CHIRON: Planning in an Open-textured Domain

Kathryn E. Sanders

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-94-38
October 1994

CHIRON: Planning in an Open-Textured Domain

by
Kathryn E. Sanders
A. B., Brown University, 1977
J. D., Harvard Law School, 1982
Sc. M., Brown University, 1988

Thesis
Submitted in partial fulfillment of the requirements for the
Degree of Doctor of Philosophy in the Department of Computer Science
at Brown University.

May 1995

Copyright
by
Kathryn E. Sanders
1994

Vita

Kathryn Elizabeth Sanders was born in Oak Ridge, Tennessee on April 11, 1955. She graduated magna cum laude with a double major in English and Classics from Brown University in 1977, received a J.D. from Harvard Law School in 1982, and received a Sc.M. in Computer Science from Brown University in 1988. Her research interests include artificial intelligence and law, planning, representation of commonsense knowledge, and case-based reasoning. She is a member of AAAI, the International Society for Artificial Intelligence and Law, and the Massachusetts Bar.

Abstract

Most work in artificial intelligence and law has concentrated on modelling the type of reasoning done by trial lawyers. In fact, most lawyers' work involves planning – for example, wills and trusts, real estate deals, and business mergers and acquisitions. Certain planning issues, such as the use of underspecified, or "open-textured" rules, are illustrated especially clearly in this domain.

In this thesis, I set forth the characteristic features of planning in law, place it in the context of past artificial intelligence work in both law and planning, and describe CHIRON, a system that I have developed implementing my theory of open-textured planning in the domain of personal income tax law.

Acknowledgements

I would like to thank my advisor, Tom Dean, for giving me the freedom to pursue this project. It's hard to imagine a better thesis advisor. Leslie Kaelbling, my second committee member, read my thesis quickly, despite an amazingly busy schedule, and her insightful comments did much to improve it. The AI faculty have made Brown a great place to do artificial intelligence research, and I feel lucky to have been here.

The AI and law community has been unusually welcoming to a new researcher. In particular, Edwina Rissland, my third committee member, has been helpful beyond even what would be expected of a primary advisor. I hope to match her professionalism and intellectual enthusiasm. I would also like to thank Kevin Ashley, Karl Branting, Barbara Cuthill, Jody Daniels, Timur Friedman, Anne Gardner, Carole Hafner, Thorne McCarty, and David Skalak for helpful comments, encouragement, and enjoyable conversations.

Leora Morgenstern first suggested that I consider working in the domain of artificial intelligence and law, and supported and encouraged the beginnings of this work. Without her, it would never have happened.

Many thanks to Margaret McCartney, a tax and estate planning lawyer who took time out of her busy schedule to discuss what it would mean to have an ideal legal planning system. Her clarity of thought and practical good sense supplied a useful perspective and reminded me what the best qualities of good lawyers are as well.

The members of the Brown AI group, past and present, have made working here challenging, argumentative, and especially, fun. Among this group, I would particularly like to thank John Arnold, Ken Basye, Mark Boddy, Robert Goldman, Mary Harper, Keiji Kanazawa, Jak Kirman, Moises Lejter, Robert McCartney, Ann Nicholson, and Lynn Stein. Robert McCartney listened to many versions of these ideas, read more drafts than I can count, and provided invaluable feedback.

My office-mates provided welcome distractions, role models, and company over the years,

and let me monopolize the computer when I needed it. In addition to those already mentioned, I'd like to thank Brook Conner, Will Cook, Philip Hubbard, Richard Hughey, Sonia Leach, Richard Ravenscroft, and Andrea Skarra.

Thanks also to the exceptional administrative and technical staff in the Computer Science Department at Brown University, for maintaining software and hardware and taking care of money and bureaucratic hassles. This work would have been a lot harder without them.

This work was supported in part by National Science Foundation Presidential Young Investigator Award IRI-8957601 to Thomas Dean, by the Air Force and the Advanced Research Projects Agency of the Department of Defense under Contract No. F30602-91-C-0041, by the Office of Naval Research and the Advanced Research Projects Agency under grant ONR N00014-91-J-4052, ARPA Order 8225, by the National Science Foundation in conjunction with the Advanced Research Projects Agency of the Department of Defense under Contract No. IRI-8905436, by IBM grants 17290066, 17291066, 17292066, and 17293066, and by National Science Foundation grant IRI-8801253.

The United States government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

Contents

Vita ii

Abstract iii

Acknowledgements iv

1 Introduction 1

2 Related Work 7

2.1 Reasoning with open-textured rules 7

2.1.1 TAXMAN and EPS 7

2.1.2 Gardner 10

2.1.3 HYPO, TAX-HYPO, CABARET, and BankXX 11

2.1.4 Bench-Capon and Sergot 15

2.1.5 PROLEXS 15

2.1.6 GREBE 16

2.1.7 Projects outside of artificial intelligence and law 19

2.2 Planning 20

2.2.1 Hierarchical and nonlinear planning 20

2.2.2 Case-based reasoning and case-based planning 21

2.3 Summary 29

3 Knowledge Representation 30

3.1 Introduction 30

3.2 Statutory rules 31

3.3 Cases 33

3.4 Representing the facts 37

3.4.1 Syntax . . . 37
3.4.2 Semantics . . . 40
3.4.3 Basic concepts . . . 41
3.4.4 Axioms . . . 47
3.4.5 Facts and rules . . . 50
3.5 Prototypes . . . 51
3.6 Summary . . . 54

4 Open-textured planning **56**
4.1 Introduction . . . 56
4.2 The hierarchical planner . . . 58
4.3 Indexing, retrieval, and similarity . . . 59
4.4 Evaluating and supporting plans . . . 64
4.5 Adaptation . . . 67
4.6 Combining strategies . . . 69
4.7 Combining case-based and hierarchical planners . . . 70
4.8 Summary . . . 71

5 Implementation **73**
5.1 Introduction . . . 73
5.2 Overview . . . 73
5.3 The interaction of the hierarchical and case-based planners . . . 76
5.4 Hierarchical planner . . . 78
5.5 Case-based planner . . . 80
5.5.1 Case-analyzer . . . 81
5.5.2 Case-Positioner . . . 83
5.5.3 Select-Mopcs . . . 87
5.5.4 Argument-Builder . . . 89
5.5.5 Support-Evaluator . . . 93
5.5.6 Adapter . . . 95

6 Extended Examples **97**
6.1 The rollover strategy is recommended . . . 98
6.2 The rollover strategy is rejected . . . 114
6.3 The rollover strategy is recommended in spite of weak facts . . . 115

6.4 The rollover and §121 strategies are both recommended 119
6.5 CHIRON rejects a plan that should have succeeded 131
6.6 Summary 134

7 Evaluating CHIRON **136**
7.1 Introduction 136
7.2 Background 136
7.3 The ideal planner 138
7.4 Discussion of CHIRON 139
7.5 Summary 143

8 Conclusions and Future Work **145**

A A sample case representation: *Hughston v. Commissioner* **151**

B Text of *Hughston v. Commissioner*, T.C.M. (P-H) 50,188 (1950) **155**

C CHIRON's cases **158**

D CHIRON's factual predicates **160**

E CHIRON's property-type hierarchy and other axioms **161**

Bibliography **170**

List of Figures

1.1 §1034(a) of the Internal Revenue Code. 3

2.1 How TAXMAN would represent "Iacocca owns 100 shares of Chrysler stock." . 8
2.2 A backward-chaining rule from TAXMAN: an individual is a stockholder if some company has issued stock and he owns a piece of that stock. 9
2.3 Sample case representation from PROLEXS. 16
2.4 CHEF's recipe for broccoli with tofu. 23

3.1 Part of the plan hierarchy. 32
3.2 Examples of CHIRON's rule schemas. 33
3.3 Part of CHIRON's representation of *Hughston v. Commissioner*. 36
3.4 Part of the property-type hierarchy. 43
3.5 Transfers that may raise income tax issues. 44
3.6 The dimension "age." 53

4.1 A simple case lattice. 61
4.2 A case lattice with more facts. 62
4.3 A case lattice with facts and abstractions. 63

5.1 The Strategy-Processor in context 74
5.2 The Combiner's place in CHIRON's architecture. 77

Chapter 1

Introduction

In this thesis, I have investigated planning issues in law. The popular view of lawyers is the trial lawyer — Perry Mason, Clarence Darrow, or the lawyers on *LA Law.* Many lawyers, however, make their living planning transactions, such as the sale of a piece of property, the establishment of a trust, or the reorganization of a corporation.

Planning problems arise in law when an individual (or corporation) wants to perform a sequence of actions that raises legal issues. For example, suppose you want to sell your house. That transaction has both real estate and tax aspects. If you give a lawyer information about your goals — do you want to reinvest the proceeds in another house? — and facts relevant to the situation — how long have you owned the house? do you live in it? do you have any other residence? — the lawyer will suggest a plan or plans for achieving your goals, taking into account the legal issues raised by the transaction.

In constructing plans, lawyers generally have two types of information to work with: statutes and cases. In some domains, such as contract law, there are no statutes, and the lawyer has only case law to work with. In others, there are statutory provisions that are so recent that no cases have been decided interpreting them. In general, however, both statutes and cases must be taken into account.

In practice, lawyers often take advantage of a third source of information, plans based on past experience of similar transactions. Often, however, no such plan is available, perhaps because the lawyer has never performed this particular type of transaction before, perhaps because the law has changed so recently that no plans are available. And even where there is such a plan, if it is challenged in court, it must be justified in terms of the statutes and case law.

In this thesis, therefore, I have examined the way in which plans are developed from statutes

and cases. Specifically, I have focused on tax planning. Almost every transaction has some tax aspect, so tax planning forms part of almost all legal planning; and the way in which the statutes and cases are used in this area is typical of general legal planning.

Statutes are rules that have been created formally, by legislation. They are published by the government and often by private companies as well. For example, consider §1034(a) of the Internal Revenue Code, governing the tax treatment of the income from the sale of a personal residence, given in Figure 1.1. The Internal Revenue Code is the most important statute for United States tax planning. It contains approximately 7000 sections.

Detailed as the Internal Revenue Code is, it still contains phrases that are not defined within the statute, for example, the phrase "principal residence" in §1034. To qualify for the benefit of §1034, a taxpayer must show, among other things, that he bought and sold properties which belong to this category. These phrases are defined partly by commonsense knowledge about the meaning of the words used and partly by example, not by statute. In determining whether a transaction satisfies a given statutory predicate, or planning a transaction that will satisfy it, a lawyer must use both rules and cases.

What makes reasoning about these statutory predicates difficult — and interesting — is that defining them is not just a matter of inferring defining characteristics from a set of examples. Generally speaking, there is no set of essential characteristics shared by all positive instances of the statutory predicate. Some examples are typical, and others are more or less similar to them along various dimensions. As a result, classifying a particular object as an instance or noninstance of one of these categories is not always a simple task.

This issue arises throughout legal reasoning, not just in tax. Indeed, it is part of a general natural language problem. Many ordinary categories, such as "tiger" or "cup," are surprisingly difficult to define. This indeterminacy has been studied in linguistics and philosophy, where it is labeled *open texture* [Waismann, 1965, Hart, 1961, Lakoff, 1987]. Any planning rule expressed in natural language, such as "be careful," "never get involved in a land war in Asia," or "buy low, sell high," suffers from the same problem.

In any domain, open-textured rules can be partially defined by examples. The legal domain has the advantage that examples are recorded and published. Each court case is an example — an application of the law to a particular set of facts. Facts and results are recorded by the courts in "opinions" and published, both by the government and by private companies. Thousands of examples are readily available in any law library.

The facts of these examples can be used as the basis for new plans. Since the courts are bound by precedent, similar cases must be decided similarly. Thus, planners attempt to

§1034. Rollover of gain on sale of principal residence.
(a) Nonrecognition of gain.–If property (in this section called "old residence") used by the taxpayer as his principal residence is sold by him and, within a period beginning 2 years before the date of such sale and ending 2 years after such date, property (in this section called "new residence") is purchased and used by the taxpayer as his principal residence, gain (if any) from such sale shall be recognized only to the extent that the taxpayer's adjusted sales price (as defined in subsection (b)) of the old residence exceeds the taxpayer's cost of purchasing the new residence.

Figure 1.1: §1034(a) of the Internal Revenue Code.

construct plans that are similar (or identical) to examples of previous successful plans.

In constructing a plan based on previous cases, a planner can make use of the court's reasoning. This must be included in the case report along with the facts and result. In a case interpreting a statute, for example, the court will suggest intermediate rules connecting the facts of the case to the open-textured statutory predicates in the statute.

These rules have some predictive value. The courts are likely to follow them in later cases. They are not required to do so, however. They are free to adopt new rules, as long as the new rules are consistent with the results in all previous cases. Thus, the courts' reasoning in previous cases is useful in constructing plans, but does not make the success of a plan certain.

For example, suppose you want to construct a plan for an academic who has just spent a year on sabbatical away from home and wants to sell his house, and you have §1034 and one case to work with. Suppose that the earlier case involved the following facts:

> John and Jane Smith bought a house in Providence, Rhode Island in June, 1980, and a second house in Chatham, Massachusetts in April, 1985. On September 3, 1988, they sold the house in Providence and bought a third house in Barrington, Rhode Island two days later. They are bankers at Hospital Trust. Before the sale, they and their children lived in the Providence house most of the year; now they live in the Barrington house most of the time. They stay in the Chatham house (on Cape Cod) for two weeks during the summer and occasional weekends during the rest of the year.

Suppose the holding in that case was that the Providence house was their "principal residence" before the sale; the Barrington house became their principal residence after the sale; and so the income from the sale of the Providence house qualifies for deferral under §1034, and the court's reasoning was that if you have more than one residence, your "principal residence" is the one where you spend most of your time.[1]

[1] Like the other examples in this thesis, this one is strictly hypothetical, and not to be relied upon as tax advice.

By the reasoning in the Smiths' case, your client's house is not his principal residence, since he has not spent any time there for the past year. On the other hand, if your case comes to court, you could argue that "principal residence" should be defined as, for example, the residence closest to your job, where you are registered to vote, and from which your tax returns are filed. Moreover, this rule would be consistent with the facts of the previous case, since the taxpayers there worked in Providence, not on the Cape. Therefore, a court would be free to adopt this rule and hold in favor of your client.

The use of open-textured rules and examples has a pervasive effect on legal reasoning. It makes the legal system flexible. The fact that terms like "principal residence" are underspecified means that the courts can respond to changing circumstances. For example, they can interpret "principal residence" to cover cooperatives and condominiums, even if those forms of ownership did not exist at the time the statute was passed. Similarly, the First Amendment protection of freedom of speech can be extended to cover television and radio, as well as newspapers.

On the other hand, because the system is flexible, it is also uncertain. In law, unlike domains such as chess, it is impossible to prove a plan correct. This uncertainty is not due to lack of factual information (we can assume complete knowledge of the facts); but to the underspecified nature of the rules. Given complete information about the client's situation and the relevant law, an experienced lawyer can give an opinion about the probability of success of a given plan, but even the most conservative plan is not certain to succeed.

This uncertainty may be an unavoidable part of the legal system. Our legal system is based in part on the deeply-held belief that it is not possible to specify in advance all the possible contingencies to which a statute should apply. Beginning in the early nineteenth century, the civil law countries, such as France, Italy, and the countries of South America, tried to design a different type of legal system. Motivated by distrust of the courts, they attempted to draft self-applying statutes: legislation so complete, detailed, and clear that its application would be a predictable, automatic process. To this day, these countries emphasize statutory law and give judges a less important role than they have here, or in other countries with the Anglo-American legal tradition. Nevertheless, the attempt failed. It is still necessary for civil law courts to interpret statutes, to fill in gaps and change the meaning of the text in response to changing conditions [Merryman, 1969].

Because there is uncertainty, there is room for argument. Lawyers are trained to find support for different conclusions in a given set of cases. A large part of a lawyer's training involves learning to make arguments for and against the application of some statutory predicate,

and for and against the similarity of a previous case to the current one. This is one of the key legal skills, particularly in the Anglo-American legal system [Ashley and Rissland, 1987, Levi, 1949, Llewellyn, 1930]. Anglo-American law "requires the presentation of competing examples." [Levi, 1949, p. 5]. Although one result may be more likely than another, it is generally possible to make these arguments in both directions.

Ideally, experts in other domains would reach the same conclusion: all doctors would give the same diagnosis, all engineers would agree on how to build a building that would be safe in an earthquake, and so forth. In fact, experts in medicine, engineering, and even mathematics disagree, and find it useful to be able to argue for and against particular conclusions (see, e.g., [Lakatos, 1976]). The ability to argue for and against a particular conclusion is useful in many domains. For lawyers it is central; and as a result, the process is particularly well-illustrated in law.

Tax planning is no exception. Here, the adversaries are the taxpayers and the government. Taxpayers seek to exclude or defer items of income and deduct expenses, and the government seeks to include income and disallow deductions. Each precedent can be viewed as the execution of a plan by the taxpayer in that case, some successful (the favorable precedents) and some unsuccessful (the unfavorable ones). Taxpayers must construct plans that are similar to favorable precedents and different from unfavorable ones in some relevant way. If the similarity is too distant, or the differences are too small, the plans will be vulnerable to challenge by the government.

The open-textured rules and examples interact in interesting ways. Being reminded of a similar case directs the lawyer's attention to the rules applied in that case; being reminded of a potentially applicable rule directs his or her attention to the cases interpreting that rule. In computational terms, cases limit the search through the statutory rules. If you know of a case involving a plan similar to your own, the case report will indicate which rules were applied to that plan. Similarly, rules limit the search through the case base. If you know what statutory rule you are interested in, it is easy to retrieve exactly the cases interpreting that particular rule.

In my thesis, I have tested this theory of open-textured planning by designing and implementing CHIRON,[2] a system to solve simple problems in the domain of personal income tax planning. CHIRON solves a cluster of problems having to do with buying, selling, renting, and owning residential housing. It generates plans that satisfy the tax laws for situations that are not identical to cases already in its case base. It uses cases to guide its search through the rules and rules to guide its search through the cases. Finally, it generates arguments for and against

[2] named for the centaur in Greek mythology, known for giving good advice.

the success of its plans based on previous cases.

The contributions of this thesis include designing and implementing a system (CHIRON) that:

- generates plans from open-textured (underspecified) rules and cases:
- reasons about open-textured rules using a prototype, a set of adaptations, and a set of cases;
- combines hierarchical and case-based planners in a hybrid system; and
- generates plans with supporting arguments for and against the success of each plan, based on comparing and contrasting the current situation with previous cases.

In Chapter 2 of this thesis, I place open-textured planning in the context of past artificial intelligence work in both law and planning. In Chapter 3, I discuss the knowledge representation language used by CHIRON and its representation of rules and cases; in Chapter 4, I discuss CHIRON's design; in Chapter 5, I describe CHIRON's implementation in detail; in Chapter 6, I illustrate CHIRON's design and knowledge representation with a detailed example of the system's operation; in Chapter 7, I evaluate the system; and in Chapter 8, I summarize what I have done and discuss future work.

Chapter 2

Related Work

CHIRON is a legal planner, reasoning with open-textured rules and cases. As such, it builds on previous work in reasoning with open-textured rules and in planning. In the following sections, I discuss each of these areas in turn.

2.1 Reasoning with open-textured rules

Most of the work on reasoning with open-textured rules and cases has been in the area of artificial intelligence and law. In this section, I focus on recent work and work that is closely related to this thesis; for more details on earlier work, see the thorough and readable survey in [Gardner, 1987].

2.1.1 TAXMAN and EPS

TAXMAN was one of the earliest AI and law projects, a program that analyzed cases in the domain of corporate tax law [McCarty, 1977]. For this project, McCarty designed a representation language using an economical set of statutory predicates that was sufficiently expressive to state the facts of an input case in this domain in detail. For example, see how TAXMAN would represent "Iacocca owns 100 shares of Chrysler stock" (Figure 2.1).

TAXMAN took as input the description of a transaction in a specialized area of law, corporate reorganizations, and, upon request from the user, determined whether the transaction qualified for tax-free treatment under certain provisions of the Internal Revenue Code. First, it processed the input facts in order, using forward-chaining rules to expand them to a greater level of detail; then it used backward-chaining rules to determine whether the expanded facts satisfied the

```
(corporation Chrysler t1)
(issue Chrysler s1 t1)
(stock s1 t1)
(common s1 t1)
(piece-of p1 s1 t1)
(nshares p1 100 t1)
(own Iacocca p1 t1)
```

Figure 2.1: How TAXMAN would represent "Iacocca owns 100 shares of Chrysler stock."

relevant provisions of the Internal Revenue Code. The system moved flexibly back and forth between concrete descriptions and abstractions.

TAXMAN was limited by the fact that all of its abstractions were defined by if-then rules. This works fairly well for some simple inferences, such as determining that someone is a stockholder. For an example of such a rule, see Figure 2.2. Like other areas of law, however, corporate tax involves open-textured concepts. Important concepts that are not defined in the statute include "business purpose" and "step transaction." McCarty concluded that TAXMAN's rules were insufficient for representing these concepts. Standard rule-based expert systems would have the same problem in this domain.

In his next project, TAXMAN II, McCarty began investigating ways of modifying TAXMAN to handle open-textured concepts [McCarty, 1980, McCarty and Sridharan, 1982]. One approach might have been to add cases to the knowledge base. Examples of "business purpose" and "step transaction" are given in various cases (see, e.g., *Gregory v. Helvering,* 293 U.S. 465 (1935), and *Helvering v. Elkhorn Coal Co.*, 95 F. 2d 732 (4th Cir. 1938), *cert. denied,* 305 U.S. 605 (1938)). If TAXMAN had some facility for representing or reasoning with cases, it might be able to make use of these examples.

But McCarty did not add cases, at least not directly. Instead, he proposed to represent open-textured concepts using a *prototype*, a concrete description expressed in the lower-level representation language, and a sequence of *deformations*, or transformations of one concrete description into another. For example, a stockholding relationship could be represented by a pointer to the prototype of a pure equity interest, represented by a particular set of rights and obligations between the owner and the issuing corporation; plus an incremental set of transformations in the direction of a debt interest. The detail and precision of his low-level representation language make it effective for representing such small incremental changes.

This prototype-and-deformation approach predates case-based reasoning, to which it bears

```
(theorem abstract (o c s p)
 (stockholder ?o ?c)
    (goal (issue ?c ?s))
    (goal (stock ?s))
    (goal (piece-of ?p ?s))
    (goal (own ?o ?p)))
```

Figure 2.2: A backward-chaining rule from TAXMAN: an individual is a stockholder if some company has issued stock and he owns a piece of that stock.

a striking resemblance (see discussion below). McCarty's contributions are first, the idea that a set of related cases could be used to represent an open-textured concept; and second, the suggestion that the set of possible transformations of a case could be limited to those which preserve *conceptual coherence* in the corresponding concept [McCarty, 1980, McCarty, 1989a]. This term is not defined precisely, but it suggests an interesting direction for future work.

Unfortunately, TAXMAN II has not been fully implemented. As a result, McCarty has not addressed the significant algorithmic issues addressed by the case-based reasoning community: how to choose the prototypes, how to index them, how to search the space of prototypes, how to search the space of transformations, and the relationship of the prototypes to actual cases.

In the course of this project, McCarty developed a deontic logic for representing the concepts of permissions and obligations that occur in the legal domain [McCarty and Sridharan, 1982, McCarty, 1986, McCarty, 1983]. More recently, he has elaborated a knowledge representation language with a formal intuitionistic semantics that incorporates time, events, and actions, as well as permissions and obligations [McCarty, 1989b, McCarty, 1989a].

In joint work with Dean Schlobohm, an estate planning attorney, McCarty has sketched out a design for a legal planning system [Schlobohm and McCarty, 1989]. They argue that lawyers construct plans by retrieving *prototype plans* and transforming them to meet the clients' goals. They discuss how trusts and the Internal Revenue Code can be represented using McCarty's representation language. However, as with TAXMAN II, algorithmic issues such as how the prototypes are chosen, indexing, and search are ignored. And again, there is no explicit facility for representing or reasoning with legal cases.

2.1.2 Gardner

Gardner has also addressed the open-textured statutory predicate problem. Her (unnamed) system [Gardner, 1987], like TAXMAN, takes a sequence of events as input and determines whether that sequence satisfies certain legal requirements. Gardner's domain is contract law, but the problem is essentially the same.

Gardner's solution is quite different from McCarty's, however. Like McCarty, she starts with a rule-based system. Unlike McCarty, she uses cases. When processing an input case, Gardner's program fires its rules until they "run out," that is, until it reaches a rule that contains a term that is not expanded by some further rule. Then it looks at the commonsense meaning of the term and past cases whose facts match the current case. Where the commonsense meaning and past cases are all consistent, Gardner's program classifies the case accordingly, either in or outside of the term in question. Where no commonsense meaning is given and there are no past cases on point, or where the past cases disagree with each other, the input is considered a "hard case" and Gardner's program leaves its classification to the user.

For each past case, Gardner determined which open-textured statutory predicates were involved in the case and which of the facts set forth in the case report were relevant to the satisfaction or nonsatisfaction of each open-textured statutory predicate. An open-textured statutory predicate and the facts relevant to it form a "case pattern," and cases are represented as a list of case patterns.

In effect, each open-textured predicate corresponds to a set of cases. The case representations are simple patterns to be matched in the facts of the current situation; Gardner leaves for future work the problem of developing and using detailed case representations that would correspond closely to court decisions [Gardner, 1987, page 155]. The relationships between the cases are not spelled out. There are no levels of abstractions between the facts and the statutory predicates. Moreover, Gardner's program has no information about case transformations, such as those suggested by McCarty. As a result, it is hard for her system to compare and contrast past cases, and it has no mechanism for handling novel situations. It cannot modify an earlier case to fit a new situation or resolve conflict between cases. Cases are used to determine that a problem exists, but not as a basis for a solution.

Each of these case patterns involves a judgment call — the system designer's decision as to which of the facts set forth in the case report are relevant to the application of a given statutory predicate. When analyzing an input case with regard to a particular open-textured statutory predicate, Gardner's program retrieves the cases involving that statutory predicate whose facts

match the facts of the statutory predicate's case pattern. If too few facts are included, it will be too easy for a new case to match the prior case; if too many, the system will predict failure where it should have found a match. Moreover, there might be facts included in the official case report that at the time of representation do not appear to be relevant to any particular statutory predicate. Apparently, those would not be included at all.

2.1.3 HYPO, TAX-HYPO, CABARET, and BankXX

Edwina Rissland and her group at the University of Massachusetts have worked on a series of projects in the area of artificial intelligence and law, emphasizing particularly algorithmic and control issues.

HYPO illustrates a use of cases that is different from Gardner's [Ashley, 1991]. Like TAXMAN and Gardner's program, HYPO takes a sequence of facts as input. As in the earlier systems, the problem is to determine whether the facts satisfy certain legal requirements (in this case, whether they constitute a trade secrets violation). Unlike the earlier systems, however, HYPO does not output a yes or no answer. Instead, it generates arguments based on previous cases.

User input and cases are both represented in simplified form, using a standard "legal case-frame" to hold the important facts of the case. Legal case frames also include such information as the date of the decision, the court deciding the case, and the official citation. Each of the cases in HYPO's case base can be retrieved using a fixed set of indices (termed "dimensions"); dimensions are also used to compare cases. When the user inputs a legal situation, expressed in the legal case-frame language, HYPO uses this representation to calculate the applicability of dimensions to the situation, and then uses these values to index into relevant cases. The system then organizes the retrieved cases into a subset lattice based on the dimensions each retrieved case shares with the current situation and uses the lattice in constructing arguments for both plaintiff and defendant. Arguments are constructed by reciting the dimensions shared with favorable cases and the differences from unfavorable cases (nonshared dimensions or differences in the values of shared dimensions).

Suppose our situation is the following:

> Company D develops a line of home-style cookies. Company N places one of its employees as a "spy" (gets him hired) at Company D. He learns the secret method, then returns to Company N and sets up the technology for making home-style cookies, which Company N markets.

From these facts, HYPO distills the salient dimensional information: Company D develops secret method, Company N gains competitive advantage by having lower development costs (due to stealing method), Company N and Company D are competitors. Using this information, the system determines that the dimensions *secrets-voluntarily-disclosed* (zero) and *competitive-advantage-gained* (positive) apply to the current situation. It uses these dimensions to index to *Telex v. IBM*, a case where IBM was successfully sued for hiring a Telex employee for big money and using his development notes to produce a competing product for less money, and any other cases sharing one of these dimensions. For simplicity, suppose *Telex v. IBM* is the only case retrieved. Then we construct an argument for the plaintiff D by reciting the dimensions D's situation shares with *Telex v. IBM*, and for the defendant N by focusing on non-shared dimensions or differences in the values of shared dimensions (e.g., the number of people to whom information was disclosed) that make N's case stronger than IBM's.

HYPO's case representation does not include all of the facts in the official case report, but it does represent specific facts rather than fact patterns. When the system is reasoning at the level of dimensions, it has only thirteen dimensions to work with, and even the legal case-frames, since they have fixed pre-determined slots, cannot capture all of the facts in the judge's opinion (the official description of the case). HYPO's representation is simpler than TAXMAN's representation language would permit. On the other hand, it is more detailed and more concrete than the representation used in Gardner's program.

The advantage of HYPO's approach is that the use of dimensions makes it easy to compare and contrast cases and propose hypotheticals. The dimensions are carefully chosen, and their significance is firmly grounded in domain knowledge. The cases that are most similar on these dimensions are likely to be most useful in arguing the current case. For example, in the trade secrets area, we have dimensions for *secrets-voluntarily-disclosed, disclosures-subject-to-restriction,* and *competitive-advantage-gained* — the values of these tend to predict how strong a trade secrets case is for the plaintiff and defendant and can be used as a point of comparison with other similar cases. The disadvantage is that the legal case-frames, like any simplification, limit a system's ability to compare and contrast cases.

HYPO does not incorporate rules explicitly, but it is addressing the same open-textured statutory predicate problem as TAXMAN II and Gardner's program. In effect, the whole system can be seen as using cases to interpret a single open-textured statutory predicate, "trade secret violation." All the cases in the case-base correspond to that predicate. Unlike Gardner's program, HYPO also includes dimensions, a level of abstraction between the case facts and the predicate being interpreted. As a result, HYPO can compare and contrast cases in a way that

Gardner's program could not have done. In Gardner's program, the open-textured predicate corresponds to an unstructured set of cases; in HYPO, the set of cases is structured by the dimensions, in a way the reflects the context of each new problem case.[1]

In some respects, HYPO's dimensions operate like McCarty's transformations. They enable the system to modify cases, for example to create a hypothetical. In addition, HYPO incorporates knowledge about which modifications strengthen or weaken a case. There is no notion of prototype, however.

Gardner's program addressed the question of how to distinguish between easy and hard questions of interpretation; HYPO provides an algorithm for dealing with hard questions. Instead of looking for a yes or no answer in difficult cases, HYPO generates arguments by comparing and contrasting past examples of a particular statutory predicate.

In their next project, Rissland's group translated HYPO into a new domain. TAX-HYPO takes as input a fact situation expressed in legal case-frames and applies a HYPO-style dimensional analysis repeatedly to determine whether the situation satisfies each of the statutory requirements for a home office deduction under §280A of the Internal Revenue Code [Rissland and Skalak, 1989a].

TAX-HYPO demonstrated that HYPO's techniques could be used to compare and contrast cases and generate arguments with respect to individual statutory predicates in a legal domain other than trade secrets law. However, this experiment confirmed the obvious hypothesis that additional knowledge is necessary to reason in a statutory domain: TAX-HYPO was unable to combine its arguments concerning individual statutory predicates into an argument for or against the application of the entire statutory provision. It could only argue for or against the application of specific phrases within the provision.

As a result, Rissland's group has been exploring ways of combining case-based with other forms of reasoning. The CABARET project [Rissland and Skalak, 1989a, Skalak, 1989a, Skalak, 1989b, Rissland and Skalak, 1989c, Rissland and Skalak, 1989b], includes three modules: a case-based reasoner, a "co-reasoner," and a separate control module. The case-based reasoner is modelled on HYPO. The "co-reasoner" is currently a simple rule-based module including both forward and backward chaining. The control module uses heuristics to add, delete, or order tasks on a common agenda.

