCO versus the Commissioner of Internal Revenue [1991]
Crawford Fitting Company versus the United States [1985]
The Harper Group versus the Commissioner of Internal Revenue [1991]
Humana versus the Commissioner of Internal Revenue [1989]
Sears and Roebuck versus the Commissioner of Internal Revenue [1991]
Weber Paper Company versus the United States [1962]

However, in the course of retrieval, the node's fact similarity matcher further distinguishes *Crawford* from the current case, because it does not involve insurance among siblings. Likewise, it distinguishes *Weber*, because it involves a mutual insurance

arrangement, and *AMERCO*, *Harper*, and *Sears* on the basis of the degree of ownership and outside risks present in those cases. Each of these distinctions relied on a fact abstraction. When the similarity matcher came to such an abstraction, it requested a value for the abstraction from the Situation Data Base. Since these abstractions had not yet been computed in the current case, the Data Base sent requests to the appropriate fact abstraction agents seeking their values.

The user sees that only *Humana* was not significantly distinguished from the current situation. The selection of *Humana* as the most similar relevant case is denoted explicitly in the browser's presentation of the cases. The differences between the Humana case and the current case are deemed by the similarity matcher to be less serious than the differences from the other five cases. The user examines the top-level justification of *Humana* with respect to sibling insurance. In the Humana case, the target assertion is justified by reliance on the claim that the risk of loss was both shifted and distributed in the Humana fact situation. Thus, the user decides to find cases that will help show the presence of risk shifting and risk distribution in the current case.

The user asks CRISTA to retrieve cases that will help to justify the assertion that risk is shifted in the current situation (Step 1). Invocation of the match-and-decompose method on this iteration finds that the root node of the ICH returns only a subissue match. The node *Insurance exists* cannot exactly match the query RiskShifted(Sibling, Captive) because the query does not deal directly with the existence of insurance. However, the node's match knowledge recognizes the predicate RiskShifted as being relevant to one of its subissues. So the node decomposes itself and asks its subnodes to try to match the query. Working from left to right in the hierarchy, the node *A related contract exists* first tries to match the query but fails. Then the node *Risk is shifted* tries and achieves an exact match (Step 2).

CRISTA reports that five of the previously-retrieved cases (all but *Weber*) are again relevant to the target assertion. And, once, again, the Humana case is judged to be the most similar of the relevant cases. The five cases are displayed in a browser window, with Humana highlighted as the most similar case (Step 3). Examination of the Humana case's justification for the claim of risk shifting reveals that all its preconditions are either facts of the current case (for example, DoesSubsidiaryInsureSiblings = Yes) or abstractions of those facts (for example, OwnershipRelation(Insurer,Insured) = None). As a result, this justification can be applied to the situation at hand. A similar sequence of requests and retrievals follows the user's request for cases to help justify the assertion that risk is distributed in the current situation.

**Table 27.** CRISTA's Case Indexing Algorithm

GIVEN: A case represented in the FR

1. Add the case to memory.
2. Add the case identifier, and a pointer to the case, to the case citation index.
3. For each function of the case:
   a. Apply the match-and-decompose algorithm to find an exact match in the issue composition hierarchy.
   b. If an exact match is found, then add the case and function to the set of cases indicated by the matching node's fact similarity matcher.
   c. If no exact match is found, then ask the user whether (1) the function should be stored at an existing node or (2) the function represents a new issue.
      i. If the function should be stored at an existing node, store the case and function as in Step 3b.
      ii. If the function represents a new issue, create a new node as directed by the user, and store the case and function as in Step 3b.

### 7.3.2 Sample Problem #2

In the second sample problem, CRISTA adds a new case to its case memory, Harper Group versus the Commissioner of Internal Revenue [1991]. This trace follows CRISTA's case indexing algorithm (Table 27), first introduced in Chapter 6. This algorithm takes as input a case represented in the FR for cases. Table 28 presents the FR case description for *Harper*.

The first two steps of the algorithm are trivial. In CRISTA, whenever a functional representation of a case is created, this FR object becomes an item in the CRISTA's Smalltalk image. Thus, having created the FR for *Harper* means that the case is now in memory (Step 1). Adding *Harper* to the case citation index (Step 2) merely requires that a new key-value pair be added to the Smalltalk dictionary that implements the index. The key is Harper-91, the case identifier, and the value is a pointer to the FR object that represents *Harper*.

Step 3 of the algorithm involves the addition of *Harper*'s three functions to the issue composition hierarchy. For each function, indexing proceeds in a similar fashion. First, apply the match-and-decompose algorithm to find the appropriate issue node. Each of *Harper*'s functions will find an exact match in the ICH — at nodes *Insurance exists*, *Risk is shifted*, and *Risk is distributed*, respectively. Second, invoke the node's fact similarity matcher to determine the category of cases in which *Harper* most appropriately belongs. If the pattern matcher finds a match at a high enough level of confidence, index *Harper* in that group of cases; otherwise create a new case group and a new fact pattern in the matcher.[6] The indexing of *Harper* at this second stage depends on the current state of CRISTA's case memory. If either of two cases — AMERCO [1991] or Sears [1991] — is already a part of the case memory, then *Harper*'s functions are placed in the same group of cases. If

---

6 The creation of an appropriate fact pattern and the corresponding update to the fact similarity matcher requires the assistance of a knowledgeable user. The system has no facility for identifying the most predictive or relevant facts from which to form a "good" patt

**Table 28.** The Case Description for Harper [1991]

Case Harper-91

**Context**

**Legal Documentation**

| | |
|---|---|
| **Plaintiff** | The Harper Group and includible subsidiaries |
| **Defendant** | Commissioner of Internal Revenue |
| **Citation** | 96 TC No. 4 |
| **Date** | January 24, 1991 |

**Procedural Context**

| | |
|---|---|
| **Setting** | U.S. Tax Court<br>Plaintiff challenges deficiencies assessed on payments to a wholly-owned captive insurer. |
| **Outcome** | In favor of plaintiff. |

**Facts**

Plaintiff owns insurer indirectly through subsidiaries.
Subsidiary meets all licensing and regulatory requirements.
Subsidiary provides insurance to plaintiff and its affiliates.
Subsidiary provides insurance to unrelated parties.
Subsidiary provides variety of insurance coverages.
Subsidiary was fully capitalized at creation.
Premium rates determined by industry-standard pricing.
Percentage of revenue from unrelated companies = 30%.
Percentage of revenue from sibling companies = 25%.

**Issues**

| | |
|---|---|
| **Question** | Do payments made by an entity and its subsidiaries to a wholly-owned subsidiary constitute deductible insurance premiums? |
| **Holding** | Yes. |
| **Function** | Determine Insurance |
| **Given** | Risk is shifted & Risk is distributed. |
| **Conclude** | Payment constitutes insurance. |
| **By** | Determine Insurance Justification |
| **Function** | Determine Risk Shifting |
| **Given** | Subsidiary was fully capitalized at creation... |
| **Conclude** | Risk is shifted. |
| **By** | Determine Risk Shifting Justification |
| **Function** | Determine Risk Distributing |
| **Given** | Percentage of unrelated revenue = 30%. |
| **Conclude** | Risk is distributed. |
| **By** | Determine Risk Distributing Justification |

neither of these cases is already in memory, then a new group of cases is created for *Harper*. In the latter case, subsequent addition of *AMERCO* or *Sears* to CRISTA will result in their being indexed into *Harper*'s case set.