[1] Rissland's earliest work [Rissland (Michener), 1978] also described a structured space of examples (of concepts and results) in the domain of mathematics. Later work on "Constrained Example Generation" [Rissland and Soloway, 1980, Rissland, 1982] explored the use of "retrieval-plus-modification" to generate new examples and counter-examples. Constrained Example Generation was an early precursor to what is now known as "adaptive" case-based reasoning.

The main contributions of this project are in the area of controlling mixed case-based and rule-based systems and developing a theory of statutory argument [Skalak and Rissland, 1992]. Rissland's group considered various possible control structures:

- rule-based reasoning dominant: the rule-based module can call the case-based module, but not vice versa;
- case-based reasoning dominant;
- equal modules, each of which can call the other; and
- equal modules, with control knowledge isolated in a separate module.

They adopted the last of these, which allows them to simulate each of the other three and in addition, facilitates experimentation with a variety of control heuristics [Skalak, 1989a].

Heuristics are production rules that add, delete, or order tasks on the system agenda. The tasks on the agenda are calls to a procedure exported by one of the other modules, such as "backchain on subgoal *foo*." Heuristics they have considered include, among others:

- begin analyzing a problem by using the case-based reasoner to find a similar case;
- begin analyzing a problem by backward-chaining through the rules until they run out;
- if one module fails, switch to the other;
- once a conclusion is reached, double-check it by using the other form of reasoning; and
- if the rule is a near miss, use case-based reasoning to establish the missing rule antecedent.

TAXMAN and Gardner's program use the second of these heuristics: they are both systems where rule-based reasoning is dominant (TAXMAN of course does not use cases at all). GREBE (below) always tries both case-based and rule-based reasoning, generates all the solutions it can find, and uses heuristics afterwards to choose among them [Branting, 1990b]. CABARET was the first to examine a more complex form of control.

In connection with this project, Rissland and Skalak also defined an expanded model of argument. Besides the "straightforward argument" used in HYPO, based on comparing and contrasting the current situation with previous cases, they suggest a variety of other types such as "make-weight," "straw man," and "slippery slope" arguments. Only the straightforward argument has been implemented in CABARET, however [Skalak and Rissland, 1992].

CABARET is designed to be module-independent, so the control heuristics are generalized. They do not attempt to take advantage of the specific features of particular types of modules. In addition, the system is limited, like HYPO, by its simplified case representation. Finally, in natural language terms, it is a parser, not a generator: it takes facts and determines whether they satisfy a statutory predicate, rather than taking the predicate and generating a set of facts to satisfy it.

Most recently, Rissland's group has explored the use of other types of legal knowledge (e.g., legal story prototypes and legal theories) to represent legal knowledge for case-based argument. The specific task of browsing through and harvesting information important for case-based argument about open-textured predicates has been explored in the BANKXX project. The BANKXX system uses heuristics and best-first search (in addition to HYPO-style case-based techniques) to gather cases, legal theories, stereotypical legal stories, etc. on the issue of whether Chapter 13 personal bankruptcy plans are "proposed in good faith" [Rissland *et al.*, 1994].

2.1.4 Bench-Capon and Sergot

Bench-Capon and Sergot encountered the problem of representing and reasoning about open-textured predicates in a series of projects on the representation of legislation as logic programs [Bench-Capon and Sergot, 1988, Sergot *et al.*, 1986]. They suggest that a legal reasoning system should handle open-textured concepts by giving the user arguments for and against the application of the concept in borderline cases. This is similar to the approach taken by HYPO and CABARET. The primary difference is that Bench-Capon and Sergot advocate storing and using the general rule of a case, annotated with its facts, rather than reasoning directly from the facts. This suggested approach does not involve any use of prototypes, at least not explicitly, and does not seem to capture the idea of one case being stronger or weaker than another that is expressed by HYPO's dimensions.

2.1.5 PROLEXS

PROLEXS is a legal expert system in the domain of Dutch landlord-tenant law [Oskamp *et al.*, 1989]. Like CABARET, PROLEXS combines various types of legal knowledge in a blackboard architecture. As in CABARET, there are distinct reasoners for different types of legal knowledge, with control knowledge isolated in a separate module. PROLEXS's case representation is quite unlike the one used by HYPO and CABARET, however, as shown in Figure 2.3. The case shown

Facets	Points
(landlord residence distance-from-tenant > 3000)	60
(landlord residence distance-from-tenant > 300)	10
(landlord residence distance-from-tenant < 301)	-10
(contract length longer-than 00/00/04)	60
(contract length longer-than 00/00/02)	40
(contract length longer-than 00/00/01)	20
(contract length smaller-than 00/00/01)	-20
Threshold: 105	
Abstract (contract status fixed-period YES)	
to (house usage short-termed-by-nature NO)	

Figure 2.3: Sample case representation from PROLEXS.

is retrieved to interpret the phrase "short term," which sounds like the kind of open-textured predicate that might be found in an Anglo-American statute, but instead of concrete facts such as "John Smith rented an apartment to Joseph Jones for three months while he went to Paris, 400 kilometers away from Amsterdam," there is a general rule that if the distance of the landlord from the apartment and the length of the contract together meet a certain threshold, the contract will not be considered short-term. PROLEXS appears to choose relevant cases if applicable, rather than comparing and contrasting them with each other, as HYPO and CABARET do.

Unlike the other projects described in this section, PROLEXS was designed as a practical system. It was intended to be used by a law student advising clients in a legal-aid clinic, and a prototype of the system has been tested in that context. Part of the work is shared by the law student operating the system, who makes the initial determination of which issues should be considered by the system, has the ability to control the system's reasoning by adding or deleting facts from the blackboard, and interprets the system's output to the client.

2.1.6 GREBE

GREBE [Branting, 1988, Branting, 1989a, Branting, 1989b, Branting, 1989c, Branting, 1990b], like HYPO, generates arguments for the satisfaction or nonsatisfaction of a given legal statutory predicate. Unlike HYPO, GREBE has a detailed case representation and uses rules as well as cases. In addition, its arguments are based on the reasoning used in previous cases, rather than a comparison of their facts.

GREBE's approach is similar to Gardner's, but it significantly extends Gardner's work. GREBE uses a more complex case representation: a semantic net in which individual facts

correspond to relation/unit/value triples and the facts of an entire case correspond to a labeled graph. Labeled subgraphs of this graph correspond to open-textured statutory predicates involved in the case: the subgraphs correspond to the facts used by the court to explain its result with respect to that statutory predicate (the *criterial facts* for that statutory predicate). The subgraph-statutory predicate pairs are similar to the rules Gardner's program uses to represent cases, but the case representation used by Gardner's program is a bundle of separate rules; here, all the facts of a particular case are joined into a larger graph. Moreover, GREBE can represent facts that are mentioned in the case report, but not cited as relevant to any open-textured statutory predicates, as part of the case graph not covered by any of the labeled subgraphs.

GREBE takes as input a case and a desired statutory predicate in the domain of workers' compensation and outputs an argument supporting the application of that statutory predicate in the given case. First, it tries to find the desired conclusion in the input facts. If that fails, it backward-chains through its rules like TAXMAN or Gardner's program, attempting to show that the desired conclusion is the consequent of some rule, all the antecedents of which are themselves satisfied. Like Gardner's program, GREBE turns to cases if its rules run out. Unlike Gardner's program, GREBE can handle partial matches. Instead of looking for cases which exactly match the current situation, it looks for the best match (determined by mapping the criterial facts of previous cases involving the desired statutory predicate onto the current situation, and choosing the case with the fewest unmatched facts). All criterial facts are treated equally; they are not weighted according to their importance [Branting, 1994]. If both positive and negative cases exceed a certain (arbitrary) threshold, GREBE constructs arguments in both directions. Finally, if the previous case is a partial match, GREBE calls itself recursively to infer any missing facts.[2]

GREBE uses the richest knowledge representation of all the systems discussed (other than TAXMAN II, which was never fully implemented). In addition, it can construct new arguments, by piecing together elements of different past cases and domain rules. Moreover, this was the first system to attempt to represent and use the reasoning from past cases. By including this information, GREBE captures some of the internal structure of its cases: the relationship between facts and result, as suggested by the court. Contrast HYPO and the other systems developed by Edwina Rissland and her group, which, as discussed above, use frames to represent the facts of cases, but do not attempt to represent the courts' reasoning.

[2]For clarity, the flow of control has been simplified here: in fact, GREBE always tries to solve its goals using *both* rules and cases, generates all the solutions it can find, and uses heuristics afterwards to choose among them.

GREBE's case representation enables it to abstract from the facts of a case to intermediate-level predicates used by the court in reasoning and (for example) to construct an argument from pieces derived from different prior cases. GREBE can retrieve a group of cases illustrating a given predicate. Moreover, because GREBE's case representation includes information at varying levels of abstraction, like HYPO's representation, it can support the tasks of comparing and contrasting the cases retrieved. It would be possible for GREBE to represent facts that are mentioned in the case report, but not cited as relevant to any open-textured statutory predicates, as part of the case graph not covered by any of the labeled subgraphs.

On the other hand, GREBE's representation is limited by the fact that its abstractions are tied to the reasoning used in previous cases. The case representation does not take into account abstractions other than those mentioned in the court opinion in which a particular fact occurs. Moreover, GREBE's representation incorporates the same judgment calls as the representation used in Gardner's program; the determination of which facts are responsible for the application of a particular predicate is built into the representation. Although these decisions are derived from court opinions rather than being solely the work of the system designer, they still make it impossible for the system to analyze the cases independently, as a human lawyer would. Although noncriterial facts could be represented, in fact, GREBE only includes facts that are used by the court to explain its result with respect to some predicate [Branting, 1994]. Finally, like Gardner's program, GREBE has no way of sorting previous cases temporally, so there is no way of determining trends.

In addition, using a richer knowledge representation has disadvantages as well as advantages: first, representing cases is time-consuming and difficult. A feature vector representation (which is essentially what HYPO uses) makes it easier to translate from case report to representation. Second, a richer representation language makes it harder to enforce consistency in the case representations. Third, the use of the court's reasoning involves judgment calls. The exact scope of the criterial facts and the precise logical connection between various reasoning steps are seldom completely explicit in legal opinions. Perhaps most critically, the process of matching cases becomes equivalent to graph isomorphism. Since the edges of GREBE's graphs are labeled, the problem is simpler than full subgraph isomorphism (which is NP-complete [Garey and Johnson, 1979]), but still more complex than simple vector comparisons.

2.1.7 Projects outside of artificial intelligence and law

Protos, a medical diagnosis system, also addressed the problem of classifying new instances of underdefined concepts [Bareiss, 1988]. Influenced by the psychological literature on category formation, Protos represented concepts extensionally using a set of cases.

Protos's representation is similar to the one suggested by McCarty for legal concepts. While McCarty suggested a set generated by a single prototype and a set of deformations, Protos used a set of prototypes, each of which was an actual case. When given a new case, it retrieved a similar case and suggested its classification to a human teacher. If the classification was correct, and the case had new features, Protos stored it as a separate exemplar. If the classification was mistaken, Protos recorded its mistake and tried again. It does not compare and contrast the current situation with more than one previous case; it does not express the relationship among the previous cases; and it does not adapt the result in the previous case (the classification) to fit the current situation.

Anapron is a recent project that addresses the problem of rules with exceptions in the domain of name pronunciation [Golding and Rosenbloom, 1991]. It applies rules to find a pronunciation for a given name and uses that pronunciation, unless the current case is judged to be "compellingly similar" to a previous exceptional case. Legal rules do sometimes have unspoken exceptions and qualifications; but the more general problem of vagueness and open texture that is found in legal rules is not found in Anapron's domain and is not addressed by this project.

In JULIA, cases are used as starting points for solving design problems in an "open world" [Hinrichs and Kolodner, 1991, Hinrichs, 1991]. JULIA is an interactive system; the system and the user cooperate to generate menus, given constraints such as the facts that one of the guests is a vegetarian and another will only eat meat and potatoes. The menus generated must fit within open-textured categories such as "vegetarian meal" or "Italian cuisine." Like GREBE, JULIA selects the single "best" case at each point in its decision-making; also like GREBE, it can combine information from several previous cases into a single solution to the current problem. Unlike the other systems discussed in this section (and unlike CHIRON), JULIA solves design problems. As a result, the system makes use of constraint-based reasoning and addresses the problem of dynamic constraints: design specifications that change as the problem is being solved.

Like the legal-reasoning systems, JULIA and Protos use cases to reason about open-textured concepts. Anapron uses cases to reason about rules with exceptions. Unlike the legal-reasoning

systems, none of these systems is implemented in an adversarial domain, so none of them needs the notion of a "safe," conservative, solution; and none of them justifies its conclusions by comparing and contrasting the current solution (diagnosis, design, or pronunciation) with previous ones.

2.2 Planning

There is a long tradition of work on planning in artificial intelligence. For purposes of this thesis, a *planner* is a program that takes a goal or goals as input, along with an initial state description, and outputs a sequence of actions, or *plan*. The plan is correct if the program user (human or robot) can execute it, starting in the initial state and after execution, the goals have been achieved. Intermediate plans may be abstract, but the final plan should contain *primitive actions* (i.e., actions that can be executed by the program user.)

Two lines of work in planning are particularly relevant to CHIRON: hierarchical and nonlinear planning and case-based planning. I focus on these in the remainder of this section.

2.2.1 Hierarchical and nonlinear planning

In linear planning, an intermediate plan consists of a sequence of primitive actions, and plans are constructed by adding steps to this prefix [Fikes and Nilsson, 1971]. Thus, for example, suppose you normally have a cup of coffee for breakfast. A linear planner with the goal, "have breakfast," would start with a primitive action, perhaps, "get filter," then select another one that can be performed after the first one, perhaps "put filter in basket," until it found a sequence that resulted in a cup of fresh coffee. This approach is receptive to formal treatment [Lifschitz, 1987]. However, it fails to take advantage of the structure that can be provided by abstractions.

In hierarchical planning, by contrast, intermediate plans are complete but abstract. Instead of simply adding primitive actions to an intermediate plan, the planner repeatedly refines the plan, gradually substituting more and more specific plans until it has a sequence of primitive actions. (See, e.g., [Sacerdoti, 1974]). Working on the above example, a hierarchical planner might start with the plan, "Make breakfast," then substitute "Make coffee," then "Get filter, put filter in basket, get coffee, put coffee in filter, ..." and so forth. The planner can delay adding details until sufficient information is available. In nonlinear planning, the planner delays ordering the tasks, as well as making them more specific [Tate, 1977, Tate, 1976]. For example, if you are planning to have cereal as well as coffee, you can prepare the cereal first

and then make the coffee, or vice versa; but if you plan at a more specific level, you will be able to interleave your steps, and, for example, pour the milk on the cereal and in your coffee before returning it to the refrigerator, instead of getting it out twice. It is not necessary (and can even be disadvantageous) to decide the ordering of these tasks at the top level.

Neither of these approaches is sufficient for our task. If we start with primitive actions, as in linear planning, we have no way of testing whether they satisfy the open-textured rules; and if we start with abstractions, we will be unable to refine the plan completely. We can use schemas based on the tax rules to refine the plan up to a point, but the rules run out. Without cases, we have no way of making the connection between the abstractions based on open-textured provisions and primitive actions.

2.2.2 Case-based reasoning and case-based planning

Recently, several projects have been developed in the area of case-based planning — using previous plans as the basis for constructing new ones (See, e.g., [McCartney and Wurst, 1991, Hinrichs, 1991, Hammond, 1986, Alterman, 1986]). Case-based planning is part of a more general research effort known as case-based reasoning (CBR). CBR has attracted a good deal of attention in recent years; it has been incorporated into a number of systems that perform a variety of tasks in a variety of domains. It is characterized by its use of memory in problem-solving. Where possible, CBR systems rely upon "memories" of past experiences to solve problems, rather than deducing the answer from first principles. The "memories" stored in the system's knowledge base are referred to as *cases*.[3]

By making use of past cases, CBR systems can gain in both efficiency and reasoning power. Even in a simple blocks-world domain, if you've solved the same problem before, it's easier to look up the solution than to figure it out again; and in a more complex domain, where your knowledge is incomplete, you can create plans based on experience when it would be impossible to deduce them logically. For example, a child can plan to turn on the light by flipping the light switch without any understanding of the electrical wiring in the building or the physics of electricity, simply because it knows that this plan has usually worked in the past. By extension, when the child wants to turn on another device, it may adapt its previous plan and look for an on-off switch, again without any understanding of how the device works.

[3]The use of the term *case* in both CBR and law has caused some confusion. It is important to distinguish between the two. Law cases, as described above, consist of a statement of facts, in which all the relevant facts are included (by definition); a result; and a suggested chain of reasoning connecting the facts and the result. CBR cases include facts and result (diagnosis, design, plan, or some other output, depending on the type of system). Generally speaking, there is no reasoning connecting the two, and there is no guarantee that all the relevant facts are present.

CBR is an attractive candidate for use in a legal reasoning system. In law, there is no strong theory connecting facts to open-textured statutory predicates. Traditional deductive approaches are not effective. CBR in general, and case-based planning in particular, are especially well-suited to domains where the reasoner has a weak causal theory.

Certain issues must be addressed by any CBR system:

1. What do you know? What cases are contained in memory, how are they represented, and what kind of information do they include?
2. How can you find it? How are cases indexed and retrieved?
3. How can you make it useful? How are cases adapted and repaired for use in the current situation?
4. How can you evaluate the output of the system?

For example, consider the case-based planner CHEF, which constructs plans, or recipes, in the cooking domain [Hammond, 1986]. It takes as input the user's goals, that is, the desired tastes, textures, and types of dishes, and uses its case base of previous cooking experiences to construct a recipe that satisfies those goals.

CHEF started with an initial library of ten recipes and created twenty-one others in response to user requests. Each plan includes a list of ingredients, a list of steps, and a type (e.g., stir-fry or souffle.) A sample recipe is given in Figure 2.4.

The issue of what to include in a case is not addressed explicitly in CHEF. CHEF's recipes, like the one given in Figure 2.4, include the kind of information that would be found in a cookbook: the ingredients and the steps to be performed. They do not include, for example, the name of the store where the ingredients were bought, their price, the utensils used, or the height above sea-level of the kitchen.

In theory, a case should include all the information that will be useful in constructing future plans. Arguably, therefore, the case descriptions in CHEF should have been much more complete. On the other hand, it's inefficient to include information that will never be used. In cooking, as in most domains, it is impossible to be certain which facts will be useful; the system designer can only make an informed guess at the time each case is represented.

In this respect, law differs from most other domains. In law, previous plans are described in the statement of facts in the official case report. Any of these facts can be used in future arguments. Thus, the system designer can be sure that these facts should be included in the case representation.

```
(def:rec broccoli-with-tofu
  (ingredients
     ingr1 (tofu lb .5)
     ingr2 (soy-sauce tablespoon 2)
     ingr3 (rice-wine spoon 1)
     ingr4 (corn-starch tablespoon .5)
     ingr5 (sugar spoon 1)
     ingr6 (broccoli lb 1)
     ingr7 (r-pepper piece 6))
  (actions
     act1 (chop object (ingr1) size (chunk))
     act2 (marinade object (result act1)
                    in (& (ingr2) (ingr3) (ingr4) (ingr5))
                    time (20))
     act3 (chop object (ingr6) size (chunk))
     act4 (stir-fry object (& (result act2) (ingr7)) time (1))
     act5 (add object (result act3) to (result act4))
     act6 (stir-fry object (result act5) time (2)))
  (style style-stir-fry))
```

Figure 2.4: CHEF's recipe for broccoli with tofu.

Two recent planners have added case-based reasoning to nonlinear planners and taken the reasoning process by which the plan was generated (i.e., the final successful search path through the plan space) as their cases [Veloso, 1992, Kambhampati and Hendler, 1989]. When solving a new problem, they adapt the method used to solve a previous problem, rather than the previous problem's solution. This approach, known as *derivational analogy*, is similar to the way in which a student will solve a calculus problem, or construct an inductive proof, by adapting a known technique to fit a new problem.

The reasoning process by which the recipes were created is not included in CHEF's cases. Intuitively, real planners in this domain (i.e., cooks) are more likely to start with a recipe and adapt it, than to build one from first principles. A legal planner could make use of the reasoning process that a previous case used as a path from first principles to a concrete plan, but given the fact that the rules "run out" in this domain, the path would be incomplete, and would need to be supplemented, either by the court's reasoning from the rules to the facts of that particular case, or by more general domain knowledge.

Suppose you have decided what to include in your cases. The next question is, how do you

find what you need? Or, in planning, how can you find a plan that can be used (possibly with modifications) to help you achieve your current goals?

The simplest solution is a linear search through the case base. But this is too inefficient. In systems such as CHEF, with a case base of ten to thirty cases, an exhaustive search would be possible. But in a system that is even slightly larger, this becomes impractical. A cook is unlikely to read through every page of a cookbook looking for a particular recipe. And in a case base the size of a law library, it would be impossible. Hundreds of cases are decided every day by the state and federal courts in this country.

To cut down on search time, the case base must be indexed in some way. Lawyers have faced this issue in practice. In response, they have developed more and more sophisticated ways of indexing case law. In addition to subject indexes and treatises on particular fields of law, for example, there are services that list every case that cites a given case, and every case that cites those cases, and so on. Now there are computer databases that contain the text of reported decisions, handed down as recently as the past couple of days. Lawyers search the databases by designing regular expressions, making it possible to find, for example, all Massachusetts decisions between 1963 and 1965 involving basketball players from Indiana, or any other combination of facts from almost any jurisdiction.

CBR has addressed the indexing problem in theory, as well as in practice. The main questions that must be answered by anyone indexing a case base are:

- What features should a case be indexed under?
- Should they be features found in the case description, or abstractions from those features, or both?
- Should they be ordered? If so, how?
- Should the feature set be static or dynamic?

The features used for indexing should be the ones that will allow you to retrieve a case in the future when you need it. In planning, you start with a goal. So one reasonable solution is to index plans under the goals that they achieve. CHEF, for example, would index the above recipe under the goals, "make stir-fry" and "include broccoli." In law, this might correspond to indexing plans under "sell house" or "satisfy the statutory predicate *principal residence*."

One planner's side effects are another planner's goals, however. Any action performed or any state achieved during the execution of a plan might be considered a goal. For example,

CHEF considers including a certain ingredient or making a certain type of dish to be goals. Performing a certain type of action is not considered a goal, so recipes are not indexed under the actions they include. But suppose that you have entered a cooking contest that requires you to demonstrate certain skills. You might be looking for a recipe that involves, say, separating eggs. If you asked CHEF for such a recipe, it would be unable to retrieve one, even though it has one (a souffle recipe) in its case base. Or suppose you're looking for a dinner that you can cook in less than an hour. Again, you could not retrieve one, since recipes are not indexed under the time they take.

Thus, the choice of feature set involves a tradeoff between efficiency and reasoning power. The ideal feature set would limit search through the case base, but not so much that the system fails to retrieve cases it could use. The same tradeoff must be faced by the designer of a legal reasoning system.

The question of whether or not to include abstractions has caused a certain amount of controversy in CBR [Hammond, 1989, Waltz, 1989]. CHEF indexes plans under both the ingredients included and abstractions of those ingredients, in order to allow partial matches. Thus, the above recipe is indexed under both "include broccoli" and "include vegetable," and it can be retrieved in response to a request for "a stir-fry recipe with green beans" as well as "a stir-fry recipe with broccoli," since green beans and broccoli are both vegetables.

In law, it seems clear that you want to include both surface features and abstractions. This corresponds to the difference between indexing under the actual words of the court's opinion, as the current computer databases do, and under abstractions, as HYPO does [Ashley, 1991]. Ideally, a system would use both.

A more difficult problem, not addressed in CHEF, is how to choose the abstractions. Note that the broccoli and tofu recipe is indexed under "stir-fry with tofu and vegetable," not "stir-fry with tofu and something green." Presumably the intuition is that type of food is relevant to a cooking plan, and color is not (except on Saint Patrick's Day). But this raises a difficult theoretical issue, the idea of relevance.

CHEF orders its indices. This simplifies search — plans are indexed in a discrimination net — but the ordering is fairly arbitrary. Type of dish is considered to be more important than ingredients, for example. Thus, the above recipe can only be retrieved if you know you are looking for a stir-fry dish. If you have broccoli and tofu in the refrigerator and want to know what to make for dinner, you will not be able to access this recipe. In law, as in cooking, there is no obvious ordering. As with the choice of features, there is a tradeoff between the efficiency provided by ordering and the corresponding loss of reasoning power.

Finally, CHEF uses a static set of indices. Static indexing is easier to implement, and as a result, has been used in most CBR systems. It makes the initial choice of features, abstractions, and ordering more important, however. Dynamic indexing gives the system the flexibility to solve problems involving a demand for information that is not originally accessible. In CHEF, for example, information about ingredients is included in the case base. Thus, if CHEF had dynamic indexing, and it was presented with a request for a recipe using broccoli and tofu, it could identify the request as a request for information that might be in its case base. Finding that the information is not accessible at the top level, it could modify its indexing and retrieve an appropriate case. With static indexing, this is not possible. There is a tradeoff between ease of implementation and flexibility; again, this is a problem that legal reasoning systems share with systems in other domains.

Once you have decided what to include in the case base and how to find it, the next question that must be answered by anyone designing a CBR system is, how can I use this information? Or, in planning, how can I use the retrieved plan (with modifications, if necessary) to achieve my current goal? At one extreme, if no adaptation is involved at all, CBR reduces to table lookup; the system can only solve problems that are exactly the same as those it has seen before. At the other extreme, it has been suggested that in some domains unlimited adaptation might be allowed [Riesbeck and Schank, 1989, page 42]. In other words, the system could generate a solution from any given case for any problem. But if you can generate a solution from any given case, you can generate one from scratch, so a system with unlimited adaptations would share the computational problems of hierarchical planning.

In CHEF, adaptation is divided into two phases, modification and repair. The modifier adapts the retrieved plan before execution. The changes it makes are relatively simple, such as the substitution of an ingredient of the same type. Thus, CHEF could generate a plan for a raspberry souffle from its strawberry souffle recipe, simply by substituting raspberries for strawberries.

The repairer can make more significant changes. It has a set of strategies for repairing each of twenty kinds of plan failure. Each of the plan failures the repairer knows about is due to interactions between the steps and states of the plan: for example, the failure of a plan to terminate because of a blocked precondition on a particular step. As a result, each of the possible repairs attempts to break a link in the causal chain that leads to the failure, by adding steps, substituting steps with different side effects, or re-ordering steps.

Translating these adaptations into the legal domain, substituting a new ingredient for an old one might correspond to taking the facts of an old tax case and substituting in the name and

address of your client. Adding a new ingredient might correspond to the following situation: suppose your client wants to sell his house, buy another house, and make a cash gift to some charity, all in the same year. Using CHEF's approach, you would retrieve the case that meets as many as possible of your client's goals, which might be a case that involves selling and buying houses, but says nothing about charitable donations. Then instead of, say, retrieving another case and combining the two, CHEF's Modifier would add steps corresponding to a standard charitable donation to the house-sale plan. Finally, the causal relationship between plan steps is often significant in law as well as cooking. For example, once you sell a rental property, you can no longer take depreciation deductions on that property; once you take the §121 exclusion (for house sales after age 55), you can never take it again; and so forth.

Kambhampati and Hendler have suggested a general criterion for adaptations in the context of their work on PRIAR. PRIAR is a domain-independent planner whose cases are plans generated by a nonlinear planner based on Nonlin and annotated during plan generation with a description of their internal causal structure. When adapting one of its cases to fit a new situation, PRIAR uses these annotations to localize adaptations and minimize the changes to the structure of the old plan. In general, they suggest that efficient adaptation strategies (which they call "refitting") should be *conservative*: they should change the old plan only as much as necessary to fit it to the new situation [Kambhampati and Hendler, 1989].

Hanks and Weld describe a formal adaptation algorithm, implemented in their system SPA. Their algorithm is sound, complete, and systematic: its plans are guaranteed to work, it always finds a plan if there is one, and it never tries the same adaptation twice [Hanks and Weld, 1992].

In both PRIAR and SPA, cases provide an increase in efficiency, but not in reasoning power. Hanks and Weld define the difference between traditional planning and case-based planning as follows:

> We view the general planning problem as a search through a directed graph of incomplete plans ... the graph's root represents the null plan and its leaves represent complete plans. Generative planning algorithms start at the root of the graph and search for a leaf node. ... The retrieval phase of a case-based planner, on the other hand, returns an arbitrary node in the graph, and the adapter begins searching from that point.

In other words, it's possible either to build a plan from scratch or to start with an old plan and modify it; the only reason for choosing the latter is that it may be more efficient [Hanks and Weld, 1992, page 97]. Similarly, Kambhampati and Hendler note that one of their goals

was to ensure that PRIAR's plans were as correct as the plans that would have been generated by the nonlinear planner alone (i.e., that any errors in a plan are due to the inadequacy of the system's domain theory and would have occurred regardless of the method by which the plan was generated) [page 948][Kambhampati and Hendler, 1989].

In domains with a weak causal theory, cases provide an increase in reasoning power, as well as efficiency. For example, a child can make toast without any knowledge of biochemistry or physics, simply by following a plan that has worked in the past: put the toast in the toaster and press the button. If required to explain the causal links between these actions and brown bread, the child would fail. In other words, the child's mental "planning graph" does not have a path from the root (eat breakfast) to the desired leaf (put toast in toaster, press button), but does have a path from another, similar leaf (toast was put in toaster and button pressed yesterday) to the new plan.

In such domains, planning is also more uncertain. Without a strong causal theory, the planner cannot be sure what effect an adaptation will have. For example, CHEF's recipe for beef and green beans allows the meat and vegetables to be stir-fried simultaneously. A simple substitution of broccoli into this recipe yields a recipe for beef and broccoli stir-fry. However, when the recipe is executed in the simulator, CHEF discovers that the broccoli is soggy. Even seemingly minor adaptations may cause a plan to fail.

In a domain where adapted plans are not guaranteed to work, and it is impossible to prove that the planner's output is correct, how can we evaluate a planner? One alternative is to try its plans out and see how they work. And this is what CHEF does. After it generates a recipe, it executes it (in a simulator), repairs it if necessary, and stores the final recipe in its case base.

For example, after CHEF generates its beef-and-broccoli recipe, it evaluates it by cooking it (in the simulator) and finds that the broccoli is soggy. It then deduces (correctly) that this error is caused by cooking the broccoli with the beef, repairs the recipe by separating the cook-broccoli and cook-beef steps, and stores the repaired recipe under "soggy broccoli," the problem it is designed to avoid, as well as under "stir-fry with beef and broccoli."

In law, the planner can neither prove a plan correct nor try it out. The equivalent of executing a plan would be having a court pass judgment on it. The courts do not have the time to rule on every plan, and even if they did, most planners would be unwilling to accept the cost and uncertainty of a trial. Legal reasoning systems have to develop new approaches to validation.

2.3 Summary

CHIRON builds on work in both planning and reasoning with open-textured concepts. The problem of reasoning about open-textured concepts has been addressed by previous systems in the context of analysis, categorization, and design. These projects suggest various approaches, including using prototypes [Bareiss, 1988, McCarty, 1980, McCarty, 1989a]; a set of simplified "case patterns" [Gardner, 1987]; a set of cases with concrete facts, related by dimensions [Ashley, 1991]; and the single "best match" from a set of cases with a rich, detailed semantic-net representation [Branting, 1990a].

In planning, the two lines of work that are most closely related to CHIRON are hierarchical, nonlinear planning and case-based planning. Hierarchical planning provides us with a way of reasoning with the legal rules on which our plans are based, but is insufficient in this domain, since schemas based on rules "run out" and must be supplemented. Case-based planning provides us with a way of reasoning with legal cases and clarifies many of the issues involved in case representation, indexing, and adaptation, but does not express the way in which cases in this domain interpret rules.

CHIRON combines hierarchical and case-based planners, in order to make use of legal rules and cases and the relationship between them. Corresponding to each rule is a prototype, which represents a safe, conservative plan; a set of dimensions along which the prototype can be deformed, or adapted; and a set of cases, which are related to the prototype by the deformations. And along with each plan, CHIRON generates arguments for and against its success, using the same type of "straightforward" argument implemented in HYPO and CABARET. CHIRON's detailed case representation is described in Chapter 3, and its design is described in Chapters 4 and 5.