## 7.4 Conclusion

This chapter describes CRISTA, a computer program that implements the ideas proposed in the preceding chapters. CRISTA embodies a model of conceptual retrieval based on the roles that prior cases can play in justifying assertions. The functionality and content of the case memory are guided by two central ideas: the problem solving context in which the case memory will be used, and the functional understanding of past justifications as abstract devices. These ideas have, in turn, greatly affected design decisions made during the development of CRISTA.

# CHAPTER 8

# COMPARISONS TO RELATED WORK: EXTENSIONS AND ELABORATIONS

## 8.1 Introduction

The preceding four chapters introduce an approach to conceptual retrieval based on the idea that justifications can be understood as abstract devices. Those chapters propose:

- a problem solving architecture, in the technical sense of Sticklen [1990], for the task of case-based legal justification,
- a representation for justifications based on the Functional Representation of devices [Sembugamoorthy and Chandrasekaran 1986], and
- a model of conceptual retrieval for the task of justification that is based on this representation.

These proposals are made in the context of two analyses — an investigation of various approaches to the problem of conceptual retrieval (Chapter 2), and an examination of the task of justificatory legal analysis in taxation (Chapter 3).

This chapter evaluates the ideas proposed in this thesis with respect to related research in AI and other disciplines. Particular attention is paid to how this work extends and elaborates ideas found in existing approaches. The remaining sections of the chapter are organized according to the two central contributions proposed in this work: a functional representation of legal justifications, and a task-directed model of conceptual retrieval.

## 8.2 A Functional Representation of Legal Justifications

In large part, the use of the Functional Representation to model legal justifications was motivated by two ideas. First, specialists in the law seem to organize their understanding of legal cases in a way that relates new justifications to past cases, to the main uses of cases in precedent-based reasoning, and to the concepts and statutes of a particular domain. This idea was one of the original intuitions underlying the development of the Functional Representation, in the context of understanding engineering devices [Sembugamoorthy and Chandrasekaran 1986]. Second, the work of Toulmin [1958] on the characterization of justificatory arguments bears a strong resemblance to the graphical representation of behaviors in the FR. This idea provided an explicit bridge to the use of the FR for understanding legal cases as abstract devices.

The case representation proposed in Chapter 5 raises several issues of interest. Two such issues are discussed here: the representation of justificatory arguments, and the issue of the appropriate level of granularity in case representation.

### 8.2.1 Representation of Justificatory Arguments

*Warrants* With the syntactic extensions to the FR outlined in Chapter 7, the representation of legal cases follows closely the principles that underlie the FR. The distinguishing feature of the case representation is its use of behavior links to embody justificatory warrants.

The idea of warrants is also central to many other approaches for representing arguments. Toulmin [1958] first sought to explicitly identify warrants and the other roles that an assertion can play in justification. Other researchers adopted his terminology and sought to identify classes of warrants that operate in various forms of argument. Birnbaum [1982] catalogued a small number of "argument molecules" that incorporated warrants for

particular beliefs in an adversarial argument. Branting [1989, 1991] has used his notion of *complementary* warrants[1] to gain leverage in automating justificatory legal analysis. Other researchers who have focused on the explicit representation of warrants in the law include Marshall [1989], Lutomski [1989], and Dick [1987].

Toward this end, the functional representation of cases offers a principled way to classify justificatory warrants in the legal domains. The three types of link pointer in the case representation are:

- By Knowledge of <some world knowledge>,
- By Justification <justification identifier>, and
- By Function <function identifier> of Case <case identifier>.

The first type allows reference to various forms of knowledge about the world. In the law, this might refer to a statute or regulation, a generalized legal rule, or a piece of common-sense knowledge outside the law. The second type allows reference to another justification in the case. Such indirection makes it possible to hide the technicalities of a justification until such time as one wishes to examine the warrant in greater detail. Finally, the third type allows direct citation of another case, and in particular direct citation of a specific line of reasoning adopted in the case.

These three types of warrant encompass those proposed by previous researchers. Birnbaum's argument molecules corresponded to **By Knowledge** pointers in the FR. While Branting recognized the need for explicit case citation in the law, his theory of complementary warrants does not differentiate among the various justificatory roles that a single case might play. Thus there is no precise sense of a case's different issues or lines of reasoning, and thus no sense of case's *function*.

---

1 Legal rules and legal cases are complementary in the sense that they can support each other in such a way that assertions not justifiable using only one type of warrant are justifiable using both in concert. More generally, Branting's complementary warrants denote compiled classification knowledge and episodic recognition knowledge.

Indeed, the richness of the FR's warrants affords further latitude in identifying meaningful citation types. One can conceive of decomposing the class of **By Knowledge** using other forms of legal knowledge. For instance, pointers to statutes could be segregated into a class of **By Statute** links. This link type would allow the user of the represented case to make direct use of the case citation in retrieving the statute from a data base. Furthermore, such a link would also facilitate access to a *structured* representation of the statute (perhaps using an FR-like formalism), thus enabling the representation and use of a case's reasoning in even greater detail.

*Function Specification* In the FR for cases, functions are specified as a ternary relation:

| | |
|---|---|
| **Given** | assumptions |
| **Conclude** | assertion |
| **By** | justification |

But in the original Functional Representation [Sembugamoorthy and Chandrasekaran 1986], functions were defined as a four-tuple:

| | |
|---|---|
| **If** | preconditions |
| **To Make** | goals |
| **Provided** | meaningful context (optional) |
| **By** | causal behavior |

The **Given/Conclude/By** triple corresponds to **If/ToMake/By** in the standard FR. But what of the **Provided** clause? Roughly, this clause specifies conditions on the state of the device that must be true in order for performance of the function to be meaningful. That is, it specifies a set of "semantic" preconditions. These conditions are not strictly necessary for the causal behavior to be performed, but they are necessary for the achieved state to be meaningful in the domain.

This clause could be used to great effect in representing justifications in the law and many other domains. Typically, a justification refers only to those assumptions that are salient to the issue in dispute. Yet any given justification will be meaningful only in a

particular context. Assertions so specified would serve as super-preconditions, in that they would have to hold in order for the justification to be persuasive. For example, in the case Humana versus the Commissioner of Internal Revenue [1989], the function Conclude Insurance might be defined as follows:

**Function** Conclude Insurance
**Given** Risk is shifted AND Risk is distributed
**Conclude** Insurance exists
**By** Conclude Insurance Behavior

There are, however, a number of other assertions that may have to hold in order for this justification to apply. The recipient of the risk may, in a tax domain, have to be a licensed and regulated insurer.

Whether a necessary assertion is classified as an assumption or as part of the semantic context of the case depends largely on the domain. In the law, this classification will also depend on the particular state of the domain, that is, the existing body of case law. As case law evolves, different case features become more or less salient and more or less the subject of adversarial debate. This shift is a necessary property of the law. Features that were once considered contextual are challenged, thus becoming part of the argument discourse as explicit assumptions (e.g., sibling companies cannot purchase insurance from one another). Conversely, features once considered part of the argument discourse as explicit assumptions become settled matters of law and thus part of the legal context (e.g., the features of risk shifting). An FR for justifications must ultimately account for this interplay, perhaps treating preconditions and contextual states identically, except for providing notational distinctions that benefit human users. This remains an interesting problem for future research.

### 8.2.2 Relationship to the Issue of Case Granularity

An important open question in the field of case-based reasoning involves determining the appropriate granularity for representing cases in memory. Past case-based systems have demonstrated the difficulties to be found in representing complete problem-solving episodes, from the initial problem to the computed solution, as individual cases. Often, only a small fragment of the problem solving done on the case will be relevant in a later situation. Representing the whole case as a single item in memory means that the whole case will have to be retrieved in the later situation, with the related computational cost of finding the appropriate fragment for application in the new situation. This approach also leads to difficulties in generalizing from comparable experiences that constitute elements of individual cases.