Chapter 3

Knowledge Representation

3.1 Introduction

What knowledge do we need to represent? In constructing plans under United States tax law, lawyers have two main types of information to work with: rules (including statutes and the associated regulations) and cases. First, then, we want to represent both of these sources of knowledge. We also want to represent safe harbor plans, or prototypes, that satisfy the rules. We want to represent the relationship between the rules, prototypes, and cases. And finally, we need to represent the input description of the taxpayer's goals and current situation.

In practice, lawyers often take advantage of another source of information, plans based on past experience of similar transactions. Often, however, no such plan is available, perhaps because the lawyer has never performed this particular type of transaction before, perhaps because the law has changed so recently that no plans are available. And even where there is such a plan, if it is challenged in court, it must be justified in terms of the statutes and case law. In CHIRON, therefore, we have not included that kind of experiential knowledge, or "war stories." For two very different projects using that kind of knowledge, see [Slator and Riesbeck, 1992, Schlobohm and McCarty, 1989].

CHIRON's statutes, cases, and input have all been hand-coded into an internal representation.[1] We will discuss the representations we use for rules, prototypes, cases, and the relationships between them in the following sections.

[1]The issues involved in processing the natural language of legal texts are outside the scope of this thesis. For interesting work in this area, see [Cohen, 1987].

3.2 Statutory rules

One of the key problems in legal reasoning, as discussed in Chapter 1, is the need to represent and reason with open-textured rules. In particular, since CHIRON is a tax planner, we need to represent part of the tax law. The rules governing the United States income tax are contained in Subtitle A of the United States Internal Revenue Code. Subtitle A in turn is broken down into more than fifteen hundred sections, each of which can be anywhere from a paragraph to several pages long.

These rules fall into a few broad categories. First, there are rules concerning the items that should be included in the taxpayer's income, for purposes of the tax. Gifts received by the taxpayer, for example, are not counted as income, but salary and interest on bank accounts generally are. Second, there are rules providing for deductions – items that may be subtracted from the taxpayer's income before the tax is calculated. There are rules for calculating the amount of income or loss generated by selling a piece of property. There are rules for computing the amount of tax, once the net income has been calculated. And there are rules specifying credits, amounts that can be subtracted from the tax itself before it is paid.

These statutory rules form the basis for the hierarchical planner's decomposition rules. Following the same basic structure, the hierarchical planner starts with a top-level system goal of reducing the taxpayer's taxes. This can be achieved by finding credits or reducing taxable income. Reducing taxable income, in turn, can be achieved by finding exclusions, showing that income should be attributed to someone else, finding deductions, or deferring income. The plan of finding an exclusion can be achieved by establishing that you have satisfied Section 121 or Section 119, and so forth. We refer to all of the plans for finding exclusions, attributing the income to someone else, finding deductions, and deferring income collectively as "strategies." A portion of this hierarchy is given in Figure 3.1.

The hierarchical planner's decomposition rules are represented in the same way as Nonlin's schemas:

```
(defstruct  schema
   (id
    name
    todo
    steps
    size
    conditions
```

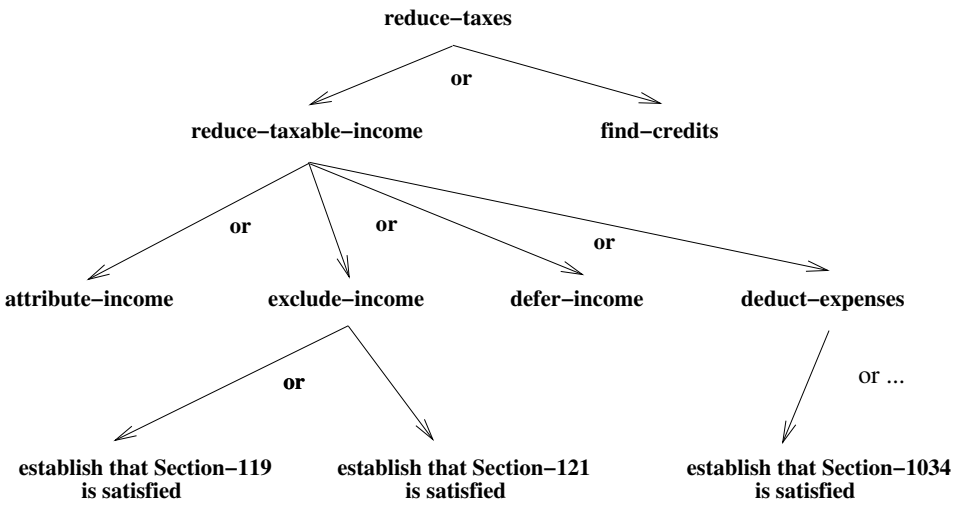

Figure 3.1: Part of the plan hierarchy.

```
effects
 vars)
```

except that our slot "steps" is called "expansion" in Nonlin.

Examples of CHIRON's rule schemas are given in Figure 3.2. The first of the schemas states that one way to reduce a taxpayer's taxable income for the tax year starting on ?start-year and ending on ?end-year is for the taxpayer to deduct expenses during that time period. This corresponds roughly to Section 63 of the Internal Revenue Code, which provides, "[T]he term 'taxable income' means gross income minus the deductions allowed by this chapter ...."

The second schema states that one way for the taxpayer to deduct expenses during a given year is to establish that he or she has satisfied Section 1034 of the Internal Revenue Code during that year. "Establish" is like "achieve," but with a specific legal meaning: perform actions such that if the case were heard in court, the court would hold that the rule was satisfied. A sentence expressed in open-textured legal language is true of the world ("satisfied") if a court holds that it is true.

The strategies, which correspond to the leaf nodes in the hierarchy shown above, represent the open-textured portions of the rules. Section 1034 is about four pages long, but we have represented it only by the token "Section-1034." We could have broken Section 1034 down into subtasks corresponding to various phrases in the section, such as "show that a given house was your principal residence," "show that you bought a new residence," "show that you used the new residence as your principal residence," and so forth, and indexed the cases under those

```
(todo ?tsk (occurs (reduce-taxable-income ?taxpayer)
                                    ?start-year ?end-year)
      (plan
       :id 14
       :steps
((step1 :action (occurs (deduct-expenses ?taxpayer)
      ?start-year ?end-year)))))

(todo ?tsk (occurs (deduct-expenses ?taxpayer)
                                    ?start-year ?end-year)
      (plan
       :id 141
       :steps
       ((step1 :open-textured
       (establish (satisfied Section-1034 ?taxpayer)
  ?start-year ?end-year)))
       :citation 1034))
```

Figure 3.2: Examples of CHIRON's rule schemas.

subtasks. But the taxpayer needs to construct a plan for the entire section. Each of the cases illustrates the entire section. The court's holding applies to the entire section: a taxpayer cannot receive a partial deduction for satisfying part of the section. So we consider the whole of Section 1034 (and similar provisions) as a single "open-textured rule," and represent and reason about it as a unit. More accurately, we have represented each open-textured rule by a prototype, a set of deformations, or ways in which the prototype can be adapted, and a set of past cases where the taxpayer attempted (successfully or unsuccessfully) to satisfy that rule. All of these knowledge sources are indexed under the rule-token and retrieved when the system reasons about the corresponding rule. Another way to look at this is that each rule corresponds to a region partially defined by a set of cases, with the prototype at the origin. The boundary of the region is extended by positive cases and limited by negative ones.

3.3 Cases

The designer of any case-based reasoning system must face the question of what to include in the case representation. The answer to this question is constrained, at least at the top level, by the process in which the cases will be used: here, reasoning about open-textured rules.

In order to reason about the relationship between open-textured rules and cases, a system must be able to:

- distinguish between two cases;
- show similarities between two or more cases;
- indicate which cases are central and which are peripheral; and
- determine trends in a series of cases illustrating a given rule.

A trend in the cases may make it possible to predict the result in the current situation. Comparisons with previous cases may help to predict the result in the current situation, suggest a plan, or highlight problems. The more similar the current plan is to previous cases that satisfy (or fail to satisfy) an open-textured rule, the more likely it is to have the same result. The more it can be distinguished from previous cases, the more likely it is to avoid the result in those cases. The more central a case is, the more likely it is to satisfy a rule.

To support these tasks, what information must be included in a case representation? All of these tasks require both detailed information and abstractions. First, a representation of the facts stated in the official case report is desirable. They are readily available, and all of them are potentially useful in comparing and contrasting cases. Second, abstractions from those facts should also be included, both those used by the court in reasoning from the facts to the relevant open-textured rules, and others derived from general domain knowledge. The more abstractions we include, the more similarities we will be able to find between cases. To distinguish between central and peripheral cases with regard to a particular rule, information about which features strengthen or weaken a case should also be included. The more central a plan is, the more likely it is to succeed. The court deciding each case and the date of the decision should be included. Including the date of the decision makes it possible to examine trends in the case law, and both date and jurisdiction could be used for comparing cases.

Our case representation includes all of these types of information. The case structure itself has fields for the name (the official legal citation), short-name (an abbreviated form of the citation for use in text), court, date, facts, strategies involved, and holdings of the case, as well as various indices (discussed in Chapter 4). Information about which features strengthen or weaken a case with regard to a particular strategy is stored separately, in *domain-knowledge*.

CHIRON's case structures are much like HYPO's legal case frames; both includes the official legal citation, a shortened form of the citation, the open-textured provisions interpreted by the

court ("claims" for HYPO, "strategies" for CHIRON), holdings, and facts. Both reflect the way that lawyers analyze cases. Law students are taught to "brief" cases, that is, to prepare short abstracts of cases including their important elements: the name, court, and date, the facts, the holdings, and the *ratio decidendi*, or reasoning in the case.

Neither system's case representation includes the court's reasoning. The Tax Court cases that CHIRON uses often have little explicit reasoning. Like HYPO, CHIRON captures some of the courts' reasoning along with general domain knowledge, without associating it with specific cases, in the form of knowledge about which features strengthen or weaken a case and the ways in which cases are compared and contrasted.

Part of the representation of *Hughston v. Commissioner*, one of the cases in CHIRON's case-base is given in Figure 3.3; for the full representation, see Appendix A, and for the original text of the case, see Appendix B.

The name field of the case gives its official legal citation, as it would appear in a legal brief or memorandum. This is used for output. The date field gives the year of the decision. The date is used for computing trends in the case law. The court deciding the case is also included, but that information is not used in the current implementation. The focus of this work is on substantive, rather than procedural, comparisons of cases. In future work, CHIRON's arguments could be refined by making use of this information, and to support more subtle arguments based on the identity of the court, it would also be useful to include the procedural context of each decision, as suggested in [Berman and Hafner, 1991].

The facts of the case are represented as a list of propositions. We have attempted to represent the facts as given in the case report as precisely as possible, plus abstractions from those facts that seem useful, based on the reasoning in the particular case, other cases, or general domain knowledge. The representation of the case facts is not exact – for example, the taxpayer in this case was employed by Shell "during 1947 and for some time prior thereto." The exact date is not given, nor is it relevant to the taxpayer's sale and purchase of two houses in 1947, so we simply approximate the starting date of his employment as 1946. In general, however, the goal is to stay as close to the facts given in the case report as possible.

The representation language used for case facts is the same as the language used for input. In the next section, we will discuss this language in more detail.

```
(make-case
 :name "Hughston v. Commissioner, T.C.M. (P-H) 50,188 (1950)"
 :short-name "Hughston"
 :court "Tax Court"
 :date 1950
 :facts '((occurs (individual-return return2) (1947) (1947))
  (occurs (taxpayer return2 Hughston) (1947) (1947))
  (occurs (house house1) (October 1945) (December 1947))
  (occurs (real-property house1) (October 1945) (December 1947))
  (occurs (spatial-part room1 house1) (1947) (1947))
  (occurs (bathroom room1) (1947) (1947))
  (occurs (room room1) (1947) (1947))
  (occurs (real-property room1) (1947) (1947))
  (occurs (floortype room1 tile) (1947) (1947))
  (occurs (house house2) (October 1945) (December 1947))
  (occurs (real-property house2) (October 1945) (December 1947))
  (occurs (physically-occupy Hughston house1) (October 1945)
          (February 27 1947))
  (occurs (physically-occupy Hughston house2)
          (February 28 1947) (1951))
  (occurs (employment employment3) (1946) (1950))
  (occurs (employer employment3 Shell-Oil-Company) (1946) (1950))
  (occurs (corporation Shell-Oil-Company) (1946) (1951))
  (occurs (employee employment3 Hughston) (1946) (1951))
  (occurs (location employment3 (Houston Texas USA)) (1946)
          (February 1947))
  (occurs (location employment3 (Midland Texas USA))
          (March 1947) (1950))
  (occurs (selling selling1) (February 27 1947) (February 27 1947))
  (occurs (seller selling1 Hughston)
          (February 27 1947) (February 27 1947))
  (occurs (object selling1 house1)
          (February 27 1947) (February 27 1947))
  (occurs (selling selling2) (February 28 1947) (February 28 1947))
  (occurs (buyer selling2 Hughston)
          (February 28 1947) (February 28 1947))
                          ... )
 :action-types '(:sale)
 :property-transferred '(:real-property :cash)
 :property-received '(:cash :real-property)
 :transfer-types
       '((:sale :real-property :cash)(:sale :cash :real-property))
 :strategies '(:like-kind-exchange)
 :holdings '((:like-kind-exchange :government)))
```

Figure 3.3: Part of CHIRON's representation of *Hughston v. Commissioner*.

3.4 Representing the facts

Designers of legal reasoning systems, like other artificial intelligence programs, face the question of how to write descriptions of the world that a computer can understand. They need a representation language, some specific domain knowledge, and a way of reasoning with that knowledge (whether nondeductive reasoning or a formal inference scheme). The fact representation is particularly important in a legal reasoning system, because comparing and contrasting the facts of cases is central to legal reasoning.

In this section, we describe CHIRON's representation language, give it a formal syntax and semantics, and discuss its basic concepts and axioms. Formal domain theories can help in implementing AI programs, even where the facts and axioms of the theory are not used as input to a theorem prover (c.f. discussion in [Davis, 1990, pages 12–14]). At a minimum, the precision required by a formal theory helps to enforce careful and consistent use of the symbols representing domain concepts. The more expressive the representation language, the greater the danger of representing equivalent facts differently (as noted also in [Branting, 1990a, page 113]). A formal analysis can help to prevent (or reveal) inconsistencies of this kind. In addition, like any precise description, a formalization can help the designer to write and debug a system, and help both the designer and others to analyze, evaluate, and extend it.

3.4.1 Syntax

Features such as space, time, action, permission, obligation, knowledge, belief, and intention are necessary in the legal domain, as other researchers have noted [McCarty, 1989b]. In addition, unpredictable idiosyncratic details occur in each new case, such as the fact that the kitchen in a house has a tile floor, or the taxpayer is a war veteran, or the taxpayer has two children, one of whom is ten years old. To represent the facts as given in the case reports as accurately as possible, we need a representation language that is both flexible and extensible.

In order to represent the basic features of the domain, such as time, action, permission, obligation, knowledge, belief, and intention, we use an extension of the temporal logic developed by Shoham in [Shoham, 1988], modified to incorporate the modal operators "know," "believe," "want" (or "goal"), and "obligated." This language is influenced by McDermott's temporal logic and also by the work of McCarty in representing legal concepts, especially [McCarty, 1989b, McCarty, 1977].

The syntax of our language is similar to Shoham's [Shoham, 1988, pages 43–44]. For convenience, we repeat his definitions here, incorporating our changes (as noted):

Given:

TC: a set of time point symbols;
C: a set of constant symbols that is disjoint from TC;
TV: a set of temporal variables;
V: a set of variables that is disjoint from TV;
TF: a set of temporal function symbols (typical ones being the arithmetic operators), each associated with a fixed arity;
F: a set of function symbols that is disjoint from TF, each associated with a fixed arity; and
R: a set of relation symbols, each with a fixed arity,

The set of *temporal terms* is defined inductively as follows:

1. All members of TC are temporal terms.
2. All members of TV are temporal terms.
3. If $trm_1, \ldots, trm_n$ are temporal terms, and f $\in$ TF is an n-ary function symbol, then (f $trm_1, \ldots, trm_n$) is a temporal term.

The set of *nontemporal terms* is defined in exactly the same way, with TC replaced by C, TV replaced by V, and TF replaced by F.

The set of *nonmodal well-formed formulas* (nonmodal wffs) is defined inductively as follows:

1. If trm_a and trm_b are temporal terms, then $trm_a = trm_b$ and $trm_a \preceq trm_b$ are nonmodal wffs.
2. If trm_a and trm_b are temporal terms, $trm_1, \ldots, trm_n$ are nontemporal terms, and r $\in$ R an n-ary relation symbol, then (occurs (r $trm_1, \ldots, trm_n$) trm_a trm_b) is a nonmodal wff.
3. If φ_1 and φ_2 are nonmodal wffs, then so are (and φ_1 φ_2) and (not φ_1).
4. If φ is a nonmodal wff and z $\in$ TV $\bigcup$ V is a variable, then $\forall$z φ is a nonmodal wff.

We have modified Shoham's notation slightly to make it more consistent with Lisp usage, since our formulas are being used as part of a Lisp system. We use the infix notation:

(f $trm_1, \ldots, trm_n$) and
(occurs (r $trm_1, \ldots, trm_n$) trm_a trm_b)

rather than Shoham's

f ($trm_1, \ldots, trm_n$) and
TRUE(trm_a, trm_b r($trm_1, \ldots, trm_n$)),

and Lisp's logical operators (e.g., "(and ϕ_1 ϕ_2)") rather than the more traditional logical notation (e.g. "$\phi_1 \wedge \phi_2$").

In addition, we refer to "nonmodal wffs" rather than "wffs" because we have added the modal operators "want," "know," "believe," and "obligated" to the symbols in the language.

The set of *modal well-formed formulas* (modal wffs) is defined as follows:

1. If O is one of the modal operators "want," "know," and "believe," x is a nontemporal term denoting some person, φ is either a nonmodal wff or another modal wff, and trm_a and trm_b are temporal terms, then $(O\ \varphi\ x\ trm_a\ trm_b)$ is a modal wff.

2. If O is the modal operator "obligated," y is a nontemporal term denoting some body of rules, φ is either a nonmodal wff or another modal wff, and trm_a and trm_b are temporal terms, then $(O\ \varphi\ y\ trm_a\ trm_b)$ is a modal wff.

For example, in the representation of *Trisko v. Commissioner,* one of the cases in CHIRON's case base, we have:

(prohibited
(occurs (physically-occupy Trisko house1)
(June 1951) (October 9 1951))
Law (June 1951) (October 9 1951))

Using a more functional notation, this would be expressed:

TRUE((June 1951), (October 9 1951)
(prohibited Law TRUE((June 1951), (October 9 1951)
physically-occupy(Trisko, house1))))

Any modal or nonmodal wff is a well-formed formula in the language.

3.4.2 Semantics

Informal semantics

Intuitively, the semantics of our language can be understood as follows. We are given a set that includes all of the possible worlds. Possible worlds have a temporal dimension. That is, each possible world is a complete possible history of the world, like a timeline extending infinitely far into the past and the future.

In all the worlds that are knowledge-accessible to an agent x, the propositions that x knows about the past, present, and future all hold. All other propositions vary from world to world. Similarly, in all the worlds that are belief-accessible to x (which may not include the "real" world, if x has beliefs that are inconsistent with reality) all of x's beliefs hold. Propositions about which x has no particular opinion vary from world to world. The knowledge-accessible worlds are a subset of the belief-accessible worlds, since knowledge implies belief. The propositions that x wants to be true are true in all the "want-accessible" worlds, the propositions x wants to be false are false, and propositions about which x is indifferent vary from world to world. Finally, the propositions that are obligated to be true according to a certain body of rules are true in all the obligated-accessible worlds, the propositions that are prohibited are false, and the propositions that are permitted are true in some worlds and false in others.

Formal semantics

Formally, the semantics of our language are as follows:

Let D be a domain of individuals, and let $P \subset D$ be a (nonempty) subset of D consisting of all of the persons in D. let $R \subset D$ be a (nonempty) subset of D consisting of all of the bodies of rules in D (e.g., religion, ethics, and law). Let PW be the set of all possible worlds. With each $x \in P$ we associate three relations on PW, B_x, W_x, and K_x, corresponding to the modal operators "believe," "want," and "know," respectively. With each $y \in R$ we associate a relation on PW, OB_y, corresponding to the modal operator "obligated." Let O represent any one of the four modal operators, and let O_x represent any of the four relations. Each of these relations is serial, i.e., it has the property that from any given world at any time, at least one other world is accessible: $\forall w_i \in PW, \forall t_i(\exists w_j \in PW(O_x\ w_i\ w_j\ t_i))$. An interpretation I is a function that maps the nonlogical symbols in the language to some element of D.

Given these definitions, a sentence ϕ is true in a world $w_i \in PW$ under an interpretation I and a variable assignment VA if and only if one of the following is true:

1. ϕ has the form $trm_1 = trm_2$ and $I(w_i, trm_1) = I(w_i, trm_2)$.

2. ϕ has the form $trm_1 \preceq trm_2$ and $I(w_i, trm_1) \leq I(w_i, trm_2)$.

3. ϕ has the form (occurs (r $trm_1, \ldots, trm_n$) trm_a trm_b) and the relation $I(w_i, I(trm_a), I(trm_b), r)$ holds on $I(w_i, I(trm_a), I(trm_b), trm_1)$ through $I(w_i, I(trm_a), I(trm_b), trm_n)$.

4. ϕ has the form $(O\ \zeta\ x\ trm_a\ trm_b)$, where O_x is the relation corresponding to O, and ζ is true in all worlds w_j such that $\forall t_i, trm_a \leq t_i \leq trm_b, (O_x\ w_i\ w_j\ t_i)$.

5. ϕ has the form (and $\phi_1\phi_2$), and both ϕ_1 and ϕ_2 are true.

6. ϕ has the form (not ϕ_1), and ϕ_1 is false.

7. ϕ has the form $\forall z\phi_1$, and ϕ is true under all variable assignments VA$'$ that agree with VA everywhere except possibly on z.

3.4.3 Basic concepts

As stated above, we use some key concepts from deontic logic: "obligated," "permitted," and "prohibited." (Cf. [Wright, 1951]). "Prohibited" and "permitted" are defined in terms of "obligated" in the usual way: you are prohibited from performing an action if you are obligated *not* to perform it, and you are permitted to perform it if you are not prohibited from performing it (i.e., not obligated *not* to perform it).

We make no distinction between states and actions; both are represented as sets of intervals. For convenience, we will refer to both as propositions. (Cf. [Shoham, 1988].) Thus, the proposition (physically-occupy Jones house17) is the set of intervals during which Jones occupies house17, in all possible worlds; and the formula "(occurs (physically-occupy Jones house17) t_i t_j)" holds in a given world if the interval (t_i, t_j) in that world belongs to that set. The negation of a proposition is the complement of the set of intervals represented by the proposition. Thus, the proposition "(not (physically-occupy Jones house17))" is the set of all intervals during which Jones does *not* occupy house17.

Timing is critical in tax problems, so it is necessary to represent the time of actions and events. Our temporal logic is interval-based. The token "*now*" is used to indicate the time at which the plan is being constructed, some point in time later than all the cases in the case base and earlier than the performance of the client's intended actions. The tokens "*beginning-of-time*" and "*end-of-time*" are arbitrarily assigned the dates of January 1 in the year 1 and

December 31, 99999, respectively. A fact that is always true is said to hold over the interval from *beginning-of-time* to *end-of-time*. The endpoints of these intervals correspond to points in time. For time constants, we use the year (e.g., (1776)), the month and year (e.g., (July 1776)), or the month, day, and year (e.g., (July 4 1776)). For our purposes, this is sufficiently precise; it would be simple to add the time of day, if that became necessary.

Certain intervals of time are also important in this domain: the length of time the taxpayer has owned a particular piece of property, the length of time during which the taxpayer occupied a piece of property, the amount of time the taxpayer had been away from a piece of property before selling it, the time elapsed between one sale and another, and so forth. Intervals are represented as triples (years, months, and days).

Note that actions and obligations may have distinct times. Generally, they will coincide, because as a rule, we want to reason about the relationship between an agent's actions and the obligations or prohibitions that affected those actions at the time they were performed. But sometimes a legislature will pass a law now obligating or prohibiting the performance of a certain action next year, and the fact that that prohibition is in effect now will affect plans made for the future, so it can be useful to express the distinction.

We must also represent various types of property, property rights such as ownership and possession, and the ways in which those rights can be transferred. The United States' income tax is transfer-based; that is, tax is triggered by the transfer of money or property from one legal entity (individual, corporation, trust, etc.) to another [Chirelstein, 1988]. Concepts of property, property rights, and property transfers are central to this domain. Some of these concepts we had already formalized [Sanders, 1989a, Sanders, 1989b]; others, we added as necessary.

Part of CHIRON's property-type hierarchy is given in Figure 3.4. We divide property into categories which are often treated differently by the tax system: real property, tangible personal property, and intangible personal property. Intuitively, real property includes land and buildings: houses, condominiums, cooperatives, and so on. Tangible personal property includes tangible property other than real property, such as cars or books. And intangible personal property includes cash, stocks, and the rights of a tenant to his or her rent-controlled apartment. A complete list of the rules defining this hierarchy is given in Appendix E.

We take "own" and "possess" as primitives. Intuitively, ownership is a collection of legal rights, including the right to possess, use, give, rent, or sell an object. By contrast, you possess an object if you physically control it. Thus, if you rent a house, you possess the house, even though it is still owned by the landlord. If you borrow a book, you possess it, even though you

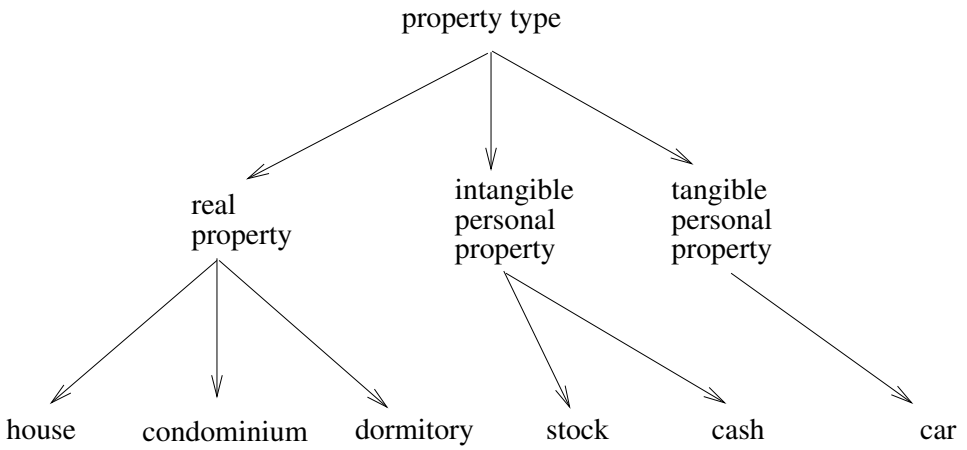

Figure 3.4: Part of the property-type hierarchy.

do not own it.

We classify transactions according to the type of rights and the type of property transferred. A transfer of property can be a sale, gift, rental, or loan. For this purpose, we define a sale as a transfer of permanent ownership of property in exchange for a price; a gift, as a transfer of permanent ownership of property in exchange for which the transferor receives nothing; a rental, as a temporary transfer of possession in exchange for a price; and a loan, as a temporary transfer of possession in exchange for which the transferor receives nothing. The transferors are individuals who are legally capable of owning property, that is, human beings, partnerships, corporations, and so forth. Services are generally performed by individuals, though a corporation or partnership may have its employees perform services on its behalf. The property transferred is anything that can be owned, tangible or intangible. Property can be transferred in exchange for cash, other property, or services; services can be transferred in exchange for cash, property, or services. A diagram of the possible transfer types can be found in Figure 3.5.

We reify these transactions. For example, in the case *Hughston v. Commissioner*, T.C.M. (P-H) 50,188 (1950), the taxpayer sold a house. Part of the facts of that case are represented as follows:

```
(occurs (selling selling1)
                        (February 27 1947) (February 27 1947))
(occurs (seller selling1 Hughston)
                        (February 27 1947) (February 27 1947))
```

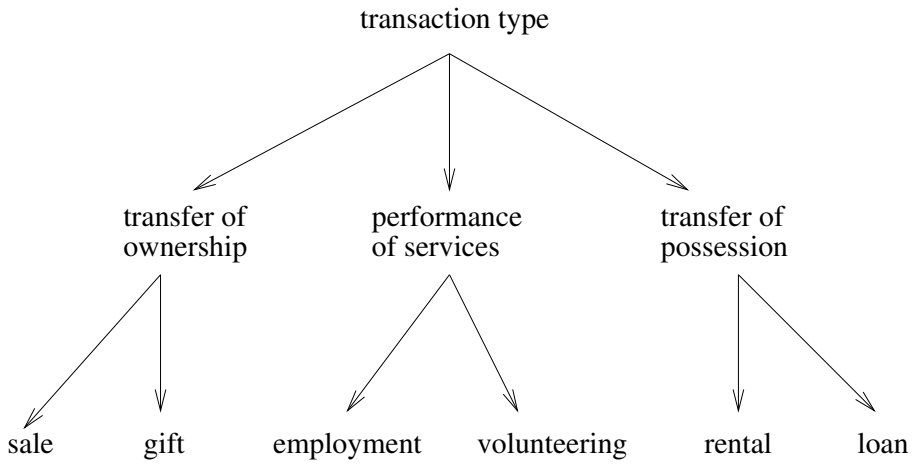

Figure 3.5: Transfers that may raise income tax issues.

```
(occurs (object selling1 house1)
                        (February 27 1947) (February 27 1947))
```

In other words, there was a sale on February 27, 1947, the seller was Hughston, and the object of the sale was a certain house, house1.

In this representation, the seller, the object, the payment, and the buyer are all expressed as separate relations to the reified "selling." Rentals, gifts, loans, employment transactions, and volunteering are represented similarly. This approach allows us to represent transactions where, as here, part of the information is unknown. Because there was a sale, there must have been a buyer, but no information about the buyer is given in the case, and it is not needed to reason about the tax consequences of the sale to the seller. We need a representation that can handle this incompleteness. The representation also allows us to represent transactions where there is more than one individual with the same role. For example, if the property being sold in the above case had been jointly-owned, so that there were two sellers, we would simply add a second "seller":

```
(occurs (seller selling1 Hughston)
                        (February 27 1947) (February 27 1947))
(occurs (seller selling1 Smith)
                        (February 27 1947) (February 27 1947))
```

In other transactions, it is important that the taxpayer received both real estate and cash in payment. This was true in a case called *Sayre v. United States*, 163 F.Supp. 495 (S.D.W.Va.

1958), where the taxpayer exchanged a farm in West Virginia for a farm in Ohio, plus some cash. The relevant portion of the representation of this case is:

```
(occurs (payment selling1 farm2) (1951) (1951))
(occurs (payment selling1 cash1) (1951) (1951))
(occurs (cash cash1) (1951) (1951))
(occurs (farm farm2) (1951) (1951))
(occurs (real-property farm2) (1951) (1951))
```

Situations where there are two buyers, two pieces of property are sold, two things are received as part of the same gift, and so forth, are handled similarly.

Note that in the above representation, the facts that farm2, one of the pieces of property received by the taxpayer, is a farm and is also a piece of real property, are both included in the case representation. For each object in the case representation, both its types and abstractions from those types are included.