In order to address this problem of large case granularity, several researchers have investigated ways of decomposing cases into more appropriately-sized fragments. Sycara and Navinchandra [1989, 1991] approached this problem as one generating indices that point directly to a particular fragment within a unitary case. Redmond [1989, 1990] attacked the problem more directly, proposing a theory of case decomposition based on problem-solving goals for case-based diagnosis. Each resulting case fragment, termed a "snippet," carries knowledge of its problem-solving context, its goal, and the way the goal was achieved. These snippets are then indexed individually for retrieval when they will be most useful.

The functional representation of cases proposed here offers many of the same advantages of Redmond's snippets. Like Redmond's approach, the case FR allows direct access to case fragments — in the FR, functions and behaviors — based on the goals they achieve. Furthermore, the FR also facilitates generalization based on the goals that each case function achieves. Cases are accessible as a whole, but they can also be thought of as distributed throughout memory and accessible as functional fragments. The FR provides

some distinct advantages over Redmond's approach, though. Explicit use of the known functions of a device focuses the selection of appropriate case fragments. One would not wish to create a fragment for each assertion in the justification, but the functional approach eliminates this possibility, instead focusing on fragments that achieve particular design goals for the device. In physical domains such as Redmond's automotive trouble shooting, the FR provides the additional benefit of allowing model-based simulation of the device. This capability enables the use of mechanisms such as Sticklen's [1987] compilation of classificatory knowledge through model-based consequence finding.

### 8.2.3 Potential Impact on Legal Practice

Finally, the functional representation for cases could have a positive impact on legal practice. The FR offers a systematic means to critique a justification. By implementing Toulmin's role-based model of argument in a computational manner, the representation makes explicit commitments to the roles that assertions play in a justification. This facilitates annotation of the case's reasoning at the level of the case itself, or at the level of individual issues, justifications, and warrants. In the situation of a court deciding a case, the majority opinion of the court would be issued as the court's holding. Judges wishing to author concurring or dissenting opinions would have two options available. They could issue a functional representation of their reasoning, but they could also generate specific criticisms of the majority opinion by annotating the court's FR of that opinion. Indeed, a similar notion has been described by Lowe [1985] in the context of a user-organized information system.

## 8.3 Conceptual Retrieval based on Problem-Solving Roles

In Chapter 2, the problem of conceptual retrieval was analyzed, and past research aimed at understanding and solving the problem was evaluated. One of the central goals of

the research described in this dissertation is to propose and evaluate a model of conceptual retrieval based on problem-solving roles. Chapter 6 proposes such a model, in particular focusing on the issue composition hierarchy (ICH). In this section, the model is appraised with respect to the important issues that it addresses.

### 8.3.1 The Use of Abstract Indices

Underlying most work on conceptual retrieval is the notion of an abstract index. Such an index corresponds to some abstract feature that characterizes a case. This feature is predictive of the case's future utility; that is, the case will likely be useful in future situations characterized by the same feature. Typically, in case-based reasoning, abstract indices have consisted of "task-related goals and features of the world causally relevant to the status of these goals" [Domeshek 1991a]. Much of the research on abstract indexing of case memory has been aimed at specifying more concretely the nature and content of these two index types.

The model of conceptual retrieval implemented by CRISTA offers one approach: indices consist of assertions to be justified. Cases are organized across these indices based on their functions, their abilities to justify assertions. This approach supports Hammond's [1989] claim that abstract indices — though often objected to because they require computational resources to generate them — may already be present in the vocabulary of the problem-solving agent. CRISTA is designed as a component of a larger problem-solving architecture for legal justification. This problem solver places particular functional demands on the performance of case memory, including the need to access cases by their functions. No additional computation is expended in generating these indices, since they correspond directly to necessary elements of the problem solver's vocabulary.

### 8.3.2 Relationship to Goel and Hafner

The two primary motivations for the model of conceptual retrieval proposed here were Hafner's [1987] issue/case discrimination tree and Goel's [1989] use of functions in the FR as indices into a memory of design cases.

*Goel and Functional Indices* Goel indexed memory by the functions that each design case could deliver. In this way, cases could be retrieved when they would be most useful in generating a new design — when they were capable of fulfilling a design need. Retrieval was accomplished by matching the functional specifications of a desired design against the functional specifications of all cases in memory. The case that matched most closely was then adapted to the current need. If multiple cases matched partially, then the problem solver applied deliberative design knowledge in ordering the cases for adaptation. The matching process itself was what Goel called "intuitive", an associative, qualitative match against every case in the memory.

While this approach outlines an index vocabulary, it did not specify an index organization or an associated retrieval algorithm sufficient for handling large case memories. The computational cost of matching a structured design specification against that of every case in a large memory would soon become unacceptable. Goel suggested two possible approaches to the organization/retrieval issue. A component hierarchy of designs would capture relationships among devices and their components. A generalization hierarchy of designs would organize cases in terms of the specificity of case features along multiple dimensions of domain interest.

CRISTA's index vocabulary was motivated by Goel's use of design knowledge about devices. Once the notion of representing justifications as abstract devices developed, through consideration of Toulmin's work, the notion of using device functions — the ability to justify assertions — developed naturally. However, the tax law domain in which

CRISTA was to operate was far too large to consider the use of a total-memory retrieval algorithm. This fact led to the integration of an existing idea to make conceptual retrieval a reasonable goal.

*Hafner and Issue Discrimination* CRISTA's index organization was influenced most directly by Hafner's [1987] issue/case discrimination tree (Section 2.3.1). This mechanism related important issues in a legal domain (broker liability) according to the influence that one issue had on another in the outcome of a case. Cases in memory were organized according to the issues that characterized them and according to how these issue related to one another. Hafner's model was intended as an information retrieval mechanism, not as a component in a larger problem-solving system. As such, there was no retrieval algorithm to speak of, since retrieval was driven by a human user.

*A Synthesis* The model of conceptual retrieval proposed here merges these two lines of research. Given an index vocabulary based on case functions, the model required a means for organizing case memory in order to provide efficient and task-directed case retrieval. The idea of the issue/case discrimination tree was adapted to take into account the functional representation of cases. Analysis of the domain was now guided by the consideration of issues as the targets for justification. Thus, issues in the index organization — now called an issue composition hierarchy — were related by their specific justificatory relationships in past cases.

Because this hierarchy also resembled Goel's design component hierarchy, his proposed retrieval method — match and refine — was adapted to the ICH, in the form of the match-and-decompose algorithm of Chapters 6 and 7. This adaptation also incorporates elements of Hafner's issue discrimination method, in which issue resolution and case outcome segregate the cases stored at any issue node.

One important difference between the ICH model and those of Hafner and Goel lies in the details of case retrieval. For Hafner, discrimination between cases at each node involves attention only to the contextual factors of issue resolution and case outcome; there is no consideration of the similarities and differences between fact situations. In CRISTA's case retrieval algorithm, these contextual factors are used as the last filter on the retrieved cases, but only after an intermediate step in which cases are filtered based on their factual similarity to the situation at hand. This step is necessary to insure that factual similarity — a crucial element in the persuasiveness of legal arguments — is provided by the case memory. However, unlike approaches that emphasize factual similarity to the exclusion of functional utility in argument, this algorithm concentrates on factual similarity only in the context of task requirements.

For Goel, the retrieval process consists in matching the functional specification of the desired component with that of each case in memory. In CRISTA, cases are first matched on their goal assertion, *not* on their preconditions, and are then matched according to factual similarity and contextual factors. This omission of the preconditions reflects the nature of legal argument. As discussed in Section 8.2.1, there is often a large degree of ambiguity in which facts and assertions constitute the assumptions of a justification and which constitute domain context. Rather than rely on uncertain assumptions, search of the ICH considers only the target assertion and the facts of the case. Distinctions based on the preconditions of an argument can be made by the problem solver.

Ultimately, the proposed model of conceptual retrieval elaborates the ideas of Goel and Hafner. The model investigates some of the issues left open by Goel with regard to memory organization and retrieval. Moreover, the model provides a principled means for generating Hafner's issue/case discrimination tree. Though the notion of case functionality is not made explicit, Hafner's approach implicitly reflects compositional relationships among domain issues. These relationships deal with the ability of a case to assist in

justifying similar assertions in a new circumstance. The issue/case discrimination tree does incorporate more general semantic relationships among its issues nodes than does the ICH, but this is mostly a result of developing the model outside the context of a specific problem-solving task.

## 8.4 Conclusion

This chapter compares the research results reported in Chapters 4 through 7 to related work in AI and information retrieval. The functional representation of justifications provides its greatest contribution in elaborating the idea of a *warrant* for inference. By extending the notion of warrants to explicit citation of device functions and behaviors, this representation establishes the basis for a case memory that is organized according to roles in justification. This FR also offers some answers to questions regarding the most appropriate granularity of case representation. Finally, one shortcoming in the representation is discussed: the lack of a function definition slot to hold assertions that are necessary in order for the justification to be meaningful. This shortcoming points to a potential avenue for extending this work.

Second, the model of case memory is shown to be a synthesis of ideas from Hafner's work on issue/case discrimination trees and Goel's work on the representation and indexing of designs in a case memory using a Functional Representation. This synthesis extends both lines of research toward the idea of conceptual retrieval by specifying an index vocabulary, index organization, and retrieval algorithm based on the functional representation of cases.

# CHAPTER 9

# CONCLUSION

## 9.1 Introduction

This chapter concludes the dissertation with a discussion of the contributions made by this work and avenues for future research. The final section of the dissertation offers a few remarks regarding the ultimate impact of this research.

## 9.2 Contributions of this Research

In short, the primary goal of this research was to develop a theory of conceptual retrieval from a problem-solving perspective — how do capable problem solvers, with their knowledge of a task and a domain, retrieve cases relevant to a problem-solving situation? Efforts aimed at answering this question have led to contributions in three broad areas:

- the modeling of legal justifications as abstract devices,
- the development of a model of conceptual retrieval based on this view of justifications, and
- the extension of Generic Task theory.

Contributions in each of these areas are discussed further below.

### 9.2.1 Modeling Justifications as Abstract Devices

Viewing a justification as an abstract device offers insight into the understanding of both justifications and the Functional Representation. Toulmin recognized the benefit of explicating the roles that assertions can play in justificatory argument, and the functional

representation of cases makes these roles explicit. The FR provides greater benefit, though, in its treatment of case citation and warrants. Through case citation, justifications can be represented at multiple levels of detail. This sort of layering is a natural part of argumentation in many domains, but its nature is made transparent in the functional representation of arguments. By providing a well-developed set of link types, the FR offers a rich vocabulary for expressing and manipulating warrants.

Furthermore, the application of the FR to the modeling of legal justifications can also contribute to the understanding of the Functional Representation itself. Considering the demands of new types of devices on the FR forces re-examination of representational commitments. Concepts such as hypothetical preconditions can be incorporated into the language only after determining their meaning within the framework of the FR.

### 9.2.2 Modeling Conceptual Retrieval Based on Problem-Solving Roles

This research contributes three new ideas in the area of conceptual retrieval. First, the representation of justifications as devices facilitates the development of a methodology for indexing and organizing case memory. This methodology relies both on the case representation and on an understanding of how cases in the domain relate to each other, in terms of their functionality in justifications. Second, this model of conceptual retrieval is integrated with a problem solving architecture for generating legal justifications. Indeed, the model owes much of its analytic power to the fact that it was developed with the problem solver in mind, allowing the functional demands of the task to shape and guide the development of the index organization and retrieval algorithm. Third, this theory integrates indexing on the basis of surface features with indexing based on abstract functional features. This integration benefits from the utility of feature-based retrieval but only within the context of providing retrieval in support of a particular problem-solving goal.

### 9.2.3 Extending Generic Task Theory

This research extends Generic Task theory in two ways. The integration of a case memory with a problem solving architecture of Generic Task agents demonstrates the natural relationship between the deliberative problem solving of generic tasks, on the one hand, and the recognition and retrieval of past cases, on the other. By incorporating case memory with other forms of "compiled" problem-solving knowledge, systems built of Generic Task agents are capable of providing a wider range of problem-solving behaviors.

The application of Generic Task analysis to the general method of case-based reasoning begins to illustrate the utility of GT constructs in explaining and building case-based systems for routine tasks. This research attempts to understand how Generic Task problem solvers can be used to model the subtasks of case characterization and adaptation. More to the point, the proposed model of conceptual retrieval relies directly on the problem-solving roles established by a Generic Task problem solver. The abstract vocabulary of task-specific problem solving offers clear advantage over *ad hoc* methods for explaining the relationship among problem-solving type and memory organization.

## 9.3 Avenues for Future Research

Even with its diverse contributions, the research reported here raises as many new questions as it answers. These new questions form the basis of a diverse program of research aimed at extending contributions in various directions and overcoming some of the difficulties that arise in applying current results. These avenues for future research are outlined with respect to the results from which they derive: the functional representation of cases, the model of conceptual retrieval based on problem-solving roles, and the problem solving architecture for legal justification.

### 9.3.1 The Functional Representation of Cases

Research into the use of the Functional Representation for modeling justifications has certainly not been completed here. At least three important issues with regard to the FR language remain to be addressed. First, this version of FR for cases treats several types of **By Knowledge** link in the same manner, when in fact the indicated knowledge differs significantly across type. In particular, one could imagine representing statutes and tax regulations using the FR, thus enabling a more meaningful use of such citations. A new kind of link, **By Statute**, could be given independent status and used in a way analogous to the use of the **By Function ... of Case** link. It is an open question whether the FR would provide the same benefits for representing legal statutes as it does for representing legal justifications. Intuitively, though, similar benefits seem possible.

Second, the idea of *negation* is poorly developed in this version of the FR. Under what conditions can one use the negative of an assertion (say, "Risk is *not* shifted") as evidence for justifying the negation of another assertion (say, "Insurance does *not* exist.")? The issue composition hierarchy relates assertion types based on their utility for justifying other types, but this formalism makes no commitments regarding negative states or negative assertions. Hafner [1981] proposed a particularly well-developed sense of negation in her Legal Information Retrieval System. A similar approach applied to the FR might greatly increase its robustness across different scenarios.

Third, as noted in Chapter 8, the FR for cases does not include a slot for semantic preconditions of the meaningfulness of functions. The **Provided** slot available in the original FR may or may not be useful in a legal domain, since the fluidity of domain concepts might mean a shift over time in what is considered contextual information and what is considered assumptive information. An attempt to apply the **Provided** clause in such a domain could offer great insight into the nature of legal concepts and their evolution.