In reasoning about property, we have also found it necessary to represent certain basic spatial concepts: location, nearness, distance, area, and the idea that one piece of property is part of another, which we call "spatial-part." For example, in *Hughston*, the taxpayer sold a 2400-square-foot house in Houston and bought another house in Midland, Texas, five hundred miles away. This portion of *Hughston*'s facts is represented as:

```
(occurs (location house1 (Houston Texas USA))
                                (October 1945) (December 1947))
(occurs (area house1 (2400 square-feet)) (1947) (1947))
(occurs (location house1 (Houston Texas USA))
                                (October 1945) (December 1947))
(occurs (distance (Houston Texas USA) (Midland Texas USA)
                                    (500 miles)) (1946) (1951))
```

Other concepts related to property that we have found useful include physical descriptions, such as the type of floor in a room and the fact that one piece of property contributes part of the value of a whole package. For example, in comparing the two houses bought and sold by the taxpayer in *Hughston*, the court noted that one had a tiled kitchen and the other did not:

```
(occurs (house house1) (October 1945) (December 1947))
(occurs (spatial-part room5 house1) (1947) (1947))
(occurs (kitchen room5) (1947) (1947))
```

```
(occurs (floortype room1 tile) (1947) (1947))

(occurs (house house2) (October 1945) (December 1947))
(occurs (spatial-part room7 house2) (1947) (1947))
(occurs (kitchen room7) (1947) (1947))
(occurs (not (floortype room7 tile)) (1947) (1947))
```

The court also noted that one of the houses had three bathrooms, plus a half-bath, and the other had only one:

```
(occurs (number-of-bathrooms house1 3.5) (1947) (1947))
(occurs (number-of-bathrooms house2 1) (1947) (1947))
```

We have also found it useful, particularly in the context of employment transactions, to express knowledge about types of organizations, occupations, and work. Employers include corporations, universities, and the government, for-profit and tax-exempt organizations. For examples, in *Hughston*, the taxpayer worked for the Shell Oil Corporation. These facts are represented as follows:

```
(occurs (employment employment3) (1946) (1950))
(occurs (employer employment3 Shell-Oil-Company)
                                        (1946) (1950))
(occurs (corporation Shell-Oil-Company) (1946) (1951))
(occurs (employee employment3 Hughston) (1946) (1951))
```

Types of services performed all fall into the general category of "labor"; they include, for example, research, teaching, and the practice of law. The taxpayer in *Hughston* was in-house counsel for Shell:

```
(occurs (services employment3 task1) (1946) (1951))
(occurs (practice-of-law task1) (1946) (1951))
```

Employees also fall into certain categories. Hughston was a lawyer:

```
(occurs (lawyer Hughston) (1946) (1951))
```

He was also an individual and a taxpayer. Individuals in other cases fall into other categories, such as "war veteran." They are students at particular institutions and parents or children of other individuals.

Some tax provisions relate to age, so we define predicates to express a person (or thing)'s age and the fact that one person (or thing) is older than another:

```
(occurs (age child1 10) (1947) (1947))
(occurs (older child1 child2) (1947) (1947))
```

Within the tax domain, we are focusing on a cluster of provisions having to do with residential housing. Actions that are important in this domain include maintaining and occupying the property in question. In *Trisko v. Commissioner*, 29 T.C. 515 (1957), the taxpayer occupied his house for several years, then moved abroad temporarily and rented the house to a tenant who took care of it in his absence, and then found when he returned that a strict rent-control law had been passed that had the effect of prohibiting him from returning home. A portion of these facts is represented as follows:

```
(occurs (physically-occupy Trisko house1)
(October 1944) (February 1948))
(occurs (maintain lessee2 house1)
                              (February 1948) (October 9 1951))
(prohibited (occurs (physically-occupy Trisko house1)
                              (1951) (1951)) Law (1951) (1951))
```

In some cases, the taxpayer is occupying more than one residence at a particular time (for example, the date of sale); if the number of the taxpayer's residences is given in the case, we represent that as well.

Finally, there are certain primitive states and events that are specifically related to the tax domain: filing and signing of forms, being a taxpayer on a particular return, and types of returns.

A complete list of CHIRON's vocabulary is given in Appendix D.

3.4.4 Axioms

We use an S5 axiom set for "know," weak S5 for "believe," T without veridicality for "want" and "obligated," and the inference rules modus ponens and universal instantiation (cf. [Hughes and Cresswell, 1968]). In addition, we use an Axiom of Noncontradiction for "want," "believe," and "obligated": $(O\ \phi\ x\ t_1\ t_2)\ \rightarrow\ \neg(O\ \neg\ \phi\ x\ t_1\ t_2)$ (where "O" is either "believe," "want," or "obligated."). Thus, for example, if during the interval $(t_1,\ t_2)$ you want some proposition ϕ to be true, you do not simultaneously want its negation to be true. In the case of "know," it

follows from the axiom of veridicality that if you know some proposition, you do not also know its negation. The Axiom of Noncontradiction provides a similar rule for "want," "believe," and "obligated."

These axioms constrain what we can say in our language. So far, they allow more inferences than we make. Tax cases rarely involve reasoning about the taxpayer's mental state, for example. We could simply delete the axioms that aren't used on the grounds that they are superfluous; but the axioms provide the basis for the semantics of the language (in that the accessibility relation is an equivalence relation). Moreover, the extra generality will become useful if the system's case-base is extended or the language is used in domains such as criminal or tort law.

CHIRON's proper axioms include inferences about property types, types of work, types of employer, and types of transaction. These are implemented as forward-chaining rules that fire when the input facts are first premissed. Thus, for example, if the input facts include the fact that a given piece of property is a house, then the rule that houses are real property will fire, and the fact that the given house is also real property will be added to the knowledge base.

Second, the system makes inferences about facts that are relevant to particular strategies. These inferences are implemented as Lisp functions called by the case-based planner, rather than as forward or backward-chaining rules, but they can be formalized in the same way.

Most of the case-based planner's inferences involve reasoning about time: for example, determining how long the taxpayer has occupied a given piece of property, how much time passed between the sale of one piece of property and the purchase of another, and what the taxpayer's age is today, based on his age at some earlier date.

All of these temporal inferences are based on a few basic operations. Time points are partially ordered; we postulate that this partial order is consistent with the order of the propositions associated with the time points, so any inferences that we draw about the ordering of time points holds for the corresponding propositions as well.

Basic operations on time points include determining whether one is earlier than another and finding the earliest of a finite set of time points. In addition, using these operations, we can compute the overlap (if any) of two intervals. Thus, we can determine for how long during a given interval (say five years before a house was sold) a given fact was true (say, the taxpayer occupied the house). To find out how long a fact was true before a given time point t, we compute the overlap of the fact's time interval with the interval (*beginning-of-time*, t).

We can also compute the difference between two timepoints, if they are ordered with respect to each other, and that difference is the length, or duration, of the time interval between the

two points. Durations are totally ordered. We can determine whether one duration is shorter than another and find the shortest of a finite set of durations. We can also add two durations, or compute the sum of a finite set of durations. Finally, given a duration and a time point, we can compute the interval of the given length that ends at that time point.

In addition to temporal inferences, the case-based planner performs a second type of inference: it counts the instances of certain propositions in the input. For example, with respect to the rollover strategy, it computes the number of residences the taxpayer has at the date of sale by retrieving all the instances of the taxpayer occupying any residence on that date, and counting them. With respect to the §121 exclusion, CHIRON determines how many times the taxpayer has filed Form 2119, again, by counting the instances of the corresponding fact in the input. This kind of inference might be difficult to formalize in general, but since the number of possible instances of each type of fact is bounded, the problem becomes relatively simple. If we make a closed-world assumption, that all the relevant facts having to do with the taxpayer's residences, tax forms filed, and so forth, are included in the input; and a unique-names assumption, that any two distinct terms can be assumed to be unequal, then we can write axioms to cover all possible cases. For example, if we wanted to count the number of residences the taxpayer had in a year, we could start with an axiom for the case where the taxpayer has 365 different residences:

$\forall$ taxpayer, place1, ..., place365, start-year, end-year, t1, ... ,t730,
((occurs (physically-occupy taxpayer place1) t1 t2) $\wedge$ ...
(occurs (physically-occupy taxpayer place365) t729 t730) $\wedge$ ...
(not (equal place1 place2)) $\wedge$ ...
(not (equal place364 place365) $\wedge$ ...
(start-year <= t1 <= t2 <= end-year) $\wedge$ ...
(start-year <= t729 <= t730 <= end-year) $\wedge$
$\rightarrow$ (occurs (number-of-residences taxpayer 365) start-year end-year))

A travelling salesman who sleeps in a different hotel every night might come close to this number; but in practice, most taxpayers have far fewer residences. If we write a similar axiom for each of the numbers from one to 364, we can be reasonably certain that we have covered all the possibilities. Other examples, such as filing tax forms, can be handled similarly.

3.4.5 Facts and rules

In addition to inferring facts from the input, CHIRON reasons about the relationship between facts and open-textured rules. The system first generates a plan and then reasons about the strength of the plan by comparing the facts of the current situation and those of previous cases. This process can be formalized as follows. Suppose in addition to our logical language, we have a meta-language with three relations, "supports," "sufficiently-supports," and "supports-more-strongly." Given two propositions p and q in our language and an open-textured rule S, we might have:

(supports p S)
(supports-more-strongly q p S)
(supports-equally-strongly q p S)
(sufficiently-supports q S)

In other words, proposition p is a fact that is known to support the conclusion that S has been satisfied, proposition q provides even stronger (or equal) support for the same conclusion, and proposition q is sufficient to conclude that S has been satisfied.

We assume that for some facts and strategies, it is known either that the given fact supports the given rule or that it weakens it. For some facts, this information may be unavailable. For each strategy, we have a conjunction of facts that sufficiently supports the strategy, that is, the prototype. Suppose the facts of the prototype plan for a given strategy in the current situation are $p_1 \wedge p_2 \wedge \ldots \wedge p_n$, and the facts of the plan generated by CHIRON are $q_1 \wedge q_2 \wedge \ldots \wedge q_m$. For each fact q_i of the suggested plan, CHIRON compares it with the corresponding fact p_i of the prototype. If the prototype is stronger than the suggested plan with respect to any of these facts, that is, if (supports-more-strongly p_i q_i S), then CHIRON looks for a case C where the corresponding fact was at least as weak as the fact in the suggested plan and the taxpayer was successful, or more formally, a case C whose facts are $r_1 \wedge r_2 \wedge \ldots \wedge r_k$, such that

((supports-more-strongly q_i r_i S) $\vee$
(supports-equally-strongly q_i r_i S)) $\wedge$
(sufficiently-supports $r_1 \wedge r_2 \wedge \ldots \wedge r_k$ S).

If the plan satisfies this test, CHIRON then builds arguments for and against its success, again using the facts of previous cases. Given only the facts and result of each case, we can take it to represent any of a set of possible rules. For example, Case C can be said to represent the rule (sufficiently-supports r_1 S), the rule (sufficiently-supports r_2 S), the rule

(sufficiently-supports $r_1 \wedge r_2 \wedge \ldots \wedge r_k$ S), or any of the other rules that can be formed by conjoining one or more of the facts of the case.

CHIRON's arguments can be interpreted as arguments about which of these rules is correct for a given case. For example, when generating an argument in favor of a plan for the taxpayer, the system looks for a case that shares one or more of the facts of the current situation, where the taxpayer was successful in executing the given strategy. Suppose each of the facts of Case C matches the corresponding fact of the current situation: r_1 matches q_1. r_2 matches q_2, and so forth up to r_j and q_j. Then CHIRON generates the argument on behalf of the taxpayer that the taxpayer should win because the current situation shares facts q_1 through q_j with Case C and implicitly, that Case C stands for the rule

(sufficiently-supports $r_1 \wedge r_2 \wedge \ldots \wedge r_j$ S).

On behalf of the government, the system counters this argument by pointing out that fact q_k in the current situation and fact r_k in Case C do not match. In effect, the system is arguing, on behalf of the government, that the rule that Case C stands for should be

(sufficiently-supports $r_1 \wedge r_2 \wedge \ldots \wedge r_k$ S),

or some other rule that includes r_k. In support of the argument that some other rule should apply, the system can also cite other cases that share more or different facts with the current situation.

This is a brief attempt to formalize the model of legal argument used by CHIRON, which is similar to HYPO's. For further attempts to formalize legal argument generally, see [Loui *et al.*, 1993] and [Gordon, 1993].

3.5 Prototypes

In addition to representations of actual legal cases, the case base includes prototype cases. Prototypes are represented using the same structure as cases, but indexed separately in *prototype-index*. They are indexed under a particular strategy, such as the §1034 rollover, and their facts are a conservative, safe set of facts, very likely to satisfy the open-textured provisions setting forth that strategy. There is currently one prototype per strategy.

CHIRON's prototypes are not real cases. In Protos, actual previous cases are used as prototypes [Bareiss, 1988]. Both approaches are influenced by work in psychology suggesting that concepts are best represented by a set of more or less typical cases, interrelated by "family

resemblances" [Rosch and Mervis, 1975]. In law, the reported cases are not typical, almost by definition: the standard, typical cases do not go to court. None of the court cases in CHIRON's case base is suitable for use as a prototype. An interesting alternative would be to base prototypes on lawyers' experiences of more typical cases, as suggested in [Schlobohm and McCarty, 1989]. As discussed in Section 3.1, however, we have chosen not to include this type of knowledge in CHIRON.

Instead, CHIRON's prototypes are based on general domain knowledge. This knowledge includes the commonsense meaning of statutory phrases. Terms such as "principal residence," although they are underspecified, do have some commonsense meaning. Additional information is provided by the cases. In any law case, there are some easy questions that are not at issue. For example, the taxpayer's old house may be clearly his principal residence, while the new one is at issue, or vice versa. The easy questions give you some information about the prototypical case. And hard questions can also provide information. For example, if the issue in a case is whether a house can qualify as a principal residence if the owner is not living there, we can infer that actually living in the house is part of the prototype.

CHIRON's prototypes are templates, or general fact patterns, instantiated for each case. Similarly, in a treatise or regulation, a lawyer would find a generalized plan, with the details of particular cases to be filled in by the lawyer. Thus, the prototype rollover plan stored in CHIRON's knowledge base refers to a taxpayer selling a house, but the prototype CHIRON uses in constructing and reasoning about a particular plan would involve a specific person and property: Joan Cook selling 6 Jay Street, for example, or Martha Todd selling 32 11th Street. We could have stored these as (imaginary) concrete cases. CHIRON's first step after retrieving a concrete prototype would be to substitute in the individuals and property involved in the current situation, however, and after that the system's operation would be the same.

A prototype plan can be adapted by changing, adding, or subtracting facts. A fact can be adapted into any other comparable fact. Every predicate in CHIRON's vocabulary has an associated comparison function in the *domain-knowledge* module. These functions indicate whether two facts are comparable. Two facts are comparable if they match, or if one is stronger than the other. Retaining a distinction found in HYPO and in the psychological literature on concept learning, we use the term "dimensions" for facts about the current plan that have possible values along a continuous, ordered sequence, like the amount of time since the taxpayer has lived in a house, or values that are members of a set, such as the color of the house; and "features" for facts that are either present or not, such as whether the taxpayer is a war veteran. Some features are known to strengthen or weaken the taxpayer's case, if present,

```
(make-fact
 :name 'age
 :user-name "The age of a given person."
 :pattern '(occurs (age ?person ?num) ?age-st ?age-end)
 :supports nil
 :compare-fn #'(lambda (f1 f2 strategy  &aux age1 age2)
 (case strategy
       (:exclusion121
(setf age1 (second (parameters f1)))
(setf age2 (second (parameters f2)))
(cond ((not (equal (modality f1) (modality f2)))
       :unknown)
      ((not (equal (fact-negations f1)
   (fact-negations f2)))
       :unknown)
      ((not (equal (predicate f1) (predicate f2)))
       :unknown)
      ((equal age1 age2) :match)
      ((< age1 age2) :weaker)
      ((< age2 age1) :stronger)))
       (t
(cond ((and (equal (modality f1) (modality f2))
    (equal (fact-negations f1)
   (fact-negations f2))
    (equal (predicate f1) (predicate f2)))
       :match)
      (t :unknown))))))
```

Figure 3.6: The dimension "age."

and that information is also stored in the *domain-knowledge* module.

A sample dimension, "age," is given in Figure 3.6. Both dimensions and features are represented using this same fact structure. Each fact has a name under which it is indexed, a user-name, and a pattern (used by the system designer to ensure that facts of this type always take the same form, with the parameters in the same order, etc.). The supports-field is used for features, where they are known to support either the taxpayer or the government. And every fact has a compare-function that is called when comparing instances of the same fact. The strategy is a parameter to the compare-function, as well as the two facts being compared, because the comparison may vary depending on the strategy. In the case of the dimension

"age," for example, if the strategy is the §121 exclusion, the fact is stronger (i.e., supports the taxpayer's case more strongly) if the taxpayer is older. There is an explicit age requirement for that strategy. For other strategies, such as the §1034 rollover, a fact of this type either matches or it doesn't: the taxpayer's age may be mentioned in the statement of facts of a case, but there's no explicit age requirement.

Dimensions are suggested by cases. For example, the cases interpreting §1034 consider whether a house is your principal residence if you have one or more other residences, if you have not lived in the house for years and cannot move back because the rent control law prohibits it, if you have never lived in the house, etc. The corresponding dimensions are: number of the taxpayer's residences, amount of time since the taxpayer occupied this house, amount of time the taxpayer ever lived in the house.

The prototypes are related to the actual cases by the features and dimensions. If you start with the prototype, modify it along the appropriate dimensions, and add or subtract the appropriate features, you will obtain an actual case. Cases indicate the possible adaptations of the prototype; they also limit their extent. Negative cases — for example, one that holds that if you've been away from a house for five years, it ceases to be your principal residence, or one that holds that if you spend an equal amount of time in each of ten houses, none of them is your principal residence – limit the degree to which a plan can vary from the prototype.

3.6 Summary

CHIRON's approach to representing open-textured rules and cases builds on several previous approaches. CHIRON's solution combines rules, prototypes, cases, and dimensions, drawing on ideas from McCarty, Gardner, HYPO, GREBE, and CABARET.

Like Gardner, we use both rules and cases. Like Gardner's, CHIRON's case representations are indexed under the rules they interpret. Each rule corresponds to a set of cases. In Gardner's program, however, the set of cases is an unstructured one: her system has no dimensions, no prototype, and no levels of abstractions between the facts and the open-textured rules. In CHIRON, as in HYPO and CABARET, the cases are related to each other by the dimensions (or deformations).

We use prototypes and deformations, as suggested by McCarty. In addition, we incorporate an explicit facility for representing and reasoning with legal cases, like Gardner's program, GREBE, HYPO, and CABARET. Cases are indexed under the rules to which they refer, and retrieved when interpreting those rules. Cases both suggest and constrain the possible deformations of

the prototype. Deformations move cases along dimensions.

Our case structures are comparable to the legal case frames of HYPO and CABARET; both are motivated by the way in which law students are taught to summarize cases. Like HYPO and CABARET, CHIRON includes the official case citation, a short form of the case name, the facts, holdings, and claims or strategies involved, in its case representation.

CHIRON's representation supports HYPO's compare-and-contrast algorithm. Like TAXMAN II's, CHIRON's fact representation language has a formal syntax and semantics, and it has the detail and extensibility of both TAXMAN II and GREBE. A translation from CHIRON's representation language into GREBE's would be straightforward.[2] Conversely, GREBE's representation could be translated into CHIRON's; the only difficulty would be the absence of temporal information. GREBE's representation apparently does not include the times of states and actions, presumably because reasoning about time is less important in GREBE's workers' compensation domain than it is for tax problems. If we supply the temporal information, either from the case reports or by assigning arbitrary dates, then GREBE's representation could be given the formal syntax and semantics set forth here.

CHIRON's representation includes facts mentioned in the official case report even if their relevance is not immediately obvious: for example, the fact that the taxpayer in *Trisko v. Commissioner* is a war veteran. Like GREBE's, CHIRON's representation includes facts at varying levels of abstraction; unlike GREBE's, CHIRON's abstractions are not tied to the reasoning in any particular case.

Because CHIRON's representation combines all of these features, it supports all of the tasks set forth earlier in this chapter as desiderata. Using this representation, CHIRON can:

- distinguish between two cases;
- show similarities between two or more cases;
- indicate which cases are central and which are peripheral; and
- determine trends in a series of cases illustrating a given rule.

None of the previous systems combined all of these features, and as a result, although all of them supported some of these tasks, none of them could support all.

[2]For a simple algorithm for translating from a logical representation into GREBE's semantic nets, see [Branting, 1990a, page 30].

Chapter 4

Open-textured planning

4.1 Introduction

CHIRON is designed to be a lawyer's assistant. The hypothetical user of the system, a lawyer, begins by entering information about a taxpayer's goals and tax situation. Given this input, the system generates plans by which the taxpayer can reduce his or her taxes, annotated with citations to relevant cases and statutes, and arguments for and against the success of each strategy in the current situation.

CHIRON's design is strongly influenced by the system's task environment, in particular, the open texture of legal rules. Lawyers use legal rules consciously, argue about them, and disagree about what they mean. Cases partially define the rules; they bridge the gap between open-textured rules and facts. Cases also extend and limit the rules. Positive cases extend the rules to cover new sets of facts; negative cases limit how far the rules can be stretched. Law is an adversarial domain, so while one side (the taxpayer, in our case) tries to extend the rules, the other (the government) tries to limit them.

We want our planner to reason with representations of legal rules and cases, and to use the cases to define, extend, and limit the rules. Accordingly, in CHIRON, we have combined hierarchical and case-based planners in a hybrid system. The hierarchical planner reasons with representations of open-textured rules; it also reasons with representations of facts. The case-based planner bridges the gap between the two, as legal cases bridge the gap between open-textured rules and facts.

First, the hierarchical planner takes the taxpayer's goals and background information, adds the general system goal (reduce taxes), and constructs a partial plan based on the statutory

tax rules. There are a number of plans it could consider; for guidance, it asks the case-based module for a list of plans tried in previous similar cases.

When the hierarchical planner has constructed a partial plan, the case-based planner retrieves a set of cases associated with that plan (cases in which previous taxpayers attempted, with or without success, to execute the same plan). To determine whether the current situation fits within the open-textured rules, the case-based planner compares and contrasts it with previous cases and generates arguments for and against the success of the plan in the current situation, based on the current situation's similarity or dissimilarity to the facts of those previous cases. If the arguments in favor of the plan are sufficient, the case-based planner generates a plan and returns it to the hierarchical planner. Finally, the hierarchical planner reduces the case-based plan to primitive actions.

Because law is an adversarial domain, it is useful to have a safe interpretation of the rules, and also to have some measure of how far you can deviate from that safe interpretation. Accordingly, we associate a prototype "safe-harbor" plan with each partial plan and use the previous cases as guidelines to show how far the prototype can be adapted.

Any legal reasoning system's ability to compare and contrast cases, generate arguments, and adapt plans is constrained by its fact representation. CHIRON's fact representation, as discussed in Chapter 3, is particularly detailed. The system's algorithms for comparing and contrasting cases and building arguments, based on the algorithms used in HYPO and CABARET, exploit this rich representation.

Besides the open texture of rules, another important feature of CHIRON's task environment is the fact that the hypothetical users of the system are experts. In general, experts are much more likely to use a program that offers advice than one that seems to take over part of their expertise; and lawyers are no exception to this rule. Since CHIRON is designed as a lawyer's assistant, it seemed best to give as much control as possible to the user.

CHIRON could have been designed to select the single "best" combination of strategies, construct a plan, and print it out without comment. It has a number of tax-reduction strategies, including, for example, selling a house and buying another to get the §1034 rollover, making a charitable donation, and selling a house and taking the §121 exclusion. These strategies can generally be used either individually or in combination. All of these strategies and all of their possible combinations are candidate plans.

Instead, the user is given a set of plans to choose from, rather than just one. In addition to suggesting individual strategies, the system informs the user which combinations of the suggested strategies have succeeded in previous cases. The user is allowed to suggest plans

or combinations of plans for the system to work on. The system attempts to construct a plan combining all the strategies requested by the user, and repeats this process as long as the user continues to request combinations. Finally, each plan constructed by the system is accompanied by arguments for and against its success in the current situation, so that the user can assess its merits. Since the system chooses strategies to analyze and constructs the individual and combined plans, the theoretical problems involved in legal planning are addressed, but without restricting the user any more than necessary.

In the following sections, we will describe the hierarchical planner, then the case-based planner, and finally the way in which the two modules have been combined.

4.2 The hierarchical planner

CHIRON's user begins by entering information about the taxpayer's current situation. Generally, the input includes both goals — the action or actions your client intends to perform — and some background information. Either of these elements can be omitted. The goals will be missing if the action has already been performed, for example if your client has already sold his house. The facts will be missing if you only know the intended actions and have no background information.

After reading in the client information entered by the lawyer, CHIRON's hierarchical planner constructs an abstract plan, including the taxpayer's goals, if any, and the general system goal of reducing the taxpayer's income tax. There are a number of ways this top-level plan can be refined: the system can attempt to show that no taxable income results from the transaction, or if that fails, exclude income, defer it, attribute it to someone other than the client, or find deductions or credits. There are more combined plans, since several of these strategies could apply simultaneously to the same transaction. Each of these strategies, in turn, can be refined in several ways. Some strategies can be repeated with different variable bindings; some can be repeated, but not during a given tax year; and others can't be repeated at all, such as the §121 exclusion, which can be used only once within the taxpayer's lifetime. The plan space is not deep, but it is potentially very wide.

For guidance in choosing among these transformations, the hierarchical planner consults the case-based module. The case-based module first classifies the transaction or transactions involved in the current situation: gift of real property, loan of cash, sale of tangible personal property, and so forth. All the cases in the case base are also classified according to the transactions involved. If any transaction was mentioned in the input, the case-based module retrieves

all the cases that involved the same type of transaction, compiles a list of the transformation sequences that the hierarchical planner would have used to reach those cases, and returns the list to the hierarchical planner. The hierarchical planner chooses a transformation, refines the plan, and repeats the process until no further transformations are applicable.

This process is similar to the case-guided search used in PRIAR or Veloso's system (discussed in Chapter 2), except that in CHIRON, cases are used as the basis for a solution to the current problem, as well as for search control. The hierarchical planner can only refine the plan up to a point; it then calls the case-based reasoner again, to continue the plan refinement.

4.3 Indexing, retrieval, and similarity

When it can't apply any more transformations, the hierarchical planner calls the case-based planner to refine its open-textured partial plan. The case-based planner starts by retrieving the corresponding prototype and cases. Prototypes and cases are indexed under the partial plans they interpret.

The features a system uses for indices should be those that allow it to retrieve a case when it is needed. If you use too many indices, you risk retrieving cases that you won't need; if too few, you risk missing something important. The choice of indices necessarily involves a judgment call, since at the time a case is indexed, the system designer cannot be certain when it will be needed.

CHIRON uses a range of indices. In part, this reflects the fact that cases can be retrieved for several different purposes, and these purposes sometimes require different indices. Besides partial plans, cases are indexed under the types of transactions they involve (e.g., sale of personal property, gift of real property, etc.) in order to guide the hierarchical planner's search. And they are indexed under all of their facts, both representations of the facts in the case report and abstractions and inferences from those facts.

Unlike most CBR systems, CHIRON does not choose a single "most similar" case on which to base its reasoning. For the task of case-guided search, for example, it retrieves all the cases involving the same type (or types) of transaction as the current situation. Cases that share the same transaction type are similar in a relevant way, since the United States income tax is based on transfers of property and services. Generally speaking, more than one of these cases is retrieved. Instead of choosing the most similar of the cases, CHIRON compiles information from all of them. All the strategies used in these cases are suggested to the hierarchical planner as possibilities.

When reasoning about a particular open-textured strategy, CHIRON also uses a group of cases. It starts by retrieving all the cases in which the taxpayer attempted (successfully or unsuccessfully) to execute that strategy. It then sorts those cases in order of their similarity to the current situation. But similarity, for CHIRON, is not a total order. As in HYPO and CABARET, similarity is measured in terms of the features that a case shares with the current situation. Like HYPO and CABARET, CHIRON uses a subset lattice to sort the cases. Those cases stored at the nodes closest to the root of the lattice are all most similar or "most-on-point" cases.

Consider the example shown in Figure 4.1. This is a very simple example of the kind of lattice used by CHIRON, HYPO, and CABARET. First, the system creates a root node, whose indices are the facts of the current situation. Here, in the current situation, a purple block is on a red block. Suppose we do not represent the colors of the blocks. Then the current situation will be represented as shown in the root node: one block is on top of another.

Next, the system retrieves all the cases that share any fact with the current situation and stores each of them in a node whose indices are the facts that that case shares with the current situation. Suppose we have four cases in our case base. In case 1, a purple block is on the table; case 2 involves a red block; case 3 involves a purple sphere; and case 4 features an orange sphere. Only case 1 and case 2, which involve blocks, will be retrieved. Case 1 will be stored in node1, because, like the current situation, it involves a block that is on top of another object. Case 2 will be stored in node2, because the only fact it shares with the current situation is that they both involve a block. Node1 is a parent of node2 because its indices (the facts it shares with the current situation) are a superset of node2's. Case 1, because it is stored at node1, is more similar to the current situation, or "more on point," than case 2.

Like HYPO and CABARET, CHIRON uses this lattice structure to support the task of comparing and contrasting the current situation with previous cases. Lawyers argue that their situation is similar to a previous case in order to obtain the same result and distinguish the current situation from previous cases when a different result is desired. HYPO and CABARET use this approach for analyzing cases; CHIRON has applied it to a planning context.

Like HYPO and CABARET, CHIRON argues that cases are similar by pointing out the features they have in common and distinguishes cases by pointing out unshared features and different values along shared dimensions. Here, for example, the system would argue that the current situation should be treated in the same way as Case 1, since in both cases, a block is on top of another object. To distinguish the cases, and argue for a different result, it would point out that in the current situation, a block is on top of another block, while in Case 1, the block is on the table.

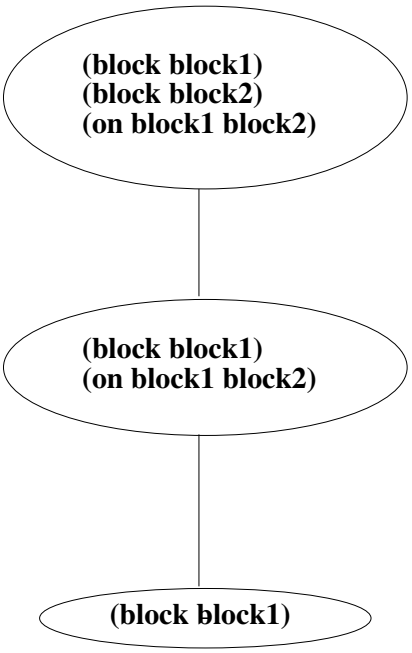

Figure 4.1: A simple case lattice.

Because CHIRON uses a particularly detailed case representation, as discussed in Chapter 3, it can find similarities and distinctions that a simpler representation would not. For example, suppose we add the colors of the objects to our representation of the above example. This would give us the case lattice shown in Figure 4.2. Case 1 is no longer stored at a node that is a parent of the node where Case 2 is found. Case 1 is stored in node1 because, like the current situation, it involves a purple block that is on top of another object. Case 2 is stored in node2 because it involves a red block. If Case 2 has the desired result and case 1 does not, this representation will be advantageous, because it allows us to make use of Case 2. Case 2 is no longer "trumped" by Case 1. Moreover, we can add Case 3 to our lattice. It is stored at node3 because, like the current situation, it involves a purple object (In the current situation, the object is a block; in the previous case, it was a sphere.) Note that this is not a complete subset lattice. For efficiency, we only create a node where there is at least one corresponding case.

CHIRON uses both surface facts (the facts given in the case report) and abstractions and inferences from those facts as indices. Cases sharing a concrete fact and its abstraction will always be stored in a parent or ancestor node of the node where cases that only share the abstraction can be found. In other words, they will be more similar, or "trumping" cases.

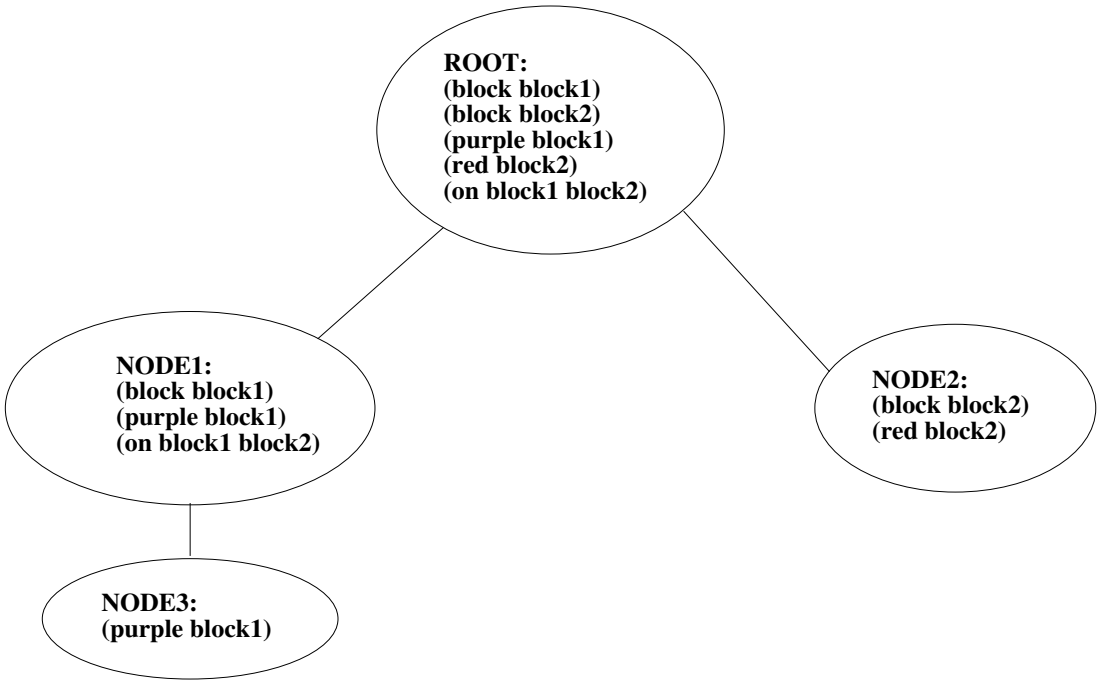

Figure 4.2: A case lattice with more facts.

For example, suppose we add to our case representation the fact that all of the blocks and spheres are "tangible property." Then we get the lattice shown in Figure 4.3. The addition of this abstraction allows you to point out a possibly important similarity between the current situation and previous cases. In addition, it allows you to consider potentially useful cases you would not otherwise have found. Adding "tangible property" to this lattice makes it possible for us to store all four cases in the lattice. The case of the orange sphere will be stored in node4 since, like the current situation, it involves a piece of tangible property. It is less similar than the other cases, which all involve tangible property as well, and share at least one additional fact with the current situation. But if the only case in our case base had been the orange sphere, we would not have been able to use it without the abstraction. It might be that the fact that both cases involve tangible property makes the previous case highly relevant, even though superficially they are not alike.

In general, indices should make it possible for the system to retrieve relevant cases. Here we define "relevant" as "useful," that is, cases that help you solve your problem. In a planning context, this means cases that either support your plan or warn you that it might fail.

How were CHIRON's indices chosen? At first, when it looks at the facts input by the user, CHIRON is faced with the gap between facts and rules in this domain. Since it has only

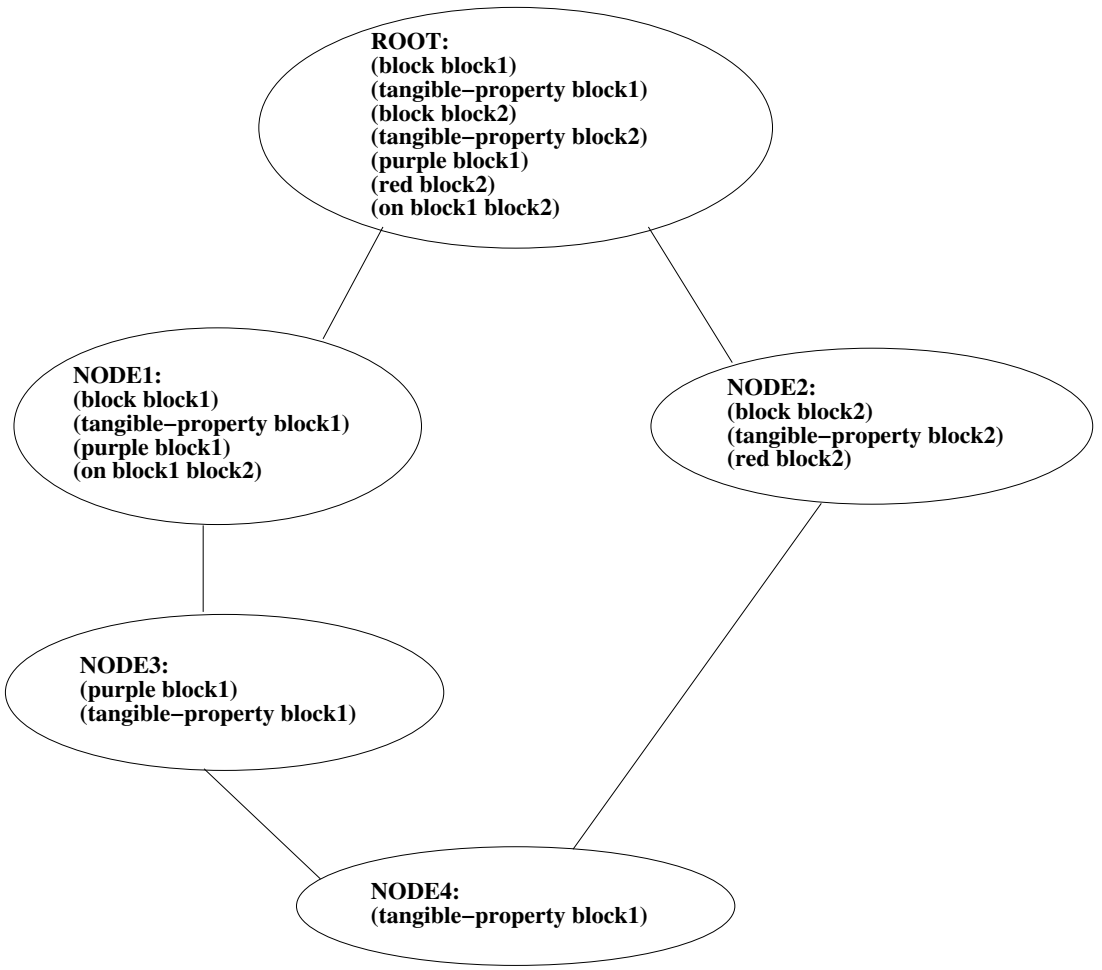

Figure 4.3: A case lattice with facts and abstractions.

facts to work with, it starts by comparing the input facts with the facts of previous cases. To get a preliminary idea of which strategies to consider, rather than do a detailed fact-by-fact comparison, it works at an abstract level, retrieving cases that share the same type of transaction. In this domain, we know that cases involving the same type of transaction are likely to be relevant.

When interpreting a particular strategy, or partial plan, CHIRON retrieves cases where the taxpayer attempted to execute that strategy. This is comparable to the standard approach of indexing plans under their goals, used in CHEF. Cases involving the same strategy are relevant to the success of the strategy in the current situation.

Like CHEF, CHIRON retrieves plans based on their goals first, and then looks at their facts.

CHIRON's fact representation is more detailed, reflecting the greater complexity of legal cases. As illustrated in Chapter 3, we start with a representation that corresponds closely to the facts given in the official case reports. In theory, any of these facts can be used in future arguments, so they should all be included.

CHIRON's abstractions mainly have to do with property types and types of transactions. Given the nature of the tax law, these are highly relevant. Categories of organization are also relevant: the fact that a taxpayer's employer is a corporation, or a nonprofit, or the government, for example. And we have also included various factors that are mentioned in the courts' reasoning in our cases, such as the amount of time that the taxpayer had been away from home at the time he sold his house (for purposes of §1034).

CHIRON's indices are ordered. It looks at the partial plans involved in a case first, and then at the facts it shares with the current situation. The use of ordering results in some loss of reasoning power. If it looked at facts first, it would retrieve some cases from other contexts, which might be useful. For example, when reasoning about §1034, we might also retrieve cases under §121, where the taxpayer must also establish that he or she has sold a principal residence.

The loss of reasoning power seems outweighed by the gain in efficiency, however. Practically speaking, you only want to look at cases interpreting different plans if you cannot find a closer match. Legal provisions are context-dependent; their meaning cannot be assumed to be the same wherever they appear [Gardner, 1987, page 53]. In a well-drafted statute such as the Internal Revenue Code, language is generally used in a consistent way, and cases interpreting one section are persuasive in arguing about another, but still, the results are not necessarily the same. We chose, therefore, to see how far the system could go using only cases from the current context. For interesting work on "cross-context remindings" in the domain of tort law, see [Cuthill, 1992].

Finally, CHIRON uses a static set of indices. Static indexing is easier to implement, and as a result, has been used in most CBR systems. Dynamic indexing gives the system the flexibility to solve problems involving a demand for information that is not originally accessible. Since CHIRON uses only limited ordering, and all of the facts and abstractions are used as indices, the loss in flexibility is minimal.

4.4 Evaluating and supporting plans

CHIRON has two uses for its claim lattices. First, like HYPO and CABARET, it uses them as

the basis for arguments for and against the application of a given open-textured provision to a particular situation. Second, since CHIRON is a planner, it also needs to decide whether to suggest a given plan at all; so it uses the lattice for a preliminary evaluation of the plan.

When deciding whether to suggest a plan, the hierarchical planner can determine whether a plan violates certain basic constraints, for example, by selling the same piece of property twice. The case-based planner has a more difficult decision to make, and it does not have rules to fall back on; it only knows that certain plans have succeeded or failed under certain circumstances.

The more case support we require, the more plans will be rejected, and the more control will be taken away from the user. The less case support we require, the more dubious plans will be suggested, and the less helpful the system's advice will be for the user. Fairly arbitrarily, we have decided on the following requirements: first, that there must be at least one most-on-point case with the desired result (i.e., in the tax domain, a case that held for the taxpayer); and second, that for each fact where the current situation is weaker than the prototype case, there must be either a favorable case that is at least as weak, or a trend in that direction.

There are three things to note about this standard. First, we are assuming that it is possible to argue about the weaknesses in a plan separately. In fact, they may be cumulative; a plan with several weaknesses, even if there is some support for each, may be too weak to consider. We compensate for this possibility by informing the user about each of the weaknesses; given the information, the user can decide whether to go ahead.

Second, the system does not attempt to find new creative arguments. For example, if the taxpayer lived in his house at the time of sale in every §1034 case, the system would not generate a plan that involves living somewhere else. It would be interesting to model this kind of creativity, but it is outside the scope of this thesis.

Third, we are making use of the timestamps on the cases. We define "trend" simply as two cases, one decided earlier than the other, where both decided in favor of the taxpayer, both were weaker than the prototype along a given dimension, and the second was weaker than the first. Taking into account the temporal dimension of the casebase is rare in CBR systems. Cuthill also uses timestamps: her system, CHASER, includes a preference for more recent cases in its retrieval algorithm [Cuthill, 1992]. Reasoning about the time of cases seems intuitive in legal reasoning systems; it could be useful for any system in a domain that changes over time.

CHIRON also uses its claim lattices as a basis for arguments for and against the success of its plans. The domains for which CBR is best suited are those with a weak domain theory: for example, law, medical diagnosis, cooking, and military strategy. CBR is effective in these

domains because we don't know enough about them to analyze a situation or generate a plan from first principles; and for the same reason, given a plan or analysis, we have no way of proving absolutely that it is correct.

In a domain where we cannot prove logically that the output of our programs is correct, we can still support our claims that they work. One method for doing this where CBR has a real advantage is so obvious it is easy to overlook: explaining for the user the way in which the system's analysis or plan was constructed, and why it was right. It is desirable for any system to be able to explain its operation. In rule-based expert systems, this is called an explanation facility. Typically, it involves displaying the rules applied in arriving at a conclusion [Waterman, 1986]. Such an explanation, although it increases the users' confidence in the system, can be opaque and difficult to follow.

CBR systems have the advantage that by using cases, they can provide an explanation that is generally much easier to understand and more compelling than a chain of rules. If a system uses cases to generate a plan or analyze a situation, simply presenting those cases to the user and pointing out the similarities between the cases and the current situation helps to persuade the user (whether it be the system designer or an independent user) that the system's output is reasonable.

There is evidence that people use cases in this way. For example, one study collected protocols of decisionmaking in four domains: design engineering, firefighting (urban and woodland), and tank platoons [Klein and Calderwood, 1988]. One of the ways these decision-makers used cases was to reassure themselves that their current plan would succeed. Similarly, in the legal domain, no memorandum or brief would be convincing unless it included citations to statutory and case law.

Perhaps because citations are essential in any legal writing, case-based legal analysis systems have generally included in their output a discussion of the cases used and how they relate to the current situation. HYPO, CABARET, GREBE, and CHASER all make use of this technique [Ashley, 1991, Rissland and Skalak, 1991, Branting, 1990a, Cuthill, 1992].

In those systems that use cases to generate a plan or analyze a situation, there is generally an implicit explanation facility. In CHEF, for example, an execution trace will show which cases were chosen and how they were used. Perhaps because these systems have been used primarily by their designers, who have access to execution traces and other internal details, no need is seen for an explicit explanation facility, and the value of cases for strengthening confidence in the operation of the program has been taken for granted.

One of CHIRON's contributions is that it incorporates an explicit explanation facility, similar

to HYPO's, in a case-based planner. Along with its plans, CHIRON prints out arguments for and against the success of each plan, based on comparing and contrasting the current situation with previous cases, and annotated with citations to the cases and statutes used.

4.5 Adaptation

If the arguments in favor of a given partial plan are sufficient, the case-based planner generates a plan for executing it in the current situation and returns its plan to the hierarchical planner.

Here, again, CHIRON uses a group of cases. Like Gardner's program, HYPO, and CABARET, it places the current situation within the context of a group of previous cases [Gardner, 1987, Ashley, 1991, Rissland and Skalak, 1991]. One of CHIRON's contributions is incorporating this technique in a problem-solving system. Most case-based reasoning systems have based their output — plans, diagnoses, explanations, or whatever — on a single "best" case. Some have combined pieces of two or more previous cases into a solution to their current problem (See, e.g., [Alterman, 1986, Branting, 1990a, Redmond, 1990, McCartney, 1993, Zito-Wolf and Alterman, 1992]). These systems choose the best case for each piece of their problem, rather than choosing a single best case and adapting it to solve the whole problem. CHIRON chooses several "best" cases and considers the whole current situation in relation to all of them.

For many problems, there is no single exact solution. Rather, there is a range of acceptable answers. Cases are not just the basis for a solution; they also indicate the boundaries within which a solution can be found. We solve problems by choosing some point within those boundaries.

For example, suppose you want to make chili for dinner. You have made the recipe with one tablespoon of chili powder and with three, and while you prefer the latter, both were acceptable. You can add the chili powder without measuring it precisely, as long as you stay within these limits. Or suppose you want to sell your house. Two similar houses have been sold recently in the same neighborhood, one for $150,000 and one for $200,000. You can estimate that your house will sell for between $150,000 and $200,000.

Similarly, the open-textured nature of legal rules enables them to cover a range of possibilities, without specifying them in complete detail. For example, a house can be your principal residence whether you've lived there for one year or fifty, whether it has one bedroom or a hundred, and so forth.

Suppose you want to take advantage of one of these open-textured rules. Say for example that you want to sell your house and obtain the benefits of §1034(a). You will consider your

situation in relation to past cases under this provision and try to construct a plan for selling your house that is supported by previous successful cases. Because the courts are bound by precedent, if your case is similar to or stronger than past successful cases, it will be decided the same way. If it is weaker than past successful cases, you might still win, but this is an adversarial domain. To avoid challenge by the government, you will try to stay within the boundaries indicated by those cases.

At least, that is what you will do if you are a typical conservative tax planner. If you are a little more aggressive, and the successful cases indicate a trend, you may go beyond them in the same direction. For example, if you have a successful case where someone sold a house they only lived in for a year, you may try selling your house after six months. In other words, you will extrapolate from the cases in your case base, rather than interpolating between them.

Cases often suggest the boundaries within which you want to construct a plan. This is particularly true for legal cases, because in this domain, reported cases are nearly always peripheral — cases where an open-textured plan is not satisfied mark the boundaries of a concept, cases where it is satisfied extend its boundaries. Central cases are easy, so they are usually not taken to court.

Within those boundaries, it is useful to have a safe interpretation of the rules: a plan that Gardner's system would accept as an "easy question," or what tax planners refer to as a "safe harbor." If you execute this plan, you will probably be safe from challenge by the government; the more you differ from it, the more likely you are to be challenged (and to lose). Our prototypes correspond to these safe harbor plans.

The further a plan is from the prototype, the more likely it is to fail. But how far is too far? We use cases to give us a measure of how much a plan can differ from the prototype. As discussed in Section 4.4, CHIRON rejects plans that are weaker than any of the successful cases in the case base, unless it can find at least a simple trend towards weakening cases along that dimension.

The adaptations used by CHIRON are suggested by the cases. Some adaptations consist simply of adding a fact to the prototype; any fact in one of the previous cases (e.g., the fact that the taxpayer was a war veteran) can potentially be added to the prototype to make a new plan. Other adaptations involve varying the parameters of a given fact. For example, the cases interpreting §1034 consider whether a house is a principal residence if you have one or more other residences or if you have not lived there in several years. The corresponding adaptations involve changing the number of residences the taxpayer has and changing the amount of time since you occupied the house.

Adaptations relate the prototypes to the cases and the cases to each other. If you start with the prototype and apply the appropriate adaptations, you will obtain an actual case. The cases suggest possible adaptations; they also limit their extent. Negative cases — for example, one that holds that if you've been away from a house for five years, it ceases to be your principal residence, or one that holds that if you spend an equal amount of time in each of ten houses, none of them is your principal residence – limit the degree to which a plan can vary from the prototype.

Within the set of acceptable possibilities, CHIRON constructs plans that are as similar to the prototype as possible. This strategy gives plans a conservative bias, which is consistent with much of tax planning. It is also consistent with the approach suggested by Kambhampati and Hendler, that adaptations should in general be conservative.

In summary, CHIRON uses a prototype, or safe harbor plan, representations of actual legal cases, and HYPO-like dimensions to define a space of possible plans and constructs plans to fit within that space. This solution is especially well-suited to adversarial domains, where varying too far from the prototype may cause a plan to be challenged. The basic idea of planning within boundaries could also be translated into a domain without explicit adversaries, such as cooking.

4.6 Combining strategies

The cases suggest positive and negative interactions among the rules. For example, if you characterize a payment related to business property as "rent" you get a current deduction; if you characterize it as a mortgage payment, you can't deduct it but you do get depreciation; and the rules conflict, because you can't characterize a given payment both ways. Cases interpreting these provisions show that a given payment was rent and not a mortgage payment or vice versa, and give the tax results of each conclusion. An example of a positive interaction is the combination of §121 and §1034, which can be used together to exclude part of the gain and defer part of the gain on the same transaction.

Some of the possible plan combinations work, some failures can be detected by the hierarchical planner, and some failures can be detected, if at all, only by the case-based planner.

A characteristic type of subgoal interaction in this domain is the interaction that cannot be detected without looking at cases, because of the rules' open texture. For example, if you rent out your house and take depreciation, in order to obtain the deduction, you must take the position that the house is being "held for the production of income." (§167). Later in the same

year, if you want to sell the house and use the proceeds to buy a residence for yourself, you will need to show that the house is your "principal residence." There is a conflict between these two claims: business and personal use of the same property; but the conflict is not apparent on the surface of the statute.

Our solution is to let the facts determine the combination. The same facts (or subsets of the same facts) are used to support all the deductions or exclusions claimed, and the facts can be used to mediate between them. If one strategy succeeds and a second fails, CHIRON later backtracks and tries the second on its own, and that may form the basis for an alternative plan even if the two couldn't be executed simultaneously.

4.7 Combining case-based and hierarchical planners

CHIRON's case-based and hierarchical planners are equal, interdependent reasoners. Each cooperates with the other. The hierarchical planner calls the case-based module for guidance in choosing adaptations and refining the open-textured predicates that remain in plans when the adaptations run out, and the case-based planner uses the hierarchical planner to help with indexing, controlling adaptations, and combining plans.

The most natural solution in a domain like tax law is to have two equal reasoners. Both rules and cases exist in the domain. The rules are not, as in many expert systems, an artifact created by the system designer. The natural structure of the tax domain is based on rules and cases, and following that structure makes the system easier to design and maintain.

We could translate all the cases into rules; but there are many rules for which each case could stand. We could translate all the rules into cases, but each rule corresponds to a set of cases. The set might be very large, even if it could be defined precisely; and given the open texture of the rules, specifying the corresponding set of cases would be difficult.

It is also natural to make the two modules interdependent. Both the rules and the cases are named. The cases refer to the rules they interpret, and books discussing the rules are annotated with references to the cases. And the rules and the cases complement each other. The rules organize cases into groups and partition the case base; the cases interpret the rules and suggest rules that might be considered.

Suppose we had chosen a purely rule-dominant design. Then the system would refine the plan as far as possible using rules, and invoke the case-based planner only when the rules run out. This is comparable to the algorithm used by Gardner's legal analysis system [Gardner, 1987]. As the rule base grows larger and larger, however, search control – controlling the

choice of plan decompositions – becomes an important issue. The case-based reasoner could suggest search paths for the hierarchical planner, if it were invoked earlier.

Suppose we had chosen a case-dominant design. Then the system would retrieve similar cases first, and apply rules second, if at all. But rules structure the case base. Cases are related to each other by the fact that they interpret the same rules. Rules also partition the case base. A case-dominant approach fails to take advantage of this structure.

We have chosen a tightly-coupled architecture that exploits our knowledge about the potential interactions of the case-based and hierarchical planners. CABARET also used two equal modules. In that system, however, all the control knowledge was isolated in a separate control module. For CABARET's implementation in the tax domain, its two modules were rule-based and case-based reasoners, but the system was designed to work with various types of modules, including case-based, rule-based, and model-based reasoners.

4.8 Summary

CHIRON is designed to be a lawyer's assistant. The hypothetical user of the system, a lawyer, begins by entering information about a taxpayer's goals and tax situation. Given this input, the system generates plans by which the taxpayer can reduce his or her taxes, annotated with citations to relevant cases and statutes, and arguments for and against the success of each strategy in the current situation. The system's task environment, in particular, the open texture of legal rules, the adversarial nature of the domain, and the fact that the hypothetical users of the system are experts, strongly influenced its design.

In order to reason about open-textured rules and cases, CHIRON combines hierarchical and case-based planners in a hybrid system. The hierarchical planner reasons with representations of open-textured rules; it also reasons with representations of facts. The case-based planner bridges the gap between the two, as legal cases bridge the gap between open-textured rules and facts.

CHIRON's case-based and hierarchical planners are equal, interdependent reasoners. Each cooperates with the other. The hierarchical planner calls the case-based module for guidance in choosing adaptations and refining the open-textured predicates that remain in plans when the adaptations run out, and the case-based planner uses the hierarchical planner to help with indexing, controlling adaptations, and combining plans.

Because law is an adversarial domain, it is useful to have a safe interpretation of the rules, and also to have some measure of how far you can deviate from that safe interpretation.

Accordingly, we associate a prototype "safe-harbor" plan with each of the hierarchical planner's strategies and use the previous cases as guidelines to show how far the prototype can be adapted.

CHIRON's adaptations are suggested by its cases. Some adaptations consist simply of adding a fact to the prototype; others involve varying the parameters of a given fact. Any fact in one of the previous cases can potentially be added to the prototype to make a new plan. Similarly, facts can be subtracted from the prototype, and parameters can be varied, as suggested by the cases. Adaptations relate the prototypes to the cases and the cases to each other. If you start with the prototype and apply the appropriate adaptations, you will obtain an actual case.

In other words, the prototype, cases, and adaptations define a space of possible plans. Within that space, CHIRON generates plans that are as similar to the prototype as possible. If a plan is weaker than any of the previous successful cases along some dimension, the system rejects it, unless there is at least a simple trend towards weakening cases along that dimension.

CHIRON's case-based planner uses a particularly detailed fact representation. The system's algorithms for indexing, adapting, comparing, and contrasting cases exploit this rich representation. Cases are indexed under all of their facts, both representations of the facts in the case report and abstractions and inferences from those facts. In addition, they are indexed under the types of transactions they involve (e.g., sale of personal property, gift of real property, etc.), and under the partial plan that they interpret. The current situation is compared and contrasted with previous cases using HYPO's algorithm, which examines the facts that the current situation shares with each previous case. And adaptations, as discussed above, are associated with each of the facts found in any of the previous cases.

Because the hypothetical users of the system are experts, CHIRON gives as much control as possible to the user. The system gives the user a set of plans to choose from, rather than just one. In addition to suggesting individual strategies, it informs the user which combinations of the suggested strategies have succeeded in previous cases. The user is allowed to suggest plans or combinations of plans for the system to work on. The system attempts to construct a plan combining all the strategies requested by the user, and repeats this process as long as the user continues to request combinations. Finally, each plan constructed by the system is accompanied by arguments for and against its success in the current situation, so that the user can assess its merits.

Chapter 5

Implementation

5.1 Introduction

In Chapters 3 and 4 we described CHIRON's design; in this chapter, we describe the implementation of that design. Section 5.2 is an overview of CHIRON's top-level architecture, including the case-guided search and control modules. The interaction between the hierarchical and case-based planners is discussed in Section 5.3, and the hierarchical and case-based planners are discussed in more detail in Sections 5.4 and 5.5.

5.2 Overview

CHIRON makes planning decisions on two levels. On the top level, it decides which tax-reduction strategies or combinations of strategies to consider. On the second level, it determines which of those strategies or combinations will work in the current situation, and constructs a concrete plan and supporting arguments for each.

At the top level, CHIRON is made up of several parts: the Classifier, the Strategy-Retriever, the Strategy-Processor, the Combination-Retriever, and the Combiner.

The **Classifier** and the **Strategy-Retriever** work together to guide the Strategy-Processor's search through the space of possible strategies, using a combination of rules and case-based reasoning.

First, the Classifier takes the input and applies rules to determine what types of transactions are involved. As discussed in Chapter 3, transactions are classified according to the type of rights and the type of property transferred: gifts, sales, rentals, or loans of money, tangible

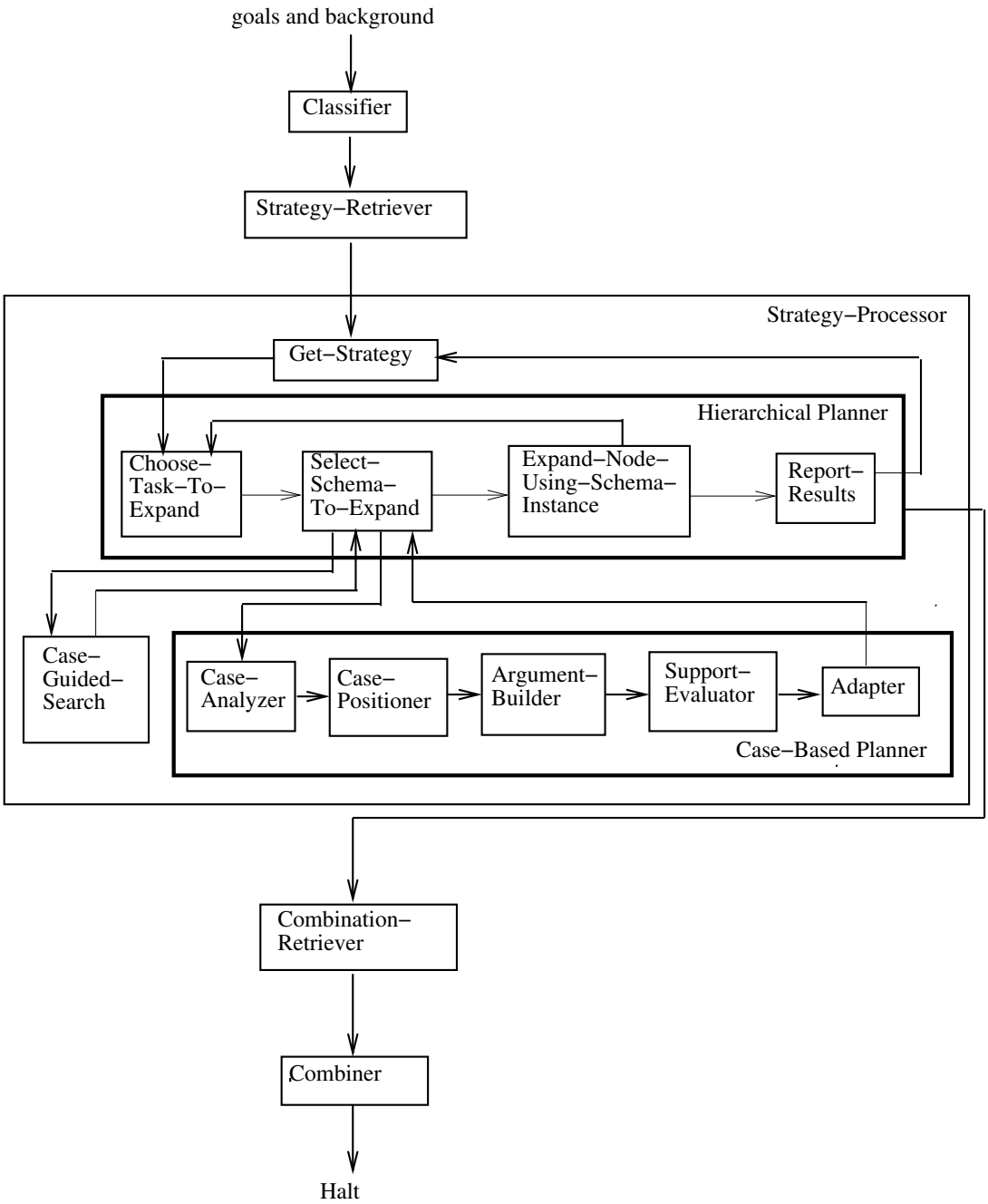

Figure 5.1: The Strategy-Processor in context

.

personal property, intangible personal property, or real estate, employment (a transfer of services in return for property), and volunteering. A diagram of the possible transfer types

can be found in Figure 3.5, and a partial diagram of the hierarchy of property types is given in Figure 3.4.

Second, the Strategy-Retriever takes the transactions, retrieves the past cases indexed under those transactions, and returns a list of the tax-reduction strategies used in those cases. These strategies are expressed in the form of a sequence of decomposition rules that, if applicable in the current situation, can be used by the hierarchical planner to transform the user's goals into one of the candidate partial plans in the hierarchical planner's repertoire. In this way, the cases guide the hierarchical planner's search by suggesting paths through the space of possible plans.

The **Strategy-Processor** takes the list of possible strategies returned by the Strategy-Retriever and processes each one. It calls the hierarchical planner, which cooperates with the case-based planner to determine which of the strategies can be used in the current fact situation, to construct plans for implementing the usable strategies, and to find supporting arguments for those plans. Its place in CHIRON's architecture is illustrated in Figure 5.1.

After processing each of the suggested strategies, CHIRON calls Report-Results to print out the results of its analysis for the user. If the strategy was rejected by the case-based planner, Report-Results prints out the weaknesses that caused the plan to be rejected and relevant case citations. Otherwise, Report-Results prints out the plan and citations to relevant provisions of the Internal Revenue Code and the most-on-point previous cases. It then gives HYPO-style arguments for and against the success of the plan, based on the information computed by the case-based planner (discussed in more detail below), and indicates the ways in which the plan is weaker than the prototype (if any) and the support provided by previous cases for each weakness.

The **Combination-Retriever** takes a list of tax-reduction strategies, retrieves the previous cases that executed any of those strategies, and returns a list of subsets of those strategies that have been used successfully in combination in past cases. This module looks only at the strategies and whether they succeeded in combination, without considering the facts of the current situation or comparing them to the facts of the previous cases. The information it provides for the user is helpful, but does not determine which combination the user should choose.

Finally, the **Combiner** obtains a request from the user for a combination of strategies. The user may choose one of the combinations suggested by the Combination-Retriever or some other combination, and may continue to request different combinations until he or she is satisfied. After each request, the Combiner calls the hierarchical and case-based planners to

construct, if possible, a plan and supporting arguments for the desired combination, and calls Report-Results to print out the results of this analysis for the user. Details of the Combiner are given in Figure 5.2.

5.3 The interaction of the hierarchical and case-based planners

The hierarchical and case-based planners work together to produce plans, either for a single tax-reduction strategy (when called by the Strategy-Processor), or for a combination of strategies requested by the user (when called by the Combiner).

The hierarchical planner starts with the user's goals (e.g., sell a house), adds a system goal to reduce the client's taxes, and refines these goals until all remaining steps are open-textured or primitive. The case-based planner then retrieves a prototype for the open-textured step (a strategy), determines whether there is sufficient support for using the strategy in the current situation, and if so, adapts the prototype as necessary and generates supporting arguments. If successful, the case-based planner returns an adapted prototype, which is converted into a schema that can be used by the hierarchical planner. The hierarchical planner refines its partial plan using the schema supplied by the case-based planner, and further refines any remaining nonprimitive actions. When this process is complete, the calling routine (either the strategy-processor or the combiner) calls report-results to print out the plan and supporting arguments, if the plan is being recommended, or if not, the reasons why it is being rejected.