### 9.3.2 The Model of Conceptual Retrieval

The methodology for indexing, organizing, and retrieving cases based on the roles they can play in problem solving is a new one and has only been applied to the abstract device of justification. One can argue for the generality of such a methodology across tasks and domains, but only by applying it to various tasks and domains can one gain a true sense of the power or ineffectiveness of the methodology. Thus, one potential avenue of research available is the one that has been applied to all the generic tasks: apply the model under a variety of conditions. These conditions might include application to other domains in which the FR has proven analytically useful, such as engineering devices, human body physiology, and ecological biosystems. One particularly important question involves the scalability of the approach to larger case law domains. In Chapter 7, it was argued that the issue composition hierarchy of approximately twenty nodes would likely be sufficient to account for many more cases in the domain of captive insurance. This claim was based on analysis of twenty-five cases from the domain. Empirical evidence for the scalability of the ICH methodology will produce a more persuasive claim.

### 9.3.3 The Problem Solving Architecture

The CRISTA program implemented only three of the four modules in the proposed Generic Task architecture for legal justification. Many interesting research issues involve the routine designer for justification assembly. The nature of the knowledge required for evaluating, critiquing, and modifying a candidate justification was specified in Chapter 4, but the exact content of this knowledge remains open. Furthermore, testing CRISTA's case retrieval performance more carefully would be facilitated by having a predictable and capable problem solver available for experimentation.

One experiment of particular interest would be to test CRISTA's case memory against alternative methodologies. One can conceive of building a copy of Ashley's HYPO

program and measuring its retrieval performance on the cases in CRISTA's memory. These measurements could then be compared to CRISTA's performance on the same case set. Such an experiment could be run under varying conditions (the density of cases at each node in the ICH, the number of dimensions available to HYPO, etc.) that would permit some judgment regarding the programs' respective strengths and weaknesses.

### 9.3.4 Practical Matters

In addition to the theoretical issues that surround various elements of this research, several practical issues also arise. To the extent that the model of conceptual retrieval proposed here provides access to relevant cases in memory based on problem-solving goals, the memory should also prove useful as a research tool for human problem solvers. One practical issue of importance is the question of how best to provide human users with access to the system. The issue composition hierarchy itself offers a starting point, being a graphic representation of the issues of the domain and one relevant relation among them. However, as Kolodner [1991] argues, many ergonomic and interface factors play a role in determining how useful a computer-aided memory will be for expert problem solvers as they perform their tasks. Research aimed at discovering the best way to provide such assistance would add greatly to the practical utility of this model of conceptual retrieval.

Of graver concern is the problem of creating a realistic case memory. For a small legal domain, such as that of captive insurance, historic cases can be encoded by hand into the functional representation. And a relatively small number of cases (say, one hundred) would suffice for many knowledge-based systems applications in narrow domains. But for more complete coverage of the law — if one wished to extend CRISTA from one hundred cases to thousands of cases — the approach of hand-encoding cases becomes untenable. The same problem arises in trying to encode new cases as they are generated by the courts.

Research addressing this problem could take one of two forms. First, one could provide assistance to the human encoder, whose job is one of abstracting a case into the FR notation. This form of assistance is already a part of many legal reporting services, such as Westlaw, in which cases are presented with "headnotes" or other external documentation. Support for the encoding task might take the form of an interactive but constrained dialogue with the case memory, which guides the user in filling a "template" for FR notation. The functional representation is considerably more complex than typical headnotes, though, and so providing encoding support may require more sophisticated techniques than those currently available.

Second, and more speculative, is the idea of providing a "sketchy parser" [DeJong 1979] capable of parsing legal cases from full-text into the FR format. Even for restrictive domains such as captive insurance taxation, this approach is far removed from current natural-language processing capabilities. A sketchy parser would still require some human involvement in the encoding process, but it could perform some of the more tedious and routine parsing tasks. Ultimately, if approaches to conceptual retrieval requiring special-purpose knowledge representations are ever to become practical, this sort of research must first be explored.

## 9.4 Final Discussion

The arrival of the Information Age has indeed changed how we approach many knowledge-intensive tasks. Yet the problem of conceptual retrieval — the problem of locating in a large data base only those items relevant to a particular semantic topic of interest — stands as an impediment to the efficient use of large computer memories. This research has sought to address the conceptual retrieval problem from a task-specific perspective. The result has been the identification of an indexing methodology that is intimately tied to a particular problem-solving task and a particular case representation. In

essence, this work marries a task-specific theory of problem solving with case-based reasoning ideas about indexing to achieve a more complete picture of conceptual memory.

The two major results of this work are a functional representation of justifications and a model of conceptual retrieval based on this representation. With respect to other related disciplines, the central contribution of this research may lie in its description of how one can employ knowledge of a device and its teleology in constructing more effective and efficient case memories.

# APPENDIX

# APPENDIX A

## LEGAL CASE REFERENCES

**AMERCO-91**. AMERCO and Subsidiaries, Republic Western Insurance Company, versus Commissioner of Internal Revenue (1991), 96 TC 18.

**ASMG-85**. Anesthesia Service Medical Group, Inc., versus Commissioner of Internal Revenue (1985), 85 TC 1031

**Beach-86**. Beach Aircraft Corporation versus United States (1986), 86-2 USTC 9601, 797 F.2d 920 (10th Circuit)

**Carnation-81**. Carnation Company versus Commissioner of Internal Revenue (1981), 81-1 USTC 9263, 640 F.2d 1010 (9th Circuit)

**Clougherty-87**. Clougherty Packing Company versus Commissioner of Internal Revenue (1987), 87-1 USTC 9204, 811 F.2d 1297 (9th Circuit)

**Consumers-60**. Consumers Oil Corporation of Trenton, New Jersey, versus United States (1960), 188 F.Supp. 796, 61-1 USTC 9124 (D.C. New Jersey, 1960)

**Crawford-85**. Crawford Fitting Company versus United States (1985), 85-1 USTC 9189, 606 F.Supp. 136 (N.D. Ohio 1985)

**Gulf-87**. Gulf Oil Corporation versus Commissioner of Internal Revenue (1987), 89 TC 1010

**Gulf-90**. Gulf Oil Corporation versus Commissioner of Internal Revenue (1990), 90-2 USTC 50,496, 914 F.2d 396 (3rd Circuit)

**Harper-91**. The Harper Group and Includible Subsidiaries versus Commissioner of Internal Revenue (1991), 96 TC 45

**Humana-87**. Humana, Inc., versus Commissioner of Internal Revenue (1987), 88 TC 197

**Humana-89**. Humana, Inc., versus Commissioner of Internal Revenue (1989), 89-2 USTC 9453, 881 F.2d 276 (6th Circuit)

**LeGierse-41**. Helvering versus Le Gierse (1941), 312 U.S. 531, 41-1 USTC 10,029

**Mobil-85**. Mobil Oil Corporation versus United States (1985), 85-2 USTC 9585, 8 Cl.Ct. 555

**Moline-43**. Moline Properties, Inc. versus Commissioner of Internal Revenue (1943), 319 U.S. 436, 43-1 USTC 9464

**NatlCarbide-48**. Commissioner of Internal Revenue versus National Carbide Corporation (1948), 167 F.2d 304, (2nd Circuit)

**NatlCarbide-49**. National Carbide Corporation versus Commissioner of Internal Revenue (1949), 336 U.S. 422, 49-1 USTC 9223

**Ocean-91**. Ocean Drilling & Exploration Company versus United States (1991), 90-80T.

**Sears-91**. Sears, Roebuck, and Co., and Affiliated Corporations versus Commissioner of Internal Revenue (1991), 96 TC 61.

**Spring-30**. Spring Canyon Coal Co. versus Commissioner of Internal Revenue (1930), 43 F.2d 78, 2 USTC 574 (10th Circuit)

**Stearns-84**. Stearns-Roger Corporation versus United States (1984), 84-1 USTC 9165, 557 F.Supp. 833 (10th Circuit)

**Stearns-85.** Stearns-Roger Corporation versus United States (1985), 85-2 USTC 9712, 774 F.2d 414 (10th Circuit)

**Steere-78**. Steere Tank Lines, Inc. versus United States (1978), 78-2 USTC 9605, 577 F.2d 279-280 (5th Circuit)

**Treganowan-50**. Commissioner of Internal Revenue versus Treganowan (1950), 183 F.2d 288, 50-1 USTC 10,770 (2nd Circuit)

**Weber-62**. Weber Paper Company versus United States (1962), 204 F.Supp. 394, 62-1 USTC 9423 (W.D. Missouri)

# BIBLIOGRAPHY

# BIBLIOGRAPHY


Allemang, D. (1990) Understanding Programs as Devices. Ph. D., Ohio State University.

Ashley, K. D. (1985). Reasoning by Analogy: A Survey of Selected AI Research with Implications for Legal Expert Systems. In C. Walter (Eds.), Computing Power and Legal Reasoning (pp. 105-127). St. Paul, Minnesota: West Publishing Co.

Ashley, K. D. (1989). Toward a Computational Theory of Arguing with Precedents: Accomodating Multiple Interpretations of Cases. In International Conference on Artificial Intelligence and Law, 1 (pp. 93-102). Vancouver, British Columbia: ACM Press.

Ashley, K. D. (1990). Modeling Legal Argument: Reasoning with Cases and Hypotheticals. Cambridge, Massachussetts: MIT Press.

Bareiss, R. (1989). Exemplar-Based Knowledge Acquisition. San Diego: Academic Press.

Belew, R. K. (1987). A Connectionist Approach to Conceptual Information Retrieval. In First International Conference on Artificial Intelligence and Law, 1 (pp. 116-126). Boston, Massachussetts: ACM Press.

Bickford, H. C. (1956). Successful Tax Practice (3rd ed.). Englewood Cliffs, NJ: Prentice-Hall Inc.

Bing, J. (1978). Legal Information Retrieval Systems: The Need for and Design of Extremely Simple Retrieval Strategies. Computer/Law Journal, 1(2), 379-401.

Bing, J. (1987). Designing Text Retrieval Systems for Conceptual Searching. In First International Conference on Artificial Intelligence and Law, 1 (pp. 43-51). Boston, Massachussetts: ACM Press.

Birnbaum, L. (1982). Argument Molecules: A Functional Representation of Argument Structure. In National Conference on Artificial Intelligence, 1 (pp. 83-85). Pittsburgh, Pennsylvania: Morgan Kaufmann.

Birnbaum, L. (1989). Panel Discussion on Indexing Vocabulary. In Workshop on Case-Based Reasoning, 1 (pp. 46). Pensacola Beach, Florida: Morgan Kaufmann.

Birnbaum, L.,Collins, G.,Brand, M.,Freed, M.,Krulwich, B., & Pryor, L. (1991). A Model-Based Approach to the Construction of Adaptive Case-Based Planning Systems. In Case Based Reasoning Workshop, 1 (pp. 215-224). Washington, DC: Morgan Kaufmann.

Birnbaum, L.,Flowers, M., & McGuire, R. (1980). Towards an AI Model of Argumentation. In National Conference on Artificial Intelligence, 1 (pp. 313-315). Stanford University: Morgan Kaufmann.

Blair, D. C., & Maron, M. E. (1985). An Evaluation of Retrieval Effectiveness for a Full-Text Document Retrieval System. Communications of the ACM, 28(3), 289-299.

Branting, L. K. (1989). Representing and Reusing Explanations of Legal Precedents. In Second International Conference on Artificial Intelligence and Law, 1 (pp. 103-110). Vancouver, British Columbia: ACM Press.

Branting, L. K. (1991). Exploiting the Complementarity of Rules and Precedents with Reciprocity and Fairness. In Case Based Reasoning Workshop, 1 (pp. 39-50). Washington, DC: Morgan Kaufmann.

Brown, D. C. (1987). Routine Design Problem Solving. In J. Gero (Eds.), KBS in Engineering and Architecture Reading, Massachusetts: Addison-Wesley.

Brown, D. C., & Chandrasekaran, B. (1986, July 1986). Knowledge and Control for a Mechanical Design Expert System. IEEE Computer, p.

Buchanan, B. G., & Headrick, T. E. (1970). Some Speculation about Artificial Intelligence and Legal Reasoning. Stanford Law Review, 23(1), 40-62.

Burstein, M. H. (1989). Analogy versus CBR: The Purpose of Mapping. In Workshop on Case-Based Reasoning, 1 (pp. 133-136). Pensacola Beach, Florida: Morgan Kaufmann.

Bylander, T.,Johnson, T. R., & Goel, A. (1991). Structured Matching: A Task-Specific Technique for Making Decisions. Knowledge Acquisition, 3, 1-20.

Bylander, T., & Mittal, S. (1986, CSRL: A Language for Classificatory Problem Solving and Uncertainty Handling. AI Magazine, p. 66-77.

Chandrasekaran, B. (1983). Towards a Taxonomy of Problem Solving Types. AI Magazine, 4(1), 9-17.

Chandrasekaran, B. (1987). Towards a Functional Architecture for Intelligence Based on Generic Information Processing Tasks. In Tenth International Joint Conference on Artificial Intelligence, 2 (pp. 1183-1192). Milan, Italy: Morgan Kaufmann Publishers.

Chandrasekaran, B. (1990, Design Problem Solving: A Task Analysis. AI Magazine, p. 59-71.

Cross, G. R., & deBessonet, C. G. (1985). Representation of Legal Knowledge for Conceptual Retrieval. Information Processing and Management, 21(1), 35-44.

deBessonet, C. (1983). An Automated Intelligent System Based on a Model of a Legal System. Rutgers Computer and Technology Law Journal, 10(1), 31-58.

DeJong, G. F. (1979). A New Approach to Natural Language Processing. Cognitive Science, 3(3), 155-170.

Dick, J. P. (1987). Conceptual Retrieval and Case Law. In First International Conference on Artificial Intelligence and Law, 1 (pp. 106-115). Boston, Massachussetts: ACM Press.

Domeshek, E. A. (1991a). Indexing Stories as Social Advice. In National Conference on Artificial Intelligence, 1 (pp. 16-21). Philadelphia: Morgan Kaufmann.

Domeshek, E. A. (1991b). What Abby Cares About. In Workshop on Case-Based Reasoning, 1 (pp. 13-24). Washington, D.C.: Morgan Kaufmann.

Duer, W. M.,Horvitz, J. S., & Coberly, J. W. (1988, Captive Insurance Companies: Deductions for Premium Expenses. The Tax Adviser, p. 218-232.

Flowers, M.,McGuire, R., & Birnbaum, L. (1982). Adversary Arguments and the Logic of Personal Attacks. In W. G. Lehnert & M. H. Ringle (Eds.), Strategies for Natural Language Processing (pp. 275-294). Hillsdale, New Jersey: Lawrence Erlbaum.

Forbus, K. D., & Gentner, D. (1991). Similarity-Based Cognitive Architecture. SIGART Bulletin, 2(4), 66-69.

Gardner, A. v. d. L. (1985). Overview of an Artificial Intelligence Approach to Legal Reasoning. In C. Walter (Eds.), Computing Power and Legal Reasoning (pp. 247-274). St. Paul, Minnesota: West Publishing Company.