When constructing a plan that combines more than one strategy, CHIRON constructs a plan for the first strategy and then attempts to add steps to achieve each additional strategy. Some kinds of interaction – duplicated steps and directly conflicting steps – can be detected by the hierarchical planner. It avoids duplication – for example, if selling your house is a step in two different plans, when combining those plans the hierarchical planner includes that step only once. And it rejects combinations that involve a direct conflict, for example if one plan requires that you sell your house and the second requires that you give it away.

With regard to combined plans, the case-based planner also provides some information. It determines whether the combination has been tried before, and if so, whether it succeeded in previous cases. If all the cases that tried a given combination before succeeded, that's a good sign. If all of them failed, that's not necessarily fatal. In any case, success or failure depends on the facts, and the facts of cases that go to court are usually borderline. Thus, if even the borderline cases win, the current situation is likely to fall within the acceptable limits. If the borderline cases lose, the taxpayer in the current situation may still win, if the facts of the

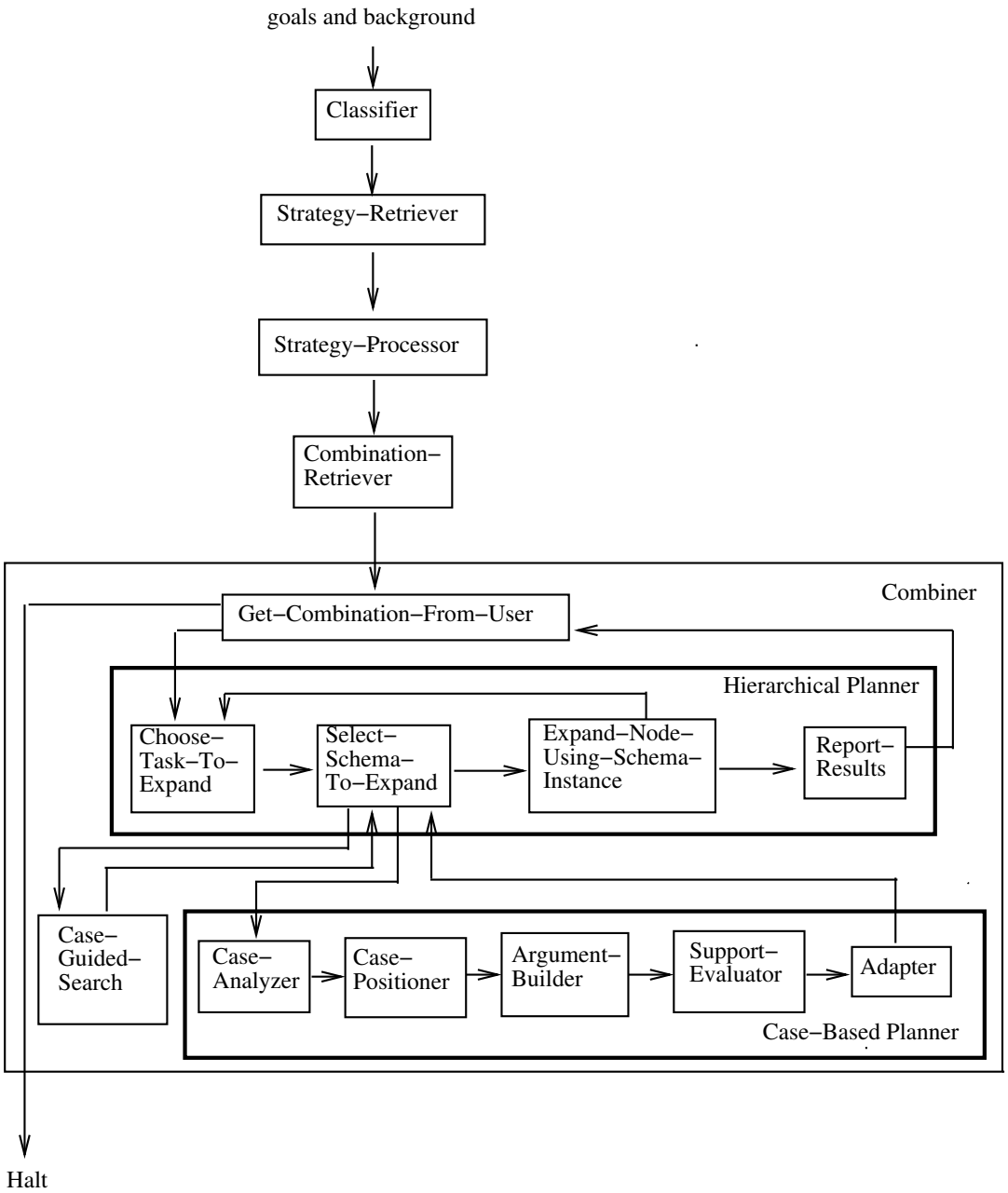

Figure 5.2: The Combiner's place in CHIRON's architecture.

current situation are more conservative.

If there is no obvious conflict, CHIRON constructs the combined plan and reports that there are no cases supporting the combination; the final decision whether to risk executing the plan is left to the user.

The hierarchical and case-based planners are discussed in more detail in the next two sections.

5.4 Hierarchical planner

CHIRON's hierarchical planner is adapted from a version of Nonlin implemented in Common Lisp at the University of Maryland [Ghosh *et al.*, 1991]. CHIRON retains the basic flow of control and architecture of the Maryland Nonlin, but stores most of its data, including the tasknet, in DUCK, a deductive retriever written at Yale [McDermott, 1985]. DUCK allows the user to store (or "premiss") and retrieve facts and supports forward and backward-chaining.

Using DUCK also provides us with a clean backtracking mechanism. DUCK allows the user to create independent but related local databases, known as "datapools." Instead of revising a variety of data structures, as the original Maryland Nonlin did, we simply establish a new datapool at each choice point and when backtracking, pop the most recent datapool off the stack.

In addition, we modified the Maryland Nonlin's design in two ways to incorporate the case-based planner: first, the case-based planner is called to help in the choice of decompositions; and second, the case-based planner is called to suggest decompositions for certain "open-textured" tasks.

With these modifications, the hierarchical planner's basic algorithm is:
While there are any non-open-textured tasks that can still be transformed:

1. choose a nonprimitive task;
2. get a list of possible decompositions for that task;
3. choose a decomposition (using suggestions from the case-based planner);
4. for each new subtask:
   (a) add it to the task network;
   (b) add ordering constraints as necessary;
   (c) add any temporal constraints specified in the decomposition;
   (d) project the subtask's effects; and
   (e) find any protection violations and solve them if possible.

The data structures used by the hierarchical planner are:

- *the tasknet*. The partially-ordered network of actions in the current plan, or partial plan. In the Maryland Nonlin, the tasknet was implemented as an array (called "allnodes.") In CHIRON, it is a set of assertions in the current datapool. In both cases, tasks have ids, types, parents, children, predecessors, successors, and marks. Thus, part of the tasknet might be represented as:

```
(task TSK244)
(parent TSK243 TSK244)
(parent TSK244 TSK263)
(pred TSK201 TSK244)
```

  where TSK201, TSK243, TSK244, and TSK263 are all task-ids, TSK244 is the child of TSK243, the parent of TSK263, and the successor of TSK201.

- *gost*. The conditions on each node and a list of contributors for each. In the Maryland Nonlin, this was implemented as a hash table, with conditions hashed on their patterns; in CHIRON, it is implemented as a set of assertions in the current datapool, like the tasknet.

- *tome*. The effects of each node. This was implemented in the Maryland Nonlin as a hashtable, with each effect indexed under its pattern, but in CHIRON, it is a set of assertions in the current datapool.

- *taskqueue*. The nodes that have not yet been expanded. In the Maryland Nonlin, this was implemented as a LIFO list of nodes. CHIRON does not need a separate data structure for the task queue; it simply retrieves all the goal, action, or open-textured tasks that have no children, from the current datapool.

- *context-list*. This is a stack of information about what to do after backtracking to each choice point. It is essentially the same as in the Maryland Nonlin.

- *schemas*. These are in the same form as Nonlin's schemas:

```
(defstruct  schema
   (id
    name
    todo
```

```
strip
size
conditions
effects
 vars)
```

except that for clarity we renamed the slot Nonlin called "expansion" and the Maryland Nonlin calls "strip," "steps." As with *gost* and *tome*, the Maryland Nonlin stored schemas in a hashtable, while we assert them in a datapool. Unlike *gost* and *tome*, the schemas are in an initial datapool that does not change as the program is run.

When called by the Strategy-Processor, the hierarchical planner attempts to construct a plan for each of the strategies suggested by the case-based planner one by one, backtracking between plans. When called by the Combiner, instead of backtracking, it constructs a combined plan by adding strategies one by one. In effect, the or-nodes in the plan space are changed to and-nodes.

5.5 Case-based planner

The case-based planner is called by the hierarchical planner to refine each tax-reduction strategy into a concrete plan. It compares the current situation to the facts of previous cases where a given strategy was executed, and if the current facts are within the limits established by previous successful cases, the case-based planner will retrieve a prototype plan for this strategy, adapt it as much as necessary to fit it to the current situation, and construct supporting arguments based on previous cases.

The basic algorithm used by the case-based planner in refining plans and constructing supporting arguments is:

1. get the facts about the current situation that are relevant to this strategy (i.e., those facts that have been mentioned in previous cases where this strategy was executed);
2. instantiate the prototype plan as it would be if it were executed by this strategy;
3. position the current situation in relation to previous similar cases (and the prototype);
4. select the cases that are most closely-related to the current situation;

5. construct arguments for and against the success of the plan, based on previous cases;

6. determine whether there is enough support for the plan by comparing its facts to those of previous cases; and

7. if there is enough support for the plan, adapt the prototype to fit the current situation.

The modules implementing these steps, the case-analyzer, the case-positioner, select-mopcs, the argument-builder, the support-evaluator, and the adapter, are discussed in more detail in the remainder of this section.

5.5.1 Case-analyzer

Task: To get the facts of the current situation that are relevant to the current strategy (by retrieving or computing them), instantiate the prototype plan, and store facts and prototype in a "plan record entry" to be used in constructing a plan and supporting arguments.
Input: a partial plan (stored in DUCK), the task currently being expanded, and the strategy.
Output: An entry in the plan-record for the given strategy. The information in the plan-record is used by the case-based-planner in comparing and contrasting the current plan with other cases. Thus, this module can be thought of as translating input from the hierarchical planner into the form used by the case-based planner.
Data structure:

- *plan-record*: includes an entry for each strategy. Each entry includes fields for:
  - today's date;
  - the current strategy;
  - relevant facts of the current situation; and
  - the prototype plan, instantiated for the current situation

  as well as slots for information computed by the case-positioner, support-evaluator, and argument-builder.

Algorithm:
The case-analyzer proceeds as follows:

- create a new plan-record entry for the given strategy;
- retrieve or compute the relevant facts and store them in the plan-record-entry;

- take the boilerplate for the prototype corresponding to the given strategy, fill it in with the appropriate values for the current situation, and store the resulting prototype plan in the plan-record-entry.

The case analyzer contains the following functions:

1. *make-plan-record-entry*: constructs a plan-record entry;
2. *get-fact-function*: returns the function that retrieves and/or computes the facts relevant to a given strategy;
3. *get-current-facts*: calls the fact-function for the given strategy (the function returned by get-fact-function) and combines the facts it returns with those generally relevant to all strategies, such as the type of return filed (or to be filed) by the taxpayer;
4. *construct-prototype*: gets the boilerplate prototype for this strategy and fills in the appropriate values from the current situation; and
5. *add-plan-record-entry*: stores the results in the plan-record entry.

One of the modules in HYPO is also called case-analyzer. The two modules are fundamentally similar: HYPO's case analyzer takes the input situation in the form of "legal case frames," determines which "factual predicates" apply, and uses the factual predicates to determine which "dimensions" apply. HYPO then compares the dimensions of the current fact situation with those of past cases to reason about a claim, such as trade secret violation or breach of contract; CHIRON compares the current plan's features and dimensions with those of past cases involving the same issues to reason about tax-reduction strategies, such as rollovers.

There are some differences, however. First, HYPO does not use prototypes, so there is no comparison of the current situation to a typical one. This is partly necessitated by the difference between their tasks: since HYPO is taking the role of a trial lawyer, analyzing the facts of a completed case, the case description can be assumed to be complete. In CHIRON, some of the facts may be given, but others will be open, so there is a use for the default values provided by a prototype.

Second, HYPO's case analyzer tests every situation for all possible features and dimensions, while CHIRON tests only for the features or dimensions associated with a particular strategy. Arguably, this is only a superficial distinction: HYPO can be seen as using cases to interpret a single open-textured statutory predicate, "trade secret violation." From this point of view, both systems are testing for the features and dimensions related to a particular strategy.

Finally, in CHIRON, unlike HYPO, the domain knowledge is stored explicitly in a separate domain-knowledge base.

5.5.2 Case-Positioner

Task: Retrieve all the cases in which the taxpayer executed or attempted to execute the given strategy and construct a subset lattice expressing the relationship of the current plan to those cases. Then retrieve all the cases in which one or more facts that match some fact in the current situation occur and construct a subset lattice expressing the relationship of the current plan to those cases. Store the first lattice in the strategy-lattice slot of the plan-record-entry for the current strategy; store the second lattice in the fact-lattice slot.
Input: The current strategy.
Output: A fact lattice and a strategy lattice in the plan-record-entry for the current strategy, to be used by the argument-builder and the support-evaluator.
Data structures:

- *lattices*. Used by Support-Evaluator, Select-Mopcs, and Argument-Builder.
- *nodes*. Structures with
  - a node-id;
  - a list of parents (nil for root);
  - a list of children;
  - a list of attributes (strategies or facts); and
  - a list of cases.
- *cases*. Structures with (among other things) a list of strategies and a list of dimensions and features (a representation of the case's statement of facts).
- *plan-record-entry*. Additional fields:
  - plan-record-entry-strategy-lattice: previous similar cases, stored according to the strategies they share with the current plan;
  - plan-record-entry-fact-lattice: previous similar cases, stored according to the facts they share with the current plan;
  - plan-record-entry-tp-dimensions: weaker cases that held for the taxpayer on the current strategy, indexed under the fact with respect to which they are weaker;

- plan-record-entry-govt-dimensions: stronger cases that held for the taxpayer on the current strategy, indexed under the fact with respect to which they are stronger;
- plan-record-entry-prototype-match: gives the mapping of the facts in the prototype to those of the current situation, including pairs of shared-facts, pairs of comparable-facts, facts that only appear in the current situation, and facts that only appear in the prototype.

- **domain-knowledge**: contains a list of fact structures. Each fact structure has a name (e.g., "House"), a pattern (e.g., (occurs (house ?place) ?t1 ?t2)), and a compare-function. The name is for use in output; the pattern is for convenience of the knowledge engineer, as a reminder of the form that facts of this type should take; and the compare-function determines for a given strategy whether two facts should be considered to match, or whether one is stronger than the other. The compare-functions return one of four possible values: :match, :stronger, :weaker, and :unknown.

  For example, with regard to the Section-121 exclusion, we might have the fact

  (occurs (age Green 60) (1990) (1990))

  That is, Green was sixty years old in 1990. If we find a fact in another case

  (occurs (age Brown 60) (1985) (1985))

  the two facts will be considered to match, because both individuals were the same age. (The times match because they each represent the tax year involved in their respective cases). If they were not the same age, for purposes of strategies where age is not a factor, the relative strengths of the two facts would be ":unknown." For purposes of the Section-121-exclusion strategy, however, since there is a minimum age for taking advantage of that strategy, the older the taxpayer the stronger the fact.

Algorithm:

The Case Positioner proceeds as follows:

1. Build the strategy lattice and store it in the plan-record-entry for the current strategy.
   (a) Make root-node. Fields:
      i. parents: nil;
      ii. children: nil;
      iii. index-list: the strategies being combined for the current plan; and

iv. case-list: nil.

(b) Retrieve all cases in which the taxpayer executed (or attempted to execute) one or more of these strategies.

(c) Construct a subset lattice for the strategies being combined, and store each case in the caselist for the node whose index-list is the strategies shared by that case with the current situation. Thus, the cases that share all of the current plan's strategies will be stored at the root node, and the more strategies a case shares with the current situation, the closer the node at which it is stored will be to the root.

(d) Store the lattice in the strategy-lattice slot of the plan-record-entry.

2. Build the fact lattice and store it in the plan-record-entry for the current strategy.

(a) Make the root-node. Fields:

i. parents: nil;

ii. children: nil;

iii. index-list: the relevant facts of the current situation, as computed by the case-analyzer; and

iv. case-list: nil.

(b) Retrieve all the cases where the taxpayer executed (or attempted to execute) the current strategy.

(c) For each such case, determine whether it involves any facts comparable to those in the current situation, and if so, compute a mapping between the facts of the previous case and those of the current situation. The results of the mapping is a list of pairs of facts that match, a list of pairs of facts that don't match but are comparable, a list of facts that appear only in the current situation, and a list of facts that appear only in the previous case.

(d) For each case, if it held for the government, store it in the govt-dimensions field of the plan-record-entry, indexed under each of the current facts for which it has a matching or comparable fact. If it held for the taxpayer, store it in the tp-dimensions field of the plan-record-entry, similarly indexed. Thus, for each fact in the current situation, the argument-builder will be able to retrieve all the previous cases involving matching or comparable facts.

(e) Construct a subset lattice for the facts of the current situation.

(f) Retrieve all the cases whose facts match any of the facts of the current situation.

(g) For each previous case:

i. map its facts to those of the current situation.

ii. Take the list of current facts that are matched by facts of the previous case, and find the node whose index-list is that list of facts.

iii. Store the case in the case-list for that node.

Thus, the cases that match all of the facts of the current situation will be stored at the root node, and the more facts a case matches, the closer the node at which it is stored will be to the root.

(h) Store the lattice in the fact-lattice slot of the plan-record-entry.

The Case Positioner uses the following functions:

1. *make-node*: creates nodes for the fact and strategy-lattices;

2. *retrieve-cases*: retrieves all the cases indexed under a given strategy;

3. *insert-lattice*: adds a new node to the fact or strategy lattice;

4. *compare-lists*: called by insert-lattice to compare two index-lists (either strategies or facts, depending on the lattice). Possible outputs are :same, :less, :greater, :overlap, and :disjoint.

5. *store-strategy-lattice*: stores strategy-lattice in the plan-record-entry for the given strategy;

6. *store-fact-lattice*: stores fact-lattice in the plan-record-entry for the given strategy;

7. *comparable-facts*: determines whether two facts are comparable with respect to a given strategy;

8. *match-facts*: takes a list of all possible pairs of facts and returns a mapping, that is, a list of matched facts and a list of comparable facts. The mapping returned is the one with the largest possible number of matched facts. If there are two mappings with equal numbers of matched facts, the one with the greater number of comparable facts is returned. Calls the compare-functions from *domain-knowledge*.

9. *store-dimension-facts*: takes each case and the associated mapping and indexes it in the tp-dimensions or the govt-dimensions field of the plan-record-entry for the current strategy, under all the facts of the current situation for which it has matching or comparable facts;

These lattices are essentially the same as HYPO's. In HYPO, as here, lattices are used as the basis for determining the similarity of past cases to the current situation. The root-node of the lattice contains cases that share all of the relevant indices with the current situation, and each of the other nodes contains cases that share some particular subset of those indices. The cases stored in nodes that are closer to the root share more indices with the current situation and are therefore more "similar" than those stored in more distant nodes.

We make two extensions to HYPO's use of lattices. First, each case is part of two lattices, rather than just one. In addition to using a lattice to show how the combination of features and dimensions in the current plan relates to those of previous cases, we construct a second lattice to show how the combination of strategies found in the current plan relates to strategies used in previous cases. In the strategy-lattice, the most "similar" cases are the ones that share the most strategies with the current plan, and those cases are stored in the nodes closest to the root of the lattice. In the fact-lattice, the most "similar" cases are those that match the most facts from the current situation, and they are stored in the nodes closest to the root of the fact-lattice. Because CHIRON uses a more detailed representation for its case-facts, the algorithms for comparing and contrasting the facts of the current situation with those of a previous case (in order to determine its place in the lattice) are necessarily more complex than HYPO's, but the function of the lattices is the same.

The second extension is that CHIRON has an additional use for its lattices: the Support-Evaluator uses the lattices to determine whether there is sufficient case support for combining a set of strategies and whether there is sufficient support for using a given strategy in the current situation.

5.5.3 Select-Mopcs

Task: Find the most-on-point cases with regard to the facts of the current situation ("mopcs," that is, the cases that share the most indices with the current situation). Similarly, find the mopcs with regard to the strategies used by the current plan. (These two tasks take different input lattices, but the algorithm is the same.)
Input: lattice.

Output: list containing two lists of mopcs, one list of cases where the taxpayer won, and one list of cases where the government won.

This module takes a case-lattice (based on either fact-indices or strategy-indices) and finds the most-on-point previous cases, or mopcs. When is a case more on-point than another? In HYPO, case1 is more on point with the current situation than case2 if the dimensions the current situation shares with case1 are a superset of those shared with case2, or, in terms of the implementation, if case1 is closer to the current situation than case2 along the same branch of the case lattice. In the abstract, we adopt the same definition, but since we include more kinds of facts in our fact-lattice, in practice, the result is slightly different.

In general, cases can be compared at various levels of abstraction. The more specific the level two cases match at, the better the match; the more abstract the level they match at, the weaker the match. Conversely, if your goal is to distinguish two cases, the more specific the level two cases match at, the weaker the difference; and the more abstract the level they match at, the stronger the difference.

We implement this relationship between specific and abstract matches by including both facts and abstractions from those facts (including features and dimensions) in the lattice. HYPO's lattices, by contrast, included only dimensions. For example, the case-facts of *Trisko v. Commissioner*, one of the cases in CHIRON's case base, include the fact that the taxpayer was employed by the Foreign Service, that he was employed by the government, and that his employer was tax-exempt.

Two cases that share a concrete fact will also share the abstractions from that fact; but two cases that share an abstraction may not share the same specific facts. For example, a plan involving a taxpayer employed by the Foreign Service would match *Trisko* on three facts; a plan involving a taxpayer who is employed by IBM would match only on one. The plan involving a Foreign Service employee is a closer match; and this is reflected in the fact that it shares more facts with the earlier case than the plan involving the IBM employee and therefore would be stored in a node closer to the root of the fact-lattice.

Another perspective on the same question is that each pro-taxpayer case can support a number of different plans, at different levels of abstraction. That is, each case supports (and generates)

1. plans obtained by instantiating the variablization of its concrete facts; and

2. abstractions from those plans.

More abstract plans are more weakly supported by any given case, but may draw support from

more than one case. At the most abstract level, a strategy is supported by all the pro-taxpayer cases using that strategy. The most specific plans are supported by exactly one case, but strongly supported. Support and adaptation are related: if a case supports part (or all) of a plan, then the case can be adapted to produce the part of the plan that it supports.

Some useful definitions that we borrow from HYPO:

1. Best support: untrumped, undistinguishable case.
2. Some support: untrumped but distinguishable case.
3. Minimal support: a citable case.
4. Trumped case: there exists another case which shares a superset of the dimensions the trumped case shares with the current plan, has the opposite result, and is citable.
5. Distinguishable case: a case that has more pro-side dimensions, fewer pro-opponent dimensions, stronger values for shared dimensions, or is not citable.
6. Citable case for a side: a case that shares at least one dimension with the current plan and holds for that side.
7. Citable in the first instance: a case that is citable and shares at least one pro-side dimension.
8. Citable in rebuttal: a case that is citable and shares at least one pro-opponent dimension.

5.5.4 Argument-Builder

Task: Generate basis for arguments for and against the success of the plan currently being constructed (to be used later by Report-Results).
Input: current strategy and plan-record.
Output: argument record(s).
Data Structures:

- *argument-records*. Used by Report-Results when printing out arguments for and against the success of the current plan. Structures with the following fields:
  - case: a pointer to the mopc on which the argument is based.
  - type: :fact or :strategy, depending on the type of lattice.

- strategy: the current strategy.
- shared-indices: all the indices shared by the current situation and the mopc. In the case of an argument based on a strategy-lattice, this is a list of strategies. In the case of an argument based on a fact-lattice, this is a list of fact-pairs, where the first element of each pair is a fact from the current situation and the second element is the matching fact from the mopc.
- only-in-cfs: indices (facts or strategies) that appear only in the current situation.
- only-in-mopc: indices (facts or strategies) that appear only in the mopc.
- tp-strengths: for fact-lattices, this includes the differences that make the current situation stronger for the taxpayer than the mopc. These differences can include facts that appear only in the current situation, facts that appear only in the mopc, or comparable facts that appear in both the current situation and the mopc, where the current situation's fact is stronger than the mopc's.
- govt-strengths: for fact-lattices, this includes the differences that make the current situation stronger for the government that the mopc.
- neutral-facts: for fact-lattices, this includes the facts (shared, only-in-mopc, or only in the current situation) that are neither in tp-strengths nor in govt-strengths.
- boundary-tpcases: weaker cases where the taxpayer won, sorted into groups according to the fact with regard to which they are weaker;
- boundary-gcases: stronger cases where the government won, sorted into groups according to the fact with regard to which they are stronger;
- superset-tpcases: cases that held for the taxpayer and that share a set of indices with the current situation that is a superset of the indices shared by the mopc and the current situation;
- superset-gcases: similarly for cases that held for the government.
- overlap-tpcases: cases that held for the taxpayer and that share a set of indices with the current situation that partially overlaps with the set of indices shared by the mopc and the current situation.
- overlap-gcases: similarly for cases that held for the government.

- *plan-record-entry*. Additional fields:

- plan-record-entry-strategy-mopcs: a list of most-on-point cases retrieved from the strategy-lattice by select-mopcs;
- plan-record-entry-strategy-argrecs: contains an argument-record for each of the strategy-mopcs, sorted into two lists according to whether the case held for the taxpayer or for the government;
- plan-record-entry-fact-mopcs: a list of most-on-point cases retrieved from the fact-lattice by select-mopcs;
- plan-record-entry-fact-argrecs: contains an argument-record for each of the fact-mopcs, sorted into two lists according to whether the case held for the taxpayer or for the government.

Algorithm:

The Argument-Builder proceeds as follows:

1. Find the most-on-point (most similar) cases using the strategy-lattice.
2. Construct an argument-record for each most-on-point case ("mopc") with regard to the current strategy combination.
   (a) Compare the mopc's and the current plan's strategies, and compute the shared-indices, only-in-mopc, and only-in-cfs slots of the argument-record.
   (b) For each of the other mopcs:
      i. compare the indices the other mopc shares with the current situation to the indices this mopc shares with the current situation;
      ii. if the other mopc's shared indices are a proper superset of the current mopc's, add the other mopc to superset-tpcases or superset-gcases, depending on whether the case held for the taxpayer or for the government;
      iii. if the other mopc's shared indices overlap this mopc's shared indices but are not a proper superset, add the other mopc to overlap-tpcases or overlap-gcases, depending on whether the case held for the taxpayer or for the government.
3. Store the resulting argument-record in plan-record-entry-strategy-argrecs.
4. Find the most-on-point (most similar) cases using the fact-lattice.
5. Construct an argument-record for each fact mopc.

(a) Let the shared-indices be the union of the matching and the comparable facts from the mapping of the current situation to the previous case computed by the case-positioner.

(b) Set only-in-cfs and only-in-mopc to be the only-in-cfs and only-in-mopc facts as computed by the case-positioner.

(c) Set the neutral-facts to be the shared-facts as computed by the case-positioner (These are "matches," where neither fact is stronger than the other).

(d) Set the tp-strengths to be the facts that make the current situation stronger for the taxpayer:

   i. the comparable-facts (as computed by the case-positioner) where the current-situation's fact is stronger;

   ii. facts supporting the taxpayer's position that appear only in the current situation, not in the mopc;

   iii. facts supporting the government's position that appear only in the mopc, not the current situation.

(e) Set the govt-strengths to be the facts that make the current situation stronger for the government than the mopc:

   i. the comparable-facts (as computed by the case-positioner) where the current-situation's fact is stronger for the government;

   ii. facts supporting the government's position that appear only in the current situation, not in the mopc;

   iii. facts supporting the taxpayer's position that appear only in the mopc, not in the current situation.

(f) For each of the other mopcs:

   i. compare the indices the other mopc shares with the current situation to the indices this mopc shares with the current situation;

   ii. if the other mopc's shared indices are a proper superset of the current mopc's, add the other mopc to superset-tpcases or superset-gcases, depending on whether the case held for the taxpayer or for the government;

   iii. if the other mopc's shared indices overlap this mopc's shared indices but are not a proper superset, add the other mopc to overlap-tpcases or overlap-gcases, depending on whether the case held for the taxpayer or for the government.

iv. For each of the shared-indices (i.e., for each of the facts of the current situation that are shared with or comparable to the mopc), find weaker cases where the taxpayer won and store them in boundary-tpcases, and find stronger cases where the government won and store them in boundary-gcases.

6. Store the resulting argument-record in plan-record-entry-fact-argrecs.

7. Construct a fact-argument-record for the prototype, regardless of whether it is a mopc, and store it in plan-record-entry-prototype-argrec.

When CHIRON prints out its results for the user, it uses these argument-records as the basis for arguments. These arguments and the methods for generating them are similar to HYPO's – based on identifying features and dimensions that the current plan shares (or does not share) with previous most-on-point cases. One of HYPO's major contributions was a computational model of legal argumentation; and we adopt that model here. We extend the model slightly by constructing arguments about proposed strategy combinations, as well as combinations of facts.

A more significant extension is the application of HYPO-style arguments to planning. Both HYPO and CABARET, the systems where this kind of argumentation has been performed before, are legal analysis systems. That is, they take the role of a trial lawyer, examining the facts of a past situation and building arguments for and against a particular legal result in that situation. CHIRON is a planner, considering facts some of which may be past and some are still in the future.

The construction of supporting arguments for a plan helps to address the question of validation. The system has no learning component; there is no mechanism for executing or simulating execution of the plan. Instead, CHIRON provides arguments for and against the success of a plan. These arguments, like the explanation facility in an expert system, help the user to evaluate the usefulness of the system's output.

5.5.5 Support-Evaluator

Task: Determine whether there is enough support to justify suggesting this plan.
Input: case lattices.
Output: yes or no; and if no, the reason for failure.
Data Structure:

- *plan-record-entry*. Additional field:

- plan-record-entry-failure. Stores a list of reasons for failure (if any).

Algorithm:
The Support-Evaluator proceeds as follows:

1. Determine whether there is sufficient support for the current strategy based on the facts of the current situation.
   (a) Get the most-on-point cases from the fact-lattice.
   (b) If there are cases supporting the government's position and no cases supporting the government's position, save the government cases in plan-record-entry-failure.
2. Determine whether there is sufficient support for the current strategy on each dimension of the current situation.
   (a) Retrieve the facts with respect to which the current situation is weaker than the prototype from the prototype argument-record (stored in the plan-record-entry).
   (b) For each of those facts, look for supporting cases: previous cases where the taxpayer won with an even weaker position than the current situation's.
   (c) If none, look for a trend: two previous cases where the taxpayer won with increasingly weak positions along the given dimension (although stronger than the current situation's position).
   (d) If neither of these is found, save the current fact in plan-record-entry-failure.

This module looks at the mopcs returned by Select-Mopcs to determine whether there is sufficient support for a plan. The hierarchical planner can weed out "easy cases" of bad plans using very general rules. For example, it would reject a plan that involved both giving away and selling a piece of property, or selling the same piece of property twice. By the time a plan is handed over to the case-based planner, then, it has passed this initial threshold and is not an obvious failure. The case-based planner must make a more subtle distinction, and it does not have rules to fall back on; it knows only that certain plans have succeeded or failed under certain circumstances.

The more case support is required, the more plans will be rejected, and the more control will be taken away from the user. The less case support is required, the more dubious plans will be suggested. We make the following (fairly arbitrary) decisions about what support to require:

1. *facts*. There must be at least one most-on-point case on the facts that held for the taxpayer.