Gardner, A. v. d. L. (1987). An Artificial Intelligence Approach to Legal Reasoning. Cambridge: MIT Press.

Gentner, D. (1983). Structure-Mapping: A Theoretical Framework for Analogy. Cognitive Science, 7(2), 155-170.

Gentner, D. (1989). Finding the Needle: Accessing and Reasoning from Prior Cases. In Workshop on Case-Based Reasoning, 1 (pp. 137-143). Pensacola Beach, Florida: Morgan Kaufmann Publishers.

Gentner, D., & Toupin, C. (1986). Systematicity and Surface Similarity in the Development of Analogy. Cognitive Science, 10, 277-300.

Gick, M. L., & Holyoak, K. J. (1980). Analogical Problem Solving. Cognitive Psychology, 12, 306-355.

Goel, A. (1989) Integration of Case-Based Reasoning and Model-Based Reasoning for Adaptive Design Problem Solving. Ph. D., Ohio State University, Department of Computer and Information Science.

Goel, A., & Chandrasekaran, B. (1989). Using Device Models in Adaptation of Design Cases. In Workshop on Case-Based Reasoning, 1 (pp. 100-109). Pensacola Beach, Florida: Morgan Kaufmann.

Goel, A.,Kolodner, J. L.,Pearce, M.,Billington, R., & Zimring, C. (1991). Towards a Case-Based Tool for Aiding Conceptual Design Problem Solving. In Workshop on Case-Based Reasoning, 1 (pp. 109-120). Washington, D.C.: Morgan Kaufmann.

Golding, A. R., & Rosenbloom, P. S. (1991). Improving Rule-Based Systems through Case-Based Reasoning. In National Conference on Artificial Intelligence, 1 (pp. 22-27). Anaheim, California: Morgan Kaufmann.

Grady, G., & Patil, R. S. (1987). An Expert System for Screening Employee Pension Plans for the Internal Revenue Service. In The First International Conference on Artificial Intelligence and Law, (pp. 137-144). Boston: ACM Press.

Greeno, J. G. (1978). Natures of Problem-Solving Abilities. In W. K. Estes (Eds.), Handbook of Learning and Cognitive Processes (pp. 239-270). Hillsdale, New Jersey: Lawrence Erlbaum.

Hafner, C. D. (1981). An Information Retrieval System Based on a Computer Model of Legal Knowledge. Ann Arbor, Michigan: UMI Research Press.

Hafner, C. D. (1987). Conceptual Organization of Case Law Knowledge Bases. In First International Conference on Artificial Intelligence and Law, 1 (pp. 35-42). Boston, Massachussetts: ACM Press.

Hammond, K. J. (1986) Case-Based Planning: An Integrated Theory of Planning, Learning, and Memory. Ph.D., Yale University, Department of Computer Science.

Hammond, K. J. (1989). On Functionally Motivated Indexing Vocabularies: An Apologia. In Workshop on Case-Based Reasoning, 1 (pp. 52-56). Pensacola Beach, Florida: Morgan Kaufmann Publishers.

Hellawell, R. (1980). A Computer Program for Legal Planning and Analysis: Taxation of Stock Redemptions. Columbia Law Review, 80(7), 1363-1398.

Horty, J. F., Jr. (1962, March 1962). The 'Key Word In Combination' Approach. Modern Uses of Logic in Law, p. 54-64.

Kehl, W. B.,Horty, J. F., Jr.,Bacon, C., R. T., & Mitchell, D. S. (1961, September 1961). An Information Retrieval Language for Legal Studies. Communications of the ACM, p. 380-389.

Kolodner, J. L. (1991). Improving Human Decision Making through Case-Based Decision Aiding. AI Magazine, 12(2), 52-68.

Koton, P. (1988). Reasoning about Evidence in Causal Explanations. In National Conference on Artificial Intelligence, 1 (pp. 256-261). St. Paul, MN: Morgan Kaufmann.

Koton, P. A. (1990). A Method for Improving the Efficiency of Model-Based Reasoning Systems. In W. Horn (Eds.), Causal AI Models (pp. 273-282). New York: Hemisphere Publishing Corporation.

Krovetz, R. (1985). The Use of Knowledge Representation Formalisms in the Modeling of Legal Concepts. In C. Walter (Eds.), Computing Power and Legal Reasoning (pp. 275-317). St. Paul, MN: West Publishing Company.

Lehnert, W. (1987). Case-Based Problem Solving with a Large Knowledge Base of Learned Cases. In National Conference on Artificial Intelligence, 1 (pp. 256-261). Seattle, Washington: Morgan Kaufmann.

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

Lowe, D. G. (1985). Co-operative Structuring of Information: The Representation of Reasoning and Debate. International Journal of Man-Machine Studies, 23, 97-111.

Lutomski, L. S. (1989). The Design of an Attorney's Statistical Consultant. In Second International Conference on Artificial Intelligence and Law, 1 (pp. 224-233). Vancouver, British Columbia: ACM Press.

MacRae, C. D. (1985). Tax Problem Solving with an If-Then System. In C. Walter (Eds.), Computing Power and Legal Reasoning (pp. 595-620). St. Paul, Minnesota: West Publishing Company.

Marshall, C. C. (1989). Representing the Structure of a Legal Argument. In Second International Conference on Artificial Intelligence and Law, 1 (pp. 121-127). Vancouver, British Columbia: ACM Press.

Martin, C. (1989). Complex Indices: A Metaphorical Example. In Workshop on Case-Based Reasoning, 1 (pp. 295-299). Pensacola Beach, Florida: Morgan Kaufmann Publishers.

McCarty, L. T. (1977). Reflections on TAXMAN: An Experiment in Artificial Intelligence and Legal Reasoning. Harvard Law Review, 90, 837-893.

McCarty, L. T. (1983). Intelligent Legal Information Systems: Problems and Prospects. Rutgers Computer and Technology Law Journal, 9(2), 265-294.

McCarty, L. T. (1985). Permissions and Obligations. In C. Walter (Eds.), Computing Power and Legal Reasoning (pp. 573-594). St. Paul, Minnesota: West Publishing Company.

McCarty, L. T. (1987). Intelligent Legal Information Systems: An Update (Technical Report No. LRP-TR-20). Department of Computer Science, Rutgers University.

McCarty, L. T.,Sridharan, N. S., & Sangster, B. C. (1979). The Implementation of Taxman II: An Experiment in Artificial Intelligence and Legal Reasoning (Technical Report No. LRP-TR-2). Laboratory for Computer Science Research, Rutgers University.

McDougal, T.,Hammond, K., & Seifert, C. (1991). A Functional Perspective on Reminding. In Workshop on Case-Based Reasoning, 1 (pp. 63-76). Washington, D.C.: Morgan Kaufmann.

Meldman, J. A. (1977). A Structural Model for Computer-Aided Legal Analysis. Rutgers Journal of Computer and Law, 6, 27-71.

Mittal, S.,Chandrasekaran, B., & Sticklen, J. (1984, Patrec: A Knowledge-Directed Database for a Diagnostic Expert System. IEEE Computer, p. 51-58.

O'Brien, & Tung (1973). Captive Off-Shore Insurance Corporations. NYU Institute on Federal Taxation, 31, 665.

Owens, C. (1989). Plan Transformations as Abstract Indices. In Workshop on Case-Based Reasoning, 1 (pp. 62-65). Pensacola Beach, Florida: Morgan Kaufmann Publishers.