2. *dimensions*. For each fact with respect to which the current situation is weaker than the prototype, there must be either (1) a weaker case that held for the taxpayer or (2) two cases case1 and case2 such that

   (a) case1 is weaker than the prototype;

   (b) case2 is weaker than case1;

   (c) the current situation is weaker than case2;

   (d) both case1 and case2 held for the taxpayer; and

   (e) the date of case1 is earlier than case2.

3. *strategy combinations*. Novel strategy combinations may succeed, and strategy combinations that have failed in the past may succeed given more conservative facts. As a result, the Support-Evaluator does not reject combinations simply because there is little or no support on the strategy level. Instead, CHIRON informs the user of the support available for the given strategy combination and leaves the decision up to the user (see discussion of Report-Results, above).

In addition, in case the user wishes to see all plans that pass the minimal test of the hierarchical planner, CHIRON provides the option of turning off the Support-Evaluator, so these factors are not taken into account.

5.5.6 Adapter

Task: Take prototype, analysis of current plan, and past cases, and produce a plan to be returned to the hierarchical planner.
Input: the current strategy and plan-record-entry.
Output: a list of facts.
Data Structures: no data structures not used by earlier modules.
Algorithm:
The Adapter proceeds as follows:

1. Retrieve the facts of the prototype plan from the plan-record-entry for the current strategy.

2. Retrieve the mapping of the current facts to the prototype facts from the prototype argument-record, also stored in the plan-record-entry for the current strategy.

3. For each pair of facts in the mapping, substitute the current fact for the prototype fact in the list of prototype-facts.

4. Return the resulting list of facts.

The facts returned by the Adapter are then transformed by an interface function, convert-list-to-schema-instances, into schemas that can be used by the hierarchical planner.

The Adapter only makes the minimal changes necessary to fit the prototype to the current situation; using the mappings (matched-facts and comparable-facts) computed by the Case-Positioner, the Adapter substitutes each current fact for the one it is mapped to in the prototype. CHIRON could adopt a more aggressive strategy by changing the facts of the current situation to match weaker pro-taxpayer cases, where the weakness would be beneficial to the taxpayer. Report-Results does point these opportunities out to the user, but we have chosen to implement a more conservative strategy, which is typical of tax planners.

Alternatively, CHIRON could adopt an even more conservative strategy by changing the facts of the current situation to match those of the prototype. For example, if you are not living in a house, CHIRON could suggest that you move in, then execute the rollover strategy. This change might even make it possible to execute plans that would be rejected by the current version of CHIRON for lack of support. Moving into a house and making it your residence would be equally effective if you had been gone for one year (in which case there would be support for a rollover even without moving back) or ten years (in which case the rollover strategy would currently be rejected). Many facts cannot be changed, however. If you sold your house five years ago and didn't buy a second one, you can't unsell it and start again. In addition, changing individual facts requires knowledge about the effect of small changes on the whole plan. It might be, for example, that in order to move into the house, you'd have to move six hundred miles away from your current job, which would mean getting a new job. Case-based planning is based in part on the assumption that you don't have that kind of information about the interactions of parts of a plan; all you have to work with are entire plans, as executed in the past. Instead of making the changes, then, CHIRON reports the plan's weaknesses to the user (whether or not they are grounds for rejecting the plan) and allows the user to decide.

Chapter 6

Extended Examples

CHIRON solves a cluster of problems having to do with buying, selling, renting, and owning residential housing. Many of the provisions of the United States Internal Revenue Code affect these transactions, directly or indirectly. In addition to the general provisions governing sale of capital assets, payment of interest on loans, and so forth, there are special benefits for residential housing. For example, under certain circumstances, the gain from the sale of an individual's principal residence can be deferred or completely excluded from taxation. §§121, 1034.

CHIRON's examples have been chosen in part because they are simple; most tax planners would agree on at least the obvious solutions. In addition, they illustrate problems such as timing and satisfaction of open-textured rules that are typical of the domain. Finally, since they all involve transfers of residential housing, the commonsense knowledge that must be formalized to handle them involves the same cluster of concepts.

In this chapter, we describe how CHIRON handles several of these examples. First, we discuss some straightforward examples using the rollover strategy: one where it is recommended, one where it is rejected, and one where it is recommended despite a weakness in the taxpayer's situation. Second, we give an example of where the system does a particularly good job distinguishing a previous case, and finally, one where it fails to recommend a plan that should have succeeded. These examples are excerpts from transcripts, edited to fit within the margins of a text document. Following the usual convention, system output will be given in `Courier`, and discussion of that output is given in a Roman font.

6.1 The rollover strategy is recommended

First, consider a straightforward example: the taxpayer, whose name is Greenlee, wants to sell a house. She has owned the house, identified by the token 32-Eleventh-Street, since October 30, 1958, and has occupied the house during that entire period. She is now forty-two years old. CHIRON takes as input the internal representation of these facts, as follows:

```
(occurs (individual-return return1) (1994) (1994))
(occurs (taxpayer return1 Greenlee) (1994) (1994))
(occurs (age Greenlee 42) (1994) (1994))
(goal (occurs (selling selling157) ft155 ft156)
   Greenlee (9 20 1994) (9 20 1994))
(goal (occurs (seller selling157 Greenlee) ft155 ft156)
   Greenlee (9 20 1994) (9 20 1994))
(goal (occurs (object selling157 32-Eleventh-Street)
   ft155 ft156) Greenlee (9 20 1994) (9 20 1994))
(occurs (house 32-Eleventh-Street)
   (0 0 0) (12 31 99999))
(occurs (physically-occupy Greenlee 32-Eleventh-Street)
   (10 30 1958) (9 20 1994))
(occurs (owns Greenlee 32-Eleventh-Street)
   (10 30 1958) (9 20 1994))
```

Next CHIRON asks:

```
>  Evaluate case support for plans? Enter t or nil.
t
```

The user has the option of seeing plans, whether or not they have adequate case support. We choose to let the system filter out the plans without case support; if there are any, it will inform us and explain its reasons.

Next, CHIRON determines that the current situation involves one type of transaction: a sale of real property in return for some other unspecified type of property.

```
The current situation involves the following
transaction types:
  (:SALE :REAL-PROPERTY :PROPERTY)
```

CHIRON then retrieves the cases in its case base that also involved a sale of real property. There are four:

```
Cases retrieved:
  "Welch v. Commissioner, T.C.M. (P-H) 79,010 (1979)".
  "Trisko v. Commissioner, 29 T.C. 515 (1957)".
  "Sayre v. United States, 163 F.Supp. 495 (S.D.W.Va. 1958)".
  "Hughston v. Commissioner, T.C.M. (P-H) 50,188 (1950)".

Strategies to consider:
  :EXCLUSION121
  :ROLLOVER
  :LIKE-KIND-EXCHANGE
```

These cases suggest three strategies: a §121 exclusion, which permits a taxpayer who is over 55 years old to sell his or her principal residence and exclude part of the gain from tax, if he or she meets certain tests; a rollover, where the taxpayer sells one principal residence and buys another; and a like-kind exchange, where the taxpayer exchanges one piece of property for another similar one.

CHIRON now attempts to construct a plan for each of these strategies in the current situation, starting with the like-kind exchange. It rejects this strategy and explains why:

```
**************************************************
                  RESULTS
**************************************************
The strategy :LIKE-KIND-EXCHANGE has been rejected for the
following reasons:

  (1) The strategy :LIKE-KIND-EXCHANGE is unlikely to succeed
because there is no support for the plan along the dimension
DURATION-OF-OCCUPANCY.
In the current situation:
   (OCCURS (DURATION-OF-OCCUPANCY GREENLEE 32-ELEVENTH-STREET
                (35 10 20)) (1 1 0) FT155).

**************************************************
```

The property involved in a like-kind exchange must be "held for productive use in a trade or business or for investment...." §1031. Personal use of of the property weakens the taxpayer's case, and there are no cases in CHIRON's case base with this weakness, so the system rejects the plan.

The next strategy, a rollover, is more promising. CHIRON constructs a plan and reports it to the user, starting with the strategy name and the relevant sections from the Internal Revenue Code.

```
**************************************************
                  RESULTS
**************************************************
The strategy :ROLLOVER is suggested.
Code sections involved in plan: (1034).
```

Then the system spells out the plan:

```
******************************************
             PLAN
******************************************

  ((OCCURS (ABSENCE-BEFORE-SALE GREENLEE SELLING157 (0 0 0))
            FT155 FT155))
  ((OCCURS (NUMBER-OF-RESIDENCES GREENLEE 1) FT155 FT155))
  ((OCCURS (DURATION-OF-OCCUPANCY GREENLEE 32-ELEVENTH-STREET
            (35 10 20)) (1 1 0) FT155))
  ((OCCURS (BUYER SELLING613 GREENLEE) FT611 FT612))
  ((OCCURS (PHYSICALLY-OCCUPY GREENLEE UNKNOWN-HOUSE610)
            FT608 FT609))
  ((OCCURS (SALE-TO-PURCHASE-TIME SELLING157 SELLING613
            (0 0 0)) FT155 FT611))
  ((OCCURS (SALE-TO-OCCUPANCY-TIME SELLING157 SELLING613
            (0 0 0)) FT155 (12 31 99999)))
  ((OCCURS (SELLING SELLING613) FT611 FT612))
  ((OCCURS (OBJECT SELLING613 UNKNOWN-HOUSE610) FT611 FT612))
  ((OCCURS (OBJECT SELLING157 32-ELEVENTH-STREET)
```

```
            FT155 FT156))
  ((OCCURS (SELLER SELLING157 GREENLEE) FT155 FT156))
  ((OCCURS (SELLING SELLING157) FT155 FT156))
```

The plan includes primitive actions and states that should be true in the future. The taxpayer is advised to live in the house until the date of sale, sell it and buy another on the same date, and occupy the new house immediately. In addition, the taxpayer should have only one residence at the time of sale. This is a very conservative plan, but it can be executed by the taxpayer, given the simple facts of our example (from a tax point of view, at least; the realities of the housing market may make selling more difficult).

Next, the system prints out arguments for and against the success of the plan, starting with the facts of the case (as input or computed by the system):

```
*******************************************

   Arguments Supporting the Strategy :ROLLOVER

*******************************************

On the question of executing the strategy :ROLLOVER
 given the facts:
(OCCURS (INDIVIDUAL-RETURN RETURN1) (1994) (1994))
(OCCURS (TAXPAYER RETURN1 GREENLEE) (1994) (1994))
(OCCURS (NUMBER-OF-RESIDENCES GREENLEE 1) FT155 FT155)
(OCCURS (ABSENCE-BEFORE-SALE GREENLEE SELLING157 (0 0 0))
          FT155 FT155)
(OCCURS (DURATION-OF-OCCUPANCY GREENLEE 32-ELEVENTH-STREET
          (35 10 20)) (1 1 0) FT155)
(OCCURS (SALE-TO-PURCHASE-TIME SELLING157 SELLING613 (0 0 0))
          FT155 FT611)
(OCCURS (SALE-TO-OCCUPANCY-TIME SELLING157 SELLING613 (0 0 0))
          FT155 (12 31 99999))
(OCCURS (OBJECT SELLING157 32-ELEVENTH-STREET) FT155 FT156)
(OCCURS (SELLER SELLING157 GREENLEE) FT155 FT156)
(OCCURS (SELLING SELLING157) FT155 FT156)
```

```
(OCCURS (PHYSICALLY-OCCUPY GREENLEE 32-ELEVENTH-STREET)
           (10 30 1958) (9 20 1994))
(OCCURS (OWNS GREENLEE 32-ELEVENTH-STREET)
           (10 30 1958) (9 20 1994))
(OCCURS (REAL-PROPERTY 32-ELEVENTH-STREET)
           (1 1 0) (12 31 99999))
(OCCURS (HOUSE 32-ELEVENTH-STREET) (1 1 0) (12 31 99999))
(OCCURS (OBJECT SELLING613 UNKNOWN-HOUSE610) FT611 FT612)
(OCCURS (REAL-PROPERTY UNKNOWN-HOUSE610)
           (1 1 0) (12 31 99999))
(OCCURS (HOUSE UNKNOWN-HOUSE610) (1 1 0) (12 31 99999))
(OCCURS (BUYER SELLING613 GREENLEE) FT611 FT612)
(OCCURS (SELLING SELLING613) FT611 FT612)
(OCCURS (PHYSICALLY-OCCUPY GREENLEE UNKNOWN-HOUSE610)
            FT608 FT609)

the taxpayer should succeed.
```

Like HYPO, CHIRON generates three-ply arguments; in other words, its arguments consist of a point, response, and (if possible), rebuttal. There is one of these three-ply arguments for each most-on-point case, whether it held for the government or for the taxpayer. In this case, which raises no real issues, there are no most-on point cases holding for the government, and in fact the prototype is the most-on-point case for the taxpayer.

```
The taxpayer can cite the following cases
for which there are no more-on-point counter-examples:

"prototype: rollover"
```

The system starts with an argument for the taxpayer, based on the prototype. It argues that the taxpayer should succeed because of the facts the current situation shares with the prototype, and it prints out the facts they have in common in matching pairs.

First, in the current situation, as in the prototype, the taxpayer buys and occupies a house.

```
==> POINT for TAXPAYER:
```

```
The TAXPAYER should succeed because the TAXPAYER succeeded in
"the rollover prototype", which shares the following FACTS
with the current plan:

Current situation:
  (OCCURS (PHYSICALLY-OCCUPY GREENLEE UNKNOWN-HOUSE610)
          FT608 FT609)
  "the rollover prototype":
  (OCCURS (PHYSICALLY-OCCUPY GREENLEE UNKNOWN-HOUSE610)
          UT623 UT625)
Current situation:
  (OCCURS (SELLING SELLING613) FT611 FT612)
  "the rollover prototype":
  (OCCURS (SELLING SELLING613) UT622 UT623)
Current situation:
  (OCCURS (BUYER SELLING613 GREENLEE) FT611 FT612)
  "the rollover prototype":
  (OCCURS (BUYER SELLING613 GREENLEE) UT622 UT623)
Current situation:
  (OCCURS (HOUSE UNKNOWN-HOUSE610) (1 1 0) (12 31 99999))
  "the rollover prototype":
  (OCCURS (HOUSE UNKNOWN-HOUSE610) (1 1 0) (12 31 99999))
Current situation:
  (OCCURS (REAL-PROPERTY UNKNOWN-HOUSE610)
          (1 1 0) (12 31 99999))
  "the rollover prototype":
  (OCCURS (REAL-PROPERTY UNKNOWN-HOUSE610)
          (1 1 0) (12 31 99999))
Current situation:
  (OCCURS (OBJECT SELLING613 UNKNOWN-HOUSE610) FT611 FT612)
  "the rollover prototype":
  (OCCURS (OBJECT SELLING613 UNKNOWN-HOUSE610) UT622 UT623)
```

In both the current situation and the prototype, the taxpayer sells a house that she had been occupying until the date of sale.

```
Current situation:
  (OCCURS (HOUSE 32-ELEVENTH-STREET) (1 1 0) (12 31 99999))
  "the rollover prototype":
  (OCCURS (HOUSE 32-ELEVENTH-STREET) (1 1 0) (12 31 99999))
Current situation:
  (OCCURS (REAL-PROPERTY 32-ELEVENTH-STREET)
          (1 1 0) (12 31 99999))
  "the rollover prototype":
  (OCCURS (REAL-PROPERTY 32-ELEVENTH-STREET)
          (1 1 0) (12 31 99999))
Current situation: (OCCURS (OWNS GREENLEE 32-ELEVENTH-STREET)
          (10 30 1958) (9 20 1994))
  "the rollover prototype":
  (OCCURS (OWNS GREENLEE 32-ELEVENTH-STREET) UT624 UT620)
Current situation:
  (OCCURS (PHYSICALLY-OCCUPY GREENLEE 32-ELEVENTH-STREET)
          (10 30 1958) (9 20 1994))
  "the rollover prototype":
  (OCCURS (PHYSICALLY-OCCUPY GREENLEE 32-ELEVENTH-STREET)
          UT624 UT620)
Current situation:
  (OCCURS (SELLING SELLING157) FT155 FT156)
  "the rollover prototype":
  (OCCURS (SELLING SELLING157) UT620 UT621)
Current situation:
  (OCCURS (SELLER SELLING157 GREENLEE) FT155 FT156)
  "the rollover prototype":
  (OCCURS (SELLER SELLING157 GREENLEE) UT620 UT621)
Current situation:
  (OCCURS (ABSENCE-BEFORE-SALE GREENLEE SELLING157 (0 0 0))
          FT155 FT155)
```

```
  "the rollover prototype":
  (OCCURS (ABSENCE-BEFORE-SALE GREENLEE SELLING157 (0 0 0))
          UT620 UT620)
Current situation:
  (OCCURS (OBJECT SELLING157 32-ELEVENTH-STREET) FT155 FT156)
  "the rollover prototype":
  (OCCURS (OBJECT SELLING157 32-ELEVENTH-STREET) UT620 UT621)
```

In both the current situation and the prototype, the taxpayer has one residence at the time of sale and files an individual tax return.

```
Current situation:
  (OCCURS (NUMBER-OF-RESIDENCES GREENLEE 1) FT155 FT155)
  "the rollover prototype":
  (OCCURS (NUMBER-OF-RESIDENCES GREENLEE 1) UT620 UT620)
Current situation:
  (OCCURS (TAXPAYER RETURN1 GREENLEE) (1994) (1994))
  "the rollover prototype":
  (OCCURS (TAXPAYER RETURN1 GREENLEE) (1994) (1994))
Current situation:
  (OCCURS (INDIVIDUAL-RETURN RETURN1) (1994) (1994))
  "the rollover prototype":
  (OCCURS (INDIVIDUAL-RETURN RETURN1) (1994) (1994))
```

Finally, both the prototype and the current situation include information about how long the taxpayer occupied her old residence and how much time elapsed between the sale of the old residence and the purchase and occupancy of the new one.

```
Current situation:
  (OCCURS (DURATION-OF-OCCUPANCY GREENLEE 32-ELEVENTH-STREET
          (35 10 20)) (1 1 0) FT155)
  "the rollover prototype":
  (OCCURS (DURATION-OF-OCCUPANCY GREENLEE 32-ELEVENTH-STREET
          (0 0 1)) UT620 UT620)
Current situation:
```

```
  (OCCURS (SALE-TO-OCCUPANCY-TIME SELLING157 SELLING613
          (0 0 0)) FT155 (12 31 99999))
  "the rollover prototype":
  (OCCURS (SALE-TO-OCCUPANCY-TIME SELLING157 SELLING613
          (2 0 0)) UT620 UT624)
Current situation:
  (OCCURS (SALE-TO-PURCHASE-TIME SELLING157 SELLING613
          (0 0 0)) FT155 FT611)
  "the rollover prototype":
  (OCCURS (SALE-TO-PURCHASE-TIME SELLING157 SELLING613
          (2 0 0)) UT620 UT622)

CITE: "prototype: rollover"
```

On these dimensions, the suggested plan is stronger than the prototype: the prototype requires only a minimal residence in the house before the date of sale, and purchase and occupancy of the second residence two years after the sale of the first; in the current situation, the taxpayer has been living in the house for over 35 years, and the system conservatively suggests buying the new residence the same day as the sale of the old one and occupying it immediately. For purposes of this part of the argument, these facts are considered to match; their relative strengths and weaknesses are discussed later.

This point is always followed by a citation to the case being relied on. For actual cases (as will be seen below), the citation is in the correct legal form.

Next, the system prints out a response for the government. The government starts by comparing the transactions involved in the two cases. This is unique to CHIRON; we added this comparison because of the centrality of transactions in this domain.

```
<== RESPONSE FOR GOVERNMENT:

DISTINCTIONS:

(1) Differences between transactions that appear in both the
current situation and "the rollover prototype":

   None.
```

```
(2) Transactions that appear only in the current situation:


   None.


(3) Transactions that appear only in "the rollover prototype":

   None.
```

After comparing the transactions in the two cases, the system proceeds as in HYPO, looking for dimensions along which the current situation is weaker for the taxpayer than the cited case (here, the prototype) and cases that are more on point or stronger than the cited case, where the government won. CHIRON finds no weaknesses in the current plan:

```
(4) Weaknesses of the taxpayer's case compared with "the
    rollover prototype":


      None. The current situation is at least as strong for
the taxpayer as "the rollover prototype".
```

but the system does find a case, *Welch v. Commissioner*, where the government won even though the facts were stronger than the prototype.

```
(5) Counter examples:
The following cases hold for the government although they are
stronger than "the rollover prototype" along some dimension:


Dimension: SALE-TO-PURCHASE-TIME
The value in "the rollover prototype" is :
   (OCCURS (SALE-TO-PURCHASE-TIME SELLING157 SELLING613
               (2 0 0)) UT620 UT622)
The government won with a value that was at least as strong
```

```
in:
Case: "Welch v. Commissioner, T.C.M. (P-H) 79,010 (1979)"
   Value: (OCCURS (SALE-TO-PURCHASE-TIME SELLING1 SELLING2
               (0 10 22)) (SEPTEMBER 11 1972) (AUGUST 3 1973))

Dimension: SALE-TO-OCCUPANCY-TIME
The value in "the rollover prototype" is :
   (OCCURS (SALE-TO-OCCUPANCY-TIME SELLING157 SELLING613
               (2 0 0)) UT620 UT624)
The government won with a value that was at least as strong
in:
Case: "Welch v. Commissioner, T.C.M. (P-H) 79,010 (1979)"
   Value: (OCCURS (SALE-TO-OCCUPANCY-TIME SELLING1 SELLING2
               (1 8 2)) (SEPTEMBER 11 1972) (MAY 13 1974))

Dimension: DURATION-OF-OCCUPANCY
The value in "the rollover prototype" is :
   (OCCURS (DURATION-OF-OCCUPANCY GREENLEE 32-ELEVENTH-STREET
                (0 0 1)) UT620 UT620)
The government won with a value that was at least as strong
in:
Case: "Welch v. Commissioner, T.C.M. (P-H) 79,010 (1979)"
   Value: (OCCURS (DURATION-OF-OCCUPANCY MARINKO PLACE1
                (7 0 0)) (1965) (1972))
```

In *Welch*, the system points out, the government won even though the taxpayer had occupied his first residence for seven years, longer than the prototype requires, bought a new piece of property only ten months and 22 days after the sale of his old residence, and occupied the new property a year, eight months, and two days after the sale, both less than the two-year interval allowed by the prototype.[1]

[1]There were two taxpayers involved in this case, one named Welch and one named Marinko. They owned the residences involved in the case jointly, but their cases were considered separately, so they apparently did not file a joint return. Both taxpayers claimed that they were entitled to defer tax under §1034, on the same facts. Marinko also argued that if he failed to qualify under §1034, he should be entitled to an exclusion under §121. For simplicity, rather than representing both cases, we have treated Marinko as the only taxpayer.

Since the government has cited a new case in response, the taxpayer offers a rebuttal. The best rebuttal here would be that *Welch* was decided under an old version of the statute. In the time period covered by this case, the statutory time limit was eighteen months. Because the taxpayer's house was not completed within the time limit, he was unable to occupy it until two months afterwards. The prototype case, on the other hand, is within the current statutory time limit.

CHIRON has no means of reasoning about the dynamic nature of statutes, however, so it must look for other distinctions. First, it notes some differences between the transactions involved in the two cases. In the current situation, the taxpayer buys a house; in *Welch*, the taxpayer bought another type of real property (in fact, land on which to build a house).

```
==> REBUTTAL for TAXPAYER:

"Welch v. Commissioner, T.C.M. (P-H) 79,010 (1979)":

DISTINCTIONS:

(1) Differences between transactions that appear in both the
current situation and "Welch":


With regard to SELLING613:

In the current situation:

(OCCURS (OBJECT SELLING613 UNKNOWN-HOUSE610) FT611 FT612)

In "Welch":

(OCCURS (OBJECT SELLING2 PARCEL1)
                          (AUGUST 3 1973) (AUGUST 3 1973))

UNKNOWN-HOUSE610, the object of SELLING613, and PARCEL1,
the object of SELLING2, are not the same type of property:
```

```
While UNKNOWN-HOUSE610 and PARCEL1 share:

((OCCURS (REAL-PROPERTY UNKNOWN-HOUSE610)
                        (1 1 0) (12 31 99999))
   (OCCURS (REAL-PROPERTY PARCEL1) (1965) (1974)))

UNKNOWN-HOUSE610:

(OCCURS (HOUSE UNKNOWN-HOUSE610)
                          (1 1 0) (12 31 99999))
(OCCURS (REAL-PROPERTY UNKNOWN-HOUSE610)
                          (1 1 0) (12 31 99999))

and PARCEL1:
(OCCURS (REAL-PROPERTY PARCEL1) (1965) (1974))
```

The types of property sold may also be different. The property in *Welch* is of some unspecified type (presumably real estate, as the court would certainly have mentioned the type otherwise); the property in the current situation is a house.

```
32-ELEVENTH-STREET, the object of SELLING157, and PLACE1, the
object of SELLING1, are not the same type of property:


While 32-ELEVENTH-STREET and PLACE1 share:

((OCCURS (REAL-PROPERTY 32-ELEVENTH-STREET)
               (1 1 0) (12 31 99999))
   (OCCURS (REAL-PROPERTY PLACE1)
               (1965) (NOVEMBER 14 1972)))

32-ELEVENTH-STREET:
```

```
(OCCURS (REAL-PROPERTY 32-ELEVENTH-STREET)
           (1 1 0) (12 31 99999))
(OCCURS (HOUSE 32-ELEVENTH-STREET)
           (1 1 0) (12 31 99999))

and PLACE1:
(OCCURS (REAL-PROPERTY PLACE1)
           (1965) (NOVEMBER 14 1972))
```

In addition, *Welch* involved a rental (the taxpayer in that case rented a place to live while his new house was being built); there is no corresponding transaction in the current situation.

```
(2) Transactions that appear only in the current situation:


   None.


(3) Transactions that appear only in "Welch":

RENTING1 appears only in "Welch".
There is no corresponding transaction in the current
situation.

The facts related to RENTING1 are:

(OCCURS (RENTING RENTING1)
           (NOVEMBER 15 1972) (MAY 13 1974))
(OCCURS (LESSEE RENTING1 MARINKO)
           (NOVEMBER 15 1972) (MAY 13 1974))
(OCCURS (OBJECT RENTING1 PLACE2)
           (NOVEMBER 15 1972) (MAY 13 1974))
```

In addition to the differences in transactions, CHIRON finds that the suggested plan is stronger than *Welch* along the shared dimensions of sale-to-purchase-time, sale-to-occupancy-time, and duration-of-occupancy.

```
(4) Strengths of the taxpayer's case compared with "Welch":

"Welch" is distinguishable because
it is stronger for the government than the current situation
along the shared features:
Current situation:
  (OCCURS (SALE-TO-PURCHASE-TIME SELLING157 SELLING613
            (0 0 0)) FT155 FT611)
  "Welch":
  (OCCURS (SALE-TO-PURCHASE-TIME SELLING1 SELLING2
            (0 10 22)) (SEPTEMBER 11 1972) (AUGUST 3 1973))
Current situation:
  (OCCURS (SALE-TO-OCCUPANCY-TIME SELLING157 SELLING613
            (0 0 0)) FT155 (12 31 99999))
  "Welch":
  (OCCURS (SALE-TO-OCCUPANCY-TIME SELLING1 SELLING2
            (1 8 2)) (SEPTEMBER 11 1972) (MAY 13 1974))
Current situation:
  (OCCURS (DURATION-OF-OCCUPANCY GREENLEE 32-ELEVENTH-STREET
            (35 10 20)) (1 1 0) FT155)
  "Welch":
  (OCCURS (DURATION-OF-OCCUPANCY MARINKO PLACE1 (7 0 0))
             (1965) (1972))
```

In summary, the arguments the system makes on behalf of the taxpayer are that *Welch* should not apply because the types of property bought and sold are different; because *Welch* involves a rental transaction, and there is no corresponding transaction in the current situation; and because, although *Welch* is stronger than the prototype along certain dimensions, the plan recommended to Greenlee is even stronger.

The first two arguments do not seem compelling; the types of property involved in the two situations are not significantly different, and the fact that the taxpayer in *Welch* temporarily rented a residence while his new house was being completed should not disqualify him.

The third argument is a strong one, however. The current plan is stronger than *Welch* along exactly the dimensions that the government had cited in comparing *Welch* to the prototype.

Thus, even though the system assumes all its cases are decided under the same version of the statute, it provides strong support for the right conclusion in this case.

After printing out these results for the user, the system backtracks and considers the final strategy recommended by its case-guided search mechanism, the §121 exclusion. This strategy is also rejected, because the statute sets a minimum age of 55 for taking advantage of this provision, and there is no case support for a lower age.

```
**************************************************
                  RESULTS
**************************************************
The strategy :EXCLUSION121 has been rejected for the following
reasons:

  (1) The strategy :EXCLUSION121 is unlikely to succeed
because there is no support for the plan along the
dimension AGE.
In the current situation:
   (OCCURS (AGE GREENLEE 42) (1994) (1994)).

**************************************************
```

The system has now considered all the strategies recommended by the case-guided search mechanism and recommended one of them. It reminds the user which strategy was successful, and offers the user a chance to consider other possibilities:

```
The suggested strategy is :ROLLOVER. Would
you like to consider another strategy (or combination)?

Enter list of strategies or nil: nil
```

From here on, the user can request any strategy or combination of strategies he would like the system to consider. For the moment, we choose not to look at any combinations; this process will be illustrated in a later example.

6.2 The rollover strategy is rejected

In this example, the taxpayer (again called Greenlee) has already sold her house. The sale occurred more than four years ago, and she has not yet bought a new residence. This input is represented internally in the following form:

```
(occurs (individual-return return6) (1994) (1994))
(occurs (taxpayer return6 Greenlee) (1994) (1994))
(occurs (age Greenlee 42) (1994) (1994))
(occurs (house 32-Eleventh-Street) (1 1 0) (12 31 99999))
(occurs (physically-occupy Greenlee 32-Eleventh-Street)
        (6 24 1984) (9 15 1989))
(occurs (physically-occupy Greenlee 7-Park-Place)
        (9 16 1989) (9 20 1994))
(occurs (owns Greenlee 32-Eleventh-Street)
        (6 24 1984) (9 15 1993))
(occurs (selling selling163) (9 15 1993) (9 15 1993))
(occurs (seller selling163 Greenlee)
        (9 15 1993) (9 15 1993))
(occurs (object selling163 32-Eleventh-Street)
        (9 15 1993) (9 15 1993))
```

Since the example involves a sale of real property, the system tries the same three strategies as in the first example. Like-kind exchange is rejected for the same reason as in the first example, because the taxpayer had occupied the property; and again, the taxpayer is too young to take advantage of the §121 exclusion. Additional reasons for rejecting the §121 exclusion in this case are that the taxpayer only occupied the house during one of the five years prior to sale, and that she was away from home for the four years immediately before the sale.

In this case, the rollover strategy is also rejected. CHIRON explains this decision to the user:

```
**************************************************
                  RESULTS
**************************************************
The strategy :ROLLOVER has been rejected for the following
reasons:
```

```
  (1) The strategy :ROLLOVER is unlikely to succeed because
there is no support for the plan along the dimension
ABSENCE-BEFORE-SALE.
In the current situation:
   (OCCURS (ABSENCE-BEFORE-SALE GREENLEE SELLING163 (4 0 0))
             (9 15 1993) (9 15 1993)).

************************************************
```

In our first example, the taxpayer was occupying the house at the time of sale; here, the taxpayer had not been occupying the house for four years before the sale. The only case in the case base where the taxpayer was away from home at the time of sale, *Trisko v. Commissioner*, involved an absence of only three years; this case is even weaker than *Trisko*. Because there is no case support for this weakness, the plan is rejected.

6.3 The rollover strategy is recommended in spite of weak facts

In the third example, the facts are almost the same as in the second. The taxpayer lived at 32 Eleventh Street for three years instead of five, moved, and sold the house a year after moving, instead of four years. These facts are represented as follows:

```
(occurs (individual-return return5) (1994) (1994))
(occurs (taxpayer return5 Greenlee) (1994) (1994))
(occurs (age Greenlee 42) (1994) (1994))
(occurs (house 32-Eleventh-Street) (1 1 0) (12 31 99999))
(occurs (physically-occupy Greenlee 32-Eleventh-Street)
          (6 24 1989) (9 15 1992))
(occurs (physically-occupy Greenlee 7-Park-Place)
          (9 16 1992) (9 21 1994))
(occurs (owns Greenlee 32-Eleventh-Street)
          (6 24 1989) (9 15 1993))
(occurs (selling selling163)
          (9 15 1993) (9 15 1993))
(occurs (seller selling163 Greenlee)
```

```
           (9 15 1993) (9 15 1993))
(occurs (object selling163 32-Eleventh-Street)
           (9 15 1993) (9 15 1993))
```

Again, the system considers the strategies like-kind exchange, §121 exclusion, and rollover, and rejects like-kind exchange and the §121 exclusion. Despite the fact that the taxpayer had not lived in the house for a year before the sale, the system recommends a rollover.

```
*************************************************
                  RESULTS
*************************************************
The strategy :ROLLOVER is suggested.
Code sections involved in plan: (1034).
```

CHIRON suggests that the taxpayer buy and occupy another house. It notes that the taxpayer lived in her house for three years, two months, and twenty-one days before selling it, and that she had only one residence on the date of sale.

```
*****************************************
             PLAN
*****************************************

  ((OCCURS (ABSENCE-BEFORE-SALE GREENLEE SELLING163
             (1 0 0)) (9 15 1993) (9 15 1993)))
  ((OCCURS (NUMBER-OF-RESIDENCES GREENLEE 1)
             (9 15 1993) (9 15 1993)))
  ((OCCURS (DURATION-OF-OCCUPANCY GREENLEE 32-ELEVENTH-STREET
             (3 2 21)) (1 1 0) (9 15 1993)))
  ((OCCURS (BUYER SELLING541 GREENLEE) FT539 FT540))
  ((OCCURS (PHYSICALLY-OCCUPY GREENLEE UNKNOWN-HOUSE538)
             FT536 FT537))
  ((OCCURS (SALE-TO-PURCHASE-TIME SELLING163 SELLING541
             (1 0 7)) (9 15 1993) FT539))
```

```
  ((OCCURS (SALE-TO-OCCUPANCY-TIME SELLING163 SELLING541
             (1 0 7)) (9 15 1993) FT536))
  ((OCCURS (SELLING SELLING541) FT539 FT540))
  ((OCCURS (OBJECT SELLING541 UNKNOWN-HOUSE538)
             FT539 FT540))

*****************************************
```

Optimistically, the system assumes that the new purchase will occur exactly a year and seven days after the sale, that is, the day the example was run. We would need more knowledge about how long it actually takes to buy a house to make a more accurate guess. If both sale and purchase are in the future, this is not a problem; and if one of them occurred more than two years ago, the system will note a weakness in the plan, as it should. For sales that occurred within the past two years, and particularly over a year ago (so that meeting the time limit may be impossible), we would like to be able to say "Buy another house within two years," and express the fact that the sale-to-purchase time will be between one and two years, but that would require more sophisticated temporal reasoning than the system is yet capable of.

Next, CHIRON prints out arguments for and against the success of the suggested plan. As in the first example, the prototype rollover plan is the only most-on-point case. The system argues (on behalf of the taxpayer) that the current plan should succeed like the prototype, because of the facts they have in common.

In response, on behalf of the government, the system admits that the current situation and the prototype contain the same transactions (two sales of real property and purchases of real property), but notes that the current situation is weaker than the prototype, in that the taxpayer, if she executes the suggested plan, would not be occupying the house at the time of sale.

```
(4) Weaknesses of the taxpayer's case compared with
"the rollover prototype":

"the rollover prototype" is distinguishable because
it is stronger for the taxpayer than the current situation
along the shared features:
Current situation:
  (OCCURS (ABSENCE-BEFORE-SALE GREENLEE SELLING163
```

```
          (1 0 0)) (9 15 1993) (9 15 1993))
  "the rollover prototype":
  (OCCURS (ABSENCE-BEFORE-SALE GREENLEE SELLING163
          (0 0 0)) UT555 UT555)
```

Next, as in the first example, CHIRON makes an argument based on *Welch* on behalf of the government and rebuts it on behalf of the taxpayer by noting the differences in transactions. In the second example, the suggested plan was stronger than *Welch* along all three of the dimensions where *Welch* was stronger than the prototype; here, the suggested plan is stronger than *Welch* only on one of those dimensions, sale-to-occupancy-time.

```
(4) Strengths of the taxpayer's case compared with "Welch":

"Welch" is distinguishable because
it is stronger for the government than the current situation
along the shared features:
Current situation:
(OCCURS (SALE-TO-OCCUPANCY-TIME SELLING163 SELLING541
          (1 0 7)) (9 15 1993) FT536)
  "Welch":
  (OCCURS (SALE-TO-OCCUPANCY-TIME SELLING1 SELLING2
          (1 8 2)) (SEPTEMBER 11 1972) (MAY 13 1974))
```

Finally, CHIRON warns the user that the suggested plan is weaker than the prototype.

```
The current situation is weaker than the prototype
in some respects. There is some support for each
of these weaknesses, however:
(1) The current situation is weaker than the
prototype along the dimension ABSENCE-BEFORE-SALE. In the
current situation
  (OCCURS (ABSENCE-BEFORE-SALE GREENLEE SELLING163
          (1 0 0)) (9 15 1993) (9 15 1993))
```

```
However, the taxpayer has won in the following
cases with a weaker position:
  Case: "Trisko v. Commissioner, 29 T.C. 515 (1957)"
   Position:
   (OCCURS (ABSENCE-BEFORE-SALE TRISKO SELLING1
            (3 8 0)) (JUNE 1941) (OCTOBER 10 1951))
**************************************************
```

In other words, the system notes that the fact that the taxpayer was not living in the house at the time of sale raises an issue with respect to this strategy, but there is case support for that weakness, and gives the user a citation to the supporting case.

6.4 The rollover and §121 strategies are both recommended

Here the facts are the same as in the first example – the taxpayer wants to sell a house that she has owned and lived in since 1958 – except that in this case, the taxpayer is 60, rather than 42.

These facts are represented as follows:

```
(occurs (individual-return return1) (1994) (1994))
(occurs (taxpayer return1 Greenlee) (1994) (1994))
(occurs (age Greenlee 60) (1994) (1994))
(goal (occurs (selling selling158) ft156 ft157) Greenlee
        (9 21 1994) (9 21 1994))
(goal (occurs (seller selling158 Greenlee) ft156 ft157)
        Greenlee (9 21 1994) (9 21 1994))
(goal (occurs (object selling158 32-Eleventh-Street)
        ft156 ft157) Greenlee (9 21 1994) (9 21 1994))
(occurs (house 32-Eleventh-Street) (1 1 0) (12 31 99999))
(occurs (physically-occupy Greenlee 32-Eleventh-Street)
        (10 30 1958) (9 21 1994))
(occurs (owns Greenlee 32-Eleventh-Street)
        (10 30 1958) (9 21 1994))
```

Given these facts, CHIRON again considers the like-kind exchange strategy and rejects it on the grounds that the taxpayer has been living in the house. It considers and recommends a

rollover plan, for the same reasons as in the first example. The fact that the taxpayer is older, the only change from the first example, makes no difference to the rollover plan.

In addition to the rollover plan, in this case, the system also recommends a §121 exclusion.

```
**************************************************
                  RESULTS
**************************************************
The strategy :EXCLUSION121 is suggested.
Code sections involved in plan: (121).
```

CHIRON suggests the following plan:

```
****************************************
             PLAN
****************************************

  ((OCCURS (121-ELECTIONS-BEFORE-SALE GREENLEE SELLING157
            0) FT155 FT155))
  ((OCCURS (121-OCCUPANCY-TIME GREENLEE 32-ELEVENTH-STREET
            (4 0 0)) (9 15 1990) FT155))
  ((OCCURS (121-OWNERSHIP-TIME GREENLEE 32-ELEVENTH-STREET
            (4 0 0)) (9 15 1990) FT155))
  ((OCCURS (ABSENCE-BEFORE-SALE GREENLEE SELLING157 (0 0 0))
            FT155 FT155))
  ((OCCURS (NUMBER-OF-RESIDENCES GREENLEE 1) FT155 FT155))
  ((OCCURS (DURATION-OF-OCCUPANCY GREENLEE 32-ELEVENTH-STREET
            (0 0 1)) FT155 FT155))
  ((OCCURS (SIGNS GREENLEE FORM-2119) FT155 FT155))
  ((OCCURS (FILES GREENLEE FORM-2119) FT155 FT155))
  ((OCCURS (OBJECT SELLING157 32-ELEVENTH-STREET)
            FT155 FT156))
  ((OCCURS (SELLER SELLING157 GREENLEE) FT155 FT156))
  ((OCCURS (SELLING SELLING157) FT155 FT156))


****************************************
```

According to this plan, the taxpayer should occupy her house until the date of sale, sell it, and make sure to have only one residence on the date of sale. She should sign and file Form 2119 in order to "make a §121 election," that is, make the official choice to benefit from this provision and make sure not to file a §121 election before that time. §121 singles out the five years immediately prior to the sale as being particularly important; if the taxpayer continues to own and occupy her house until the date of sale, she will have owned and lived in it during at least four out of those five years (more if the sale does not occur immediately).

Next, the system prints out arguments for and against the success of the plan, starting with the facts of the case.

```
*******************************************

   Arguments Supporting the Strategy :EXCLUSION121

*******************************************
```

As in the first example, the taxpayer is filing an individual return, will have one residence on the date of sale, and will live in the house until it is sold.

```
(OCCURS (INDIVIDUAL-RETURN RETURN1) (1994) (1994))
(OCCURS (TAXPAYER RETURN1 GREENLEE) (1994) (1994))
(OCCURS (NUMBER-OF-RESIDENCES GREENLEE 1) FT155 FT155)
(OCCURS (ABSENCE-BEFORE-SALE GREENLEE SELLING157
          (0 0 0)) FT155 FT155)
```

The taxpayer will have owned and lived in the house for four out of the five years prior to the sale (at least). She will not have filed any previous §121 elections, and is now sixty years old.

```
(OCCURS (121-OCCUPANCY-TIME GREENLEE 32-ELEVENTH-STREET
          (4 0 0)) (9 15 1990) FT155)
(OCCURS (121-OWNERSHIP-TIME GREENLEE 32-ELEVENTH-STREET
          (4 0 0)) (9 15 1990) FT155)
(OCCURS (121-ELECTIONS-BEFORE-SALE GREENLEE SELLING157 0)
          FT155 FT155)
(OCCURS (AGE GREENLEE 60) (1994) (1994))
```

The taxpayer will sell a house she has owned and lived in since 1958, and will sign and file Form 2119.

```
(OCCURS (OBJECT SELLING157 32-ELEVENTH-STREET) FT155 FT156)
(OCCURS (SELLER SELLING157 GREENLEE) FT155 FT156)
(OCCURS (SELLING SELLING157) FT155 FT156)
(OCCURS (PHYSICALLY-OCCUPY GREENLEE 32-ELEVENTH-STREET)
          (10 30 1958) (9 15 1994))
(OCCURS (OWNS GREENLEE 32-ELEVENTH-STREET)
          (10 30 1958) (9 15 1994))
(OCCURS (REAL-PROPERTY 32-ELEVENTH-STREET)
          (1 1 0) (12 31 99999))
(OCCURS (HOUSE 32-ELEVENTH-STREET) (1 1 0) (12 31 99999))
(OCCURS (FILES GREENLEE FORM-2119) FT155 FT155)
(OCCURS (SIGNS GREENLEE FORM-2119) FT155 FT155)
```

Note that these are not the same facts given when CHIRON is considering the rollover plan. For example, the taxpayer's age is not considered when analyzing the rollover cases; it is important here. Conversely, whether or not the taxpayer has made a previous §121 election is not considered with regard to the rollover strategy, but is very important here. For each strategy, the system retrieves "relevant" facts, defined as those that have been mentioned in previous cases involving the same strategy.

Although this plan is very similar to the §121 prototype, surprisingly, the system finds most-on-point cases for both sides.

```
The TAXPAYER can cite the following cases
for which there are no more-on-point counter-examples:
   "prototype for Section 121 exclusion"

The GOVERNMENT can cite the following cases
for which there are no more-on-point counter-examples:
   "Rev. Rul. 88-29, 1988-1 C.B. 75"
```

The taxpayer is relying on the prototype; the government, on a Revenue Ruling. Revenue Rulings are cases decided by the Internal Revenue Service after an administrative hearing.

They are not as strong a precedent as a court case, but they are indicative of the IRS's position. If they hold in favor of the taxpayer, they provide useful support. If not, they suggest issues the taxpayer may need to address.

The system begins by making a point on behalf of the taxpayer, based on the similarity of the current situation to the prototype:

CHIRON then generates a three-ply argument based on each of these cases. First, it argues that the taxpayer should win because of the facts the current situation shares with the prototype for this plan, and lists the facts in matched pairs, as in earlier examples. In both the suggested plan and the prototype, the taxpayer sells a house that she has occupied up until the date of sale, and signs and files Form 2119. In both cases, she has only one residence at the time of sale; she has not made any previous §121 election; and she is filing an individual return. She is sixty, while in the prototype, the taxpayer is only 55; and she will have owned and lived in the house for four of the five years prior to sale, while the prototype requires only three.

In response, the system finds no differences between the transactions in the prototype case and the suggested plan, no transactions that occur in one but not in the other, and no dimensions along which the suggested plan is weaker than the prototype. It does find a counter example, another case where the government won on facts that are, in some respects, stronger than the prototype.

This ruling involved a taxpayer in New York City who sold his rights to a rent-controlled apartment to his landlord and wanted to take the §121 exclusion for part of the (apparently sizeable) payment, arguing that he had sold his principal residence. He was sixty years old at the time of sale and still living in the apartment, where he had lived for twenty-five years. He had no other residence. The IRS took the position that these rights, no matter how valuable, were not the kind of ownership rights required by the statute, and it rejected the taxpayer's claim.

Here, the government attempts to use this case, noting that the ruling held for the government even though the taxpayer was older, and had owned the property sold, longer than required by the prototype.

```
(5) Counter examples:
The following cases hold for the government although they are
  stronger than   "prototype for Section 121 exclusion" along
  some dimension:
```

```
Dimension: 121-OWNERSHIP-TIME
The value in "prototype for Section 121 exclusion" is :
   (OCCURS (121-OWNERSHIP-TIME GREENLEE 32-ELEVENTH-STREET
       (3 0 0)) FT155 FT155)
The government won with a value that was at least as strong
in:
Case: "Rev. Rul. 88-29, 1988-1 C.B. 75"
   Value: (OCCURS (121-OWNERSHIP-TIME TAXPAYER RIGHTS1
       (5 0 0)) (1987) (1987))

Dimension: AGE
The value in "prototype for Section 121 exclusion" is :
   (OCCURS (AGE GREENLEE 55) FT155 FT155)
The government won with a value that was at least as strong
in:
Case: "Rev. Rul. 88-29, 1988-1 C.B. 75"
   Value: (OCCURS (AGE TAXPAYER 60) (1987) (1987))
```

In rebuttal, CHIRON identifies exactly the right distinction between the current situation and the Revenue Ruling: they involve the sale of different types of property.

```
==> REBUTTAL for TAXPAYER:

"Rev. Rul. 88-29, 1988-1 C.B. 75":

DISTINCTIONS:

(1) Differences between transactions that appear in both
the current situation and "Rev. Rul. 88-29":


With regard to SELLING157:


In "Rev. Rul. 88-29":
```

```
(OCCURS (BUYER SELLING2 LESSOR1) (1987) (1987))

32-ELEVENTH-STREET, the object of SELLING157, and RIGHTS1,
the object of SELLING2, are not the same type of property:

32-ELEVENTH-STREET:

(OCCURS (REAL-PROPERTY 32-ELEVENTH-STREET)
                                    (1 1 0) (12 31 99999))
(OCCURS (HOUSE 32-ELEVENTH-STREET) (1 1 0) (12 31 99999))

and RIGHTS1:
(OCCURS (INTANGIBLE-PERSONAL-PROPERTY RIGHTS1) (1962) (1987))
(OCCURS (TENANTS-RIGHTS RENTING1 RIGHTS1) (1962) (1987))



There are no corresponding facts in the current situation.
```

In the current situation, the taxpayer is selling a piece of real property; in the Revenue Ruling, the taxpayer sold intangible rights. This is precisely the rationale given by the IRS for its decision. Incidentally, this is also the reason why the Revenue Ruling does not turn up during the process of case-guided search: unlike the current situation, it does not involve a sale of real property.

In addition, CHIRON notes that the Revenue Ruling involved a rental transaction, and the current situation does not.

```
RENTING1 appears only in "Rev. Rul. 88-29".
There is no corresponding transaction in the current
situation.
The facts related to RENTING1 are:

(OCCURS (RENTING RENTING1) (1987) (1987))
(OCCURS (RENT-CONTROLLED RENTING1) (1987) (1987))
```

```
(OCCURS (LESSEE RENTING1 TAXPAYER) (1987) (1987))
(OCCURS (LESSOR RENTING1 LESSOR1) (1987) (1987))
(OCCURS (OBJECT RENTING1 APARTMENT1) (1987) (1987))
(OCCURS (TENANTS-RIGHTS RENTING1 RIGHTS1) (1962) (1987))
```

Next, the system gives a three-ply argument based on the same Revenue Ruling it just used to counter the taxpayer's argument. It starts by arguing that the government should win because of the facts it shares with the Revenue Ruling.

Here the system discusses the facts of the Revenue Ruling in more detail than it used in its counter-example. In both cases, the taxpayer owns property, sells it, and signs and files Form 2119. In both cases, the taxpayer is sixty and has only one residence at the time of sale. Neither taxpayer has ever made a §121 election before, and both are filing individual returns. In the current situation, the taxpayer will have owned the property for four out of the five years prior to sale; in the Revenue Ruling, the taxpayer had owned the rights to his apartment for all five of the relevant years.

```
==> POINT for GOVERNMENT:
The GOVERNMENT should succeed because the GOVERNMENT succeeded
in "Rev. Rul. 88-29", which shares the following FACTS with
the current plan:

Current situation:
  (OCCURS (SIGNS GREENLEE FORM-2119) FT155 FT155)
  "Rev. Rul. 88-29":
  (OCCURS (SIGNS TAXPAYER FORM-2119) (1987) (1987))
Current situation:
  (OCCURS (FILES GREENLEE FORM-2119) FT155 FT155)
  "Rev. Rul. 88-29":
  (OCCURS (FILES TAXPAYER FORM-2119) (1987) (1987))
Current situation:
  (OCCURS (AGE GREENLEE 60) (1994) (1994))
  "Rev. Rul. 88-29":
  (OCCURS (AGE TAXPAYER 60) (1987) (1987))
Current situation:
  (OCCURS (NUMBER-OF-RESIDENCES GREENLEE 1) FT155 FT155)
```

```
  "Rev. Rul. 88-29":
  (OCCURS (NUMBER-OF-RESIDENCES TAXPAYER 1) (1987) (1987))
Current situation:
  (OCCURS (OWNS GREENLEE 32-ELEVENTH-STREET)
          (10 30 1958) (9 15 1994))
  "Rev. Rul. 88-29":
  (OCCURS (OWNS TAXPAYER RIGHTS1) (1962) (1987))
Current situation:
  (OCCURS (SELLING SELLING157) FT155 FT156)
  "Rev. Rul. 88-29":
  (OCCURS (SELLING SELLING2) (1987) (1987))
Current situation:
  (OCCURS (SELLER SELLING157 GREENLEE) FT155 FT156)
  "Rev. Rul. 88-29":
  (OCCURS (SELLER SELLING2 TAXPAYER) (1987) (1987))
Current situation:
  (OCCURS (121-ELECTIONS-BEFORE-SALE GREENLEE SELLING157 0)
          FT155 FT155)
  "Rev. Rul. 88-29":
  (OCCURS (121-ELECTIONS-BEFORE-SALE TAXPAYER SELLING2 0)
          (1987) (1987))
Current situation:
  (OCCURS (ABSENCE-BEFORE-SALE GREENLEE SELLING157
          (0 0 0)) FT155 FT155)
  "Rev. Rul. 88-29":
  (OCCURS (ABSENCE-BEFORE-SALE TAXPAYER SELLING2
          (0 0 0)) (1987) (1987))
Current situation:
  (OCCURS (OBJECT SELLING157 32-ELEVENTH-STREET) FT155 FT156)
  "Rev. Rul. 88-29":
  (OCCURS (OBJECT SELLING2 RIGHTS1) (1987) (1987))
Current situation:
  (OCCURS (TAXPAYER RETURN1 GREENLEE) (1994) (1994))
  "Rev. Rul. 88-29":
```

```
  (OCCURS (TAXPAYER RETURN1 TAXPAYER) (1987) (1987))
Current situation:
  (OCCURS (INDIVIDUAL-RETURN RETURN1) (1994) (1994))
  "Rev. Rul. 88-29":
  (OCCURS (INDIVIDUAL-RETURN RETURN1) (1987) (1987))
Current situation:
   (OCCURS (121-OWNERSHIP-TIME GREENLEE 32-ELEVENTH-STREET
           (4 0 0)) (9 15 1990) FT155)
  "Rev. Rul. 88-29":
  (OCCURS (121-OWNERSHIP-TIME TAXPAYER RIGHTS1
           (5 0 0)) (1987) (1987))

CITE: "Rev. Rul. 88-29, 1988-1 C.B. 75"
```

In response, on behalf of the taxpayer, the system distinguishes the Revenue Ruling in the same way it used earlier. It cites no additional cases, so there is no rebuttal on behalf of the government.

Next, CHIRON reminds the user which strategies it recommended.

```
The suggested strategies are
:EXCLUSION121 and :ROLLOVER

There are no cases in the casebase in which
a combination of these strategies has succeeded.
```

For the first time, there are two recommended strategies, so even though there are no cases in CHIRON's case base where the combination has succeeded, we ask it to consider combining the two.

```
Would you like an analysis of a strategy combination
for the current fact situation? Enter desired combination
or nil: (:rollover :exclusion121)
```

CHIRON recommends the combined plan.

```
************************************************
```

```
         COMBINED RESULTS
**************************************************

Based on the facts of the current situation, the
combination of strategies :ROLLOVER and
:EXCLUSION121 should succeed.

Code sections involved in plan: (121 1034).
```

The combined plan includes the steps from both of the earlier plans:

```
*****************************************
             PLAN
*****************************************

  ((OCCURS (ABSENCE-BEFORE-SALE GREENLEE SELLING157 (0 0 0))
             FT155 FT155))
  ((OCCURS (NUMBER-OF-RESIDENCES GREENLEE 1) FT155 FT155))
  ((OCCURS (DURATION-OF-OCCUPANCY GREENLEE 32-ELEVENTH-STREET
             (35 10 15)) (1 1 0) FT155))
  ((OCCURS (BUYER SELLING1725 GREENLEE) FT1723 FT1724))
  ((OCCURS (PHYSICALLY-OCCUPY GREENLEE UNKNOWN-HOUSE1722)
             FT1720 FT1721))
  ((OCCURS (SALE-TO-PURCHASE-TIME SELLING157 SELLING1725
             (0 0 0)) FT155 FT1723))
  ((OCCURS (SALE-TO-OCCUPANCY-TIME SELLING157 SELLING1725
             (0 0 0)) FT155 (12 31 99999)))
  ((OCCURS (SELLING SELLING1725) FT1723 FT1724))
  ((OCCURS (OBJECT SELLING1725 UNKNOWN-HOUSE1722)
             FT1723 FT1724))
  ((OCCURS (OBJECT SELLING157 32-ELEVENTH-STREET)
             FT155 FT156))
  ((OCCURS (SELLER SELLING157 GREENLEE) FT155 FT156))
  ((OCCURS (SELLING SELLING157) FT155 FT156))
  ((OCCURS (121-ELECTIONS-BEFORE-SALE GREENLEE SELLING157
```

```
              0) FT155 FT155))
  ((OCCURS (121-OCCUPANCY-TIME GREENLEE 32-ELEVENTH-STREET
              (4 0 0)) (9 15 1990) FT155))
  ((OCCURS (121-OWNERSHIP-TIME GREENLEE 32-ELEVENTH-STREET
              (4 0 0)) (9 15 1990) FT155))
  ((OCCURS (DURATION-OF-OCCUPANCY GREENLEE 32-ELEVENTH-STREET
              (0 0 1)) FT155 FT155))
  ((OCCURS (SIGNS GREENLEE FORM-2119) FT155 FT155))
  ((OCCURS (FILES GREENLEE FORM-2119) FT155 FT155))


*****************************************
```

The arguments here are on the question of combining the strategies, rather than about the success on the facts of each individual strategy. The system has no cases where the combination was successful, but one where both strategies failed:

```
*****************************************************

   Arguments Supporting the Combination of Strategies
    :EXCLUSION121 and :ROLLOVER

*****************************************************

On the question of combining the strategies
:EXCLUSION121 and :ROLLOVER
 there are no cases in the case-base where this
 combination has been successful.
The government can cite the following cases
for which there are no more on point counter-examples:
   "Welch v. Commissioner, T.C.M. (P-H) 79,010 (1979)"

==> POINT for GOVERNMENT:
The GOVERNMENT should succeed because the GOVERNMENT
succeeded in "Welch", which shares the following
```

```
STRATEGIES with the current plan:

  :ROLLOVER
  :EXCLUSION121

CITE: "Welch v. Commissioner, T.C.M. (P-H) 79,010 (1979)"

<== RESPONSE FOR TAXPAYER:

None. "Welch v. Commissioner, T.C.M. (P-H) 79,010 (1979)"
includes all the strategies combined in the current
situation.

**************************************************
```

If there were several of these cases, the government might have a strong argument. For example, there are a number of cases where the taxpayer claims deductions on the grounds that he was away from home on business, and also claims moving expenses for the return trip, on the grounds that he was moving to a new principal place of work. (See, e.g. *Goldman v. Commissioner*, 497 F.2d 382 (6th Cir. 1974); Rev. Rul. 74-242, 1974-1 C.B. 69; and *Schweighardt v. Commissioner*, 54 T.C. 1273 (1970)). In general, this combination always loses.

The combination of rollover and §121 exclusion, on the other hand, is not at all unusual; an elderly taxpayer who wants to sell her house and move to a smaller place might well choose to take advantage of both strategies. A single negative case is not significant.

6.5 CHIRON rejects a plan that should have succeeded

Finally, we tried an example with the same facts as *Welch*, twenty years later: the taxpayer sells a residence in Minneapolis that she had been occupying for seven years, buys some land, builds a new house, and occupies the new house twenty months after the sale of the old residence. In the meanwhile, like the taxpayer in *Welch*, she rents a place to live. To simplify the example, we make one minor change: the taxpayer is 42 rather than, like Marinko, old enough to qualify for the §121 exclusion. This way the system will not consider the §121 exclusion, and it

also rejects the like-kind exchange on the grounds that the taxpayer has been occupying the property.

The facts of this example are represented as follows:

```
(OCCURS (TAX-RETURN RETURN2) (1992) (1992))
(OCCURS (TAXPAYER RETURN2 GREENLEE) (1992) (1992))
(OCCURS (AGE GREENLEE 42) (1992) (1992))
(OCCURS (OWNS GREENLEE PLACE1) (1985) (SEPTEMBER 11 1992))
(OCCURS (REAL-PROPERTY PLACE1) (1985) (NOVEMBER 14 1992))
(OCCURS (LOCATION PLACE1 MINNEAPOLIS)
           (1985) (SEPTEMBER 11 1992))
(OCCURS (LOCATION PLACE1 MINNESOTA)
           (1985) (SEPTEMBER 11 1992))
(OCCURS (LOCATION PLACE1 USA)
           (1985) (SEPTEMBER 11 1992))
(OCCURS (PHYSICALLY-OCCUPY GREENLEE PLACE1)
           (1985) (NOVEMBER 14 1992))
(OCCURS (SELLING SELLING1)
           (SEPTEMBER 11 1992) (SEPTEMBER 11 1992))
(OCCURS (SELLER SELLING1 GREENLEE)
           (SEPTEMBER 11 1992) (SEPTEMBER 11 1992))
(OCCURS (OBJECT SELLING1 PLACE1)
           (SEPTEMBER 11 1992) (SEPTEMBER 11 1992))
(OCCURS (RENTING RENTING1)
           (NOVEMBER 15 1992) (MAY 13 1994))
(OCCURS (LESSEE RENTING1 GREENLEE)
           (NOVEMBER 15 1992) (MAY 13 1994))
(OCCURS (OBJECT RENTING1 PLACE2)
           (NOVEMBER 15 1992) (MAY 13 1994))
(OCCURS (REAL-PROPERTY PLACE2)
           (NOVEMBER 15 1992) (MAY 13 1994))
(OCCURS (PHYSICALLY-OCCUPY GREENLEE PLACE2)
           (NOVEMBER 15 1992) (MAY 12 1994))
(OCCURS (LAND PARCEL1) (1985) (1994))
```

```
(OCCURS (REAL-PROPERTY PARCEL1) (1985) (1994))
(GOAL (OCCURS (BUILD GREENLEE HOUSE3) (1992) (1994))
      GREENLEE (1992) (1994))
(GOAL (OCCURS (LOCATION HOUSE3 PARCEL1) (1992) (1994))
      GREENLEE (1992) (1994))
(GOAL (OCCURS (PHYSICALLY-OCCUPY GREENLEE HOUSE3)
      (MARCH 11 1994) (MARCH 12 1994)) GREENLEE (1992) (1994))
(GOAL (OCCURS (PHYSICALLY-OCCUPY GREENLEE PARCEL1)
      (MARCH 11 1994) (MARCH 12 1994)) GREENLEE (1992) (1994))
(OCCURS (BUILD GREENLEE HOUSE3) (1992) (1994))
(OCCURS (HOUSE HOUSE3) (1994) (1994))
(OCCURS (REAL-PROPERTY HOUSE3) (1994) (1994))
(OCCURS (LOCATION HOUSE3 PARCEL1) (1994) (1994))
(OCCURS (PHYSICALLY-OCCUPY GREENLEE HOUSE3)
           (MAY 13 1994) (MAY 14 1994))
(OCCURS (PHYSICALLY-OCCUPY GREENLEE PARCEL1)
           (MAY 13 1994) (MAY 14 1994))
(OCCURS (SELLING SELLING2) (AUGUST 3 1993) (AUGUST 3 1993))
(OCCURS (BUYER SELLING2 GREENLEE)
           (AUGUST 3 1993) (AUGUST 3 1993))
(OCCURS (OBJECT SELLING2 PARCEL1)
           (AUGUST 3 1993) (AUGUST 3 1993))
```

A rollover should have succeeded on these facts; they are no weaker than the prototype, and under current law, they seem quite unproblematic. However, CHIRON rejects the rollover strategy as well:

```
**************************************************
                  RESULTS
**************************************************
The strategy :ROLLOVER has been rejected for the
following reasons:

  (1) The strategy :ROLLOVER is unlikely to succeed,
given the facts of the current situation.
```

```
All of the most-on-point cases held for the government,
including:
"Welch v. Commissioner, T.C.M. (P-H) 79,010 (1979)"

**************************************************
```

CHIRON correctly identifies *Welch* as the only "most-on-point" case. It would be difficult for any other case to share as many facts with the current situation, since the facts are the same, except for the taxpayer's age and the dates. Because the taxpayer lost in *Welch*, the system concludes, the taxpayer will lose in this case as well.

In order to solve this problem correctly, CHIRON would need some way of reasoning about the dynamic nature of statutes. *Welch* is still useful as a precedent; it can be cited, for example, for the proposition that the statutory time limit is fixed and cannot be stretched, even if circumstances beyond the taxpayer's control delay his occupancy of the new residence. It cannot be cited for the proposition that a twenty-month delay exceeds the time limit. To capture this aspect of the relationship between rules and cases, statutes, as well as cases, would need to be time-stamped, and even then, it would be hard to distinguish between those aspects of a case that are no longer meaningful, and those that are still relevant to the current version of the statute. This remains a difficult, but interesting problem for future work.

6.6 Summary

In this chapter, we have given five examples of CHIRON solving problems related to the sale of residential housing. The first three are straightforward examples of the rollover strategy: one where it is recommended, one where it is rejected, and one where it is recommended despite a weakness in the taxpayer's situation. These examples illustrate