Peterson, M. A., & Waterman, D. A. (1985). An Expert Systems Approach to Evaluating Product Liability Cases. In C. Walter (Eds.), Computing Power and Legal Reasoning (pp. 627-659). St. Paul, Minnesota: West Publishing Company.

Popp, W. G., & Schlink, B. (1975). Judith, A Computer Program to Advise Lawyers in Reasoning a Case. Jurimetrics, 15(4), 303-314.

Raphael, B. (1968). SIR: A Computer Program for Semantic Information Retrieval. In M. Minsky (Eds.), Semantic Information Processing (pp. 33-145). Cambridge, Massachussetts: The MIT Press.

Redmond, M. (1989). Learning from Others' Experience: Creating Cases from Examples. In Workshop on Case-Based Reasoning, 1 (pp. 309-312). Pensacola Beach, Florida: Morgan Kaufmann.

Redmond, M. (1990). Distributed Cases for Case-Based Reasoning: Facilitating Use of Multiple Cases. In National Conference on Artificial Intelligence, 1 (pp. 304-309). Boston: Morgan Kaufmann.

Riesbeck, C., & Schank, R. (1989). Inside Case-Based Reasoning. Hillsdale, NJ: Lawrence Erlbaum Associates.

Rissland, E. L. (1985). Argument Moves and Hypotheticals. In C. Walter (Eds.), Computing Power and Legal Reasoning (pp. 129-143). St. Paul, Minnesota: West Publishing Co.

Rissland, E. L. (1990). Artificial Intelligence and Law: Stepping Stones to a Model of Legal Reasoning. Yale Law Journal, 99(8), 1957-1981.

Rosch, E. H. (1975). Cognitive Representations of Semantic Categories. Journal of Experimental Psychology: General, 104, 192-233.

Rose, D. E., & Belew, R. K. (1989). Legal Information Retrieval: A Hybrid Approach. In Second International Conference on Artificial Intelligence and Law, 1 (pp. 138-146). Vancouver, British Columbia: ACM Press.

Schank, R. C. (1982). Remindings and Memory. In Dynamic Memory Cambridge, England: Cambridge University Press.

Schlobohm, D. A. (1985). TA — A Prolog Program which Analyzes Income Tax Issues under Section 318(a) of the Internal Revenue Code. In C. Walter (Eds.), Computing Power and Legal Reasoning (pp. 765-815). St. Paul, Minnesota: West Publishing Company.

Schlobohm, D. A., & McCarty, L. T. (1989). EPS II: Estate Planning with Prototypes. In The Second International Conference on Artificial Intelligence and Law, (pp. 1-10). Vancouver, British Columbia: ACM Press.

Schlobohm, D. A., & Waterman, D. A. (1987). Explanation for an Expert System that Performs Estate Planning. In The First International Conference on Artificial Intelligence and Law, (pp. 18-27). Boston: ACM Press.

Seifert, C. M. (1988). Goals in Reminding. In Workshop on Case-Based Reasoning, 1 (pp. 357-369). Clearwater Beach, Florida: Morgan Kaufmann.

Seifert, C. M., & Hammond, K. J. (1989). Why There's No Analogical Transfer. In Workshop on Case-Based Reasoning, 1 (pp. 148-152). Pensacola Beach, Florida: Morgan Kaufmann Publishers.

Seifert, C. M.,McKoon, G.,Abelson, R. P., & Ratcliff, R. (1986). Memory Connections Between Thematically Similar Episodes. Journal of Experimental Psychology: Learning, Memory, and Cognition, 12, 220-231.

Sembugamoorthy, V., & Chandrasekaran, B. (1986). Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems. In J. Kolodner & C. Riesbeck (Eds.), Experience, Memory, and Reasoning (pp. 47-73). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Shpilberg, D.,Graham, L. E., & Schatz, H. (1986, July 1986). ExperTax: An Expert System for Corporate Tax Planning. Expert Systems, p. 136-151.

Simoudis, E., & Miller, J. S. (1991). The Application of CBR to Help Desk Applications. In Workshop on Case-Based Reasoning, 1 (pp. 25-36). Washington, D.C.: Morgan Kaufmann.

Sprowl, J. A. (1979). Automating the Legal Reasoning Process: A Computer that Uses Regulations and Statutes to Draft Legal Documents. American Bar Foundation Research Journal, 1979(1), 1-81.

Stanfill, C. (1988). Learning to Read: A Memory-Based Model. In Workshop on Case-Based Reasoning, 1 (pp. 402-413). Clearwater Beach, Florida: Morgan Kaufmann.

Sternberg, R. J. (1977). Intelligence, Information Processing, and Analogical Reasoning. Hillsdale, NJ: Lawrence Erlbaum Associates.

Sticklen, J. (1987) MDX2: An Integrated Medical Diagnostic System. Ph.D., The Ohio State University.

Sticklen, J. (1990). Problem Solving Architecture at the Knowledge Level. Journal of Experimental and Theoretical Artificial Intelligence, 1(3), 1-52.

Sticklen, J.,Chandrasekaran, B., & Josephson, J. R. (1987). Modularity of Domain Knowledge. Expert Systems: Research and Applications, 1.

Sticklen, J., & Tufankji, R. (1992). Utilizing a Functional Approach for Modeling Biological Systems. Mathematical and Computer Modeling, 16, 145-160.

Sycara, K. P., & Navinchandra, D. (1989). Index Transformation and Generation for Case Retrieval. In Workshop on Case-Based Reasoning, 1 (pp. 324-328). Pensacola Beach, Florida: Morgan Kaufmann.

Sycara, K. P., & Navinchandra, D. (1991). Influences: A Thematic Abstraction for Creative Multiple Use of Cases. In Workshop on Case-Based Reasoning, 1 (pp. 133-144). Washington, D.C.: Morgan Kaufmann.

Thagard, P., & Holyoak, K. J. (1989). Why Indexing is the Wrong Way to Think about Analog Retrieval. In Workshop on Case-Based Reasoning, 1 (pp. 36-40). Pensacola Beach, Florida: Morgan Kaufmann.

Tong, R. M.,Reid, C. A.,Crowe, G. J., & Douglas, P. R. (1987). Conceptual Legal Document Retrieval Using the RUBRIC System. In First International Conference on Artificial Intelligence and Law, 1 (pp. 28-34). Boston, Massachussetts: ACM Press.

Tong, R. M., & Shapiro, D. G. (1985). Experimental Investigations of Uncertainty in a Rule-Based System for Information Retrieval. International Journal of Man-Machine Studies, 22, 265-282.

Toulmin, S. E. (1958). The Uses of Argument. Cambridge: Cambridge University Press.

Walter, C. (Ed.). (1985). Computing Power and Legal Reasoning. St. Paul, MN: West Publishing Company.

Waltz, D. L. (1989). Is Indexing Used for Retrieval? In Workshop on Case-Based Reasoning, 1 (pp. 41-44). Pensacola Beach, Florida: Morgan Kaufmann.

Waterman, D. A.,Paul, J., & Peterson, M. (1986, October 1986). Expert Systems for Legal Decision Making. Expert Systems, p. 212-226.

Zarri, G. P. (1985). Inference Techniques for Intelligent Information Retrieval. In C. Walter (Eds.), Computing Power and Legal Reasoning (pp. 215-246). St. Paul, MN: West Publishing Company.

Zhang, X., & Waltz, D. L. (1989). Protein Structure Prediction Using Memory-Based Reasoning: A Case Study of Data Exploration. In Workshop on Case-Based Reasoning, 1 (pp. 351-355). Pensacola Beach, Florida: Morgan Kaufmann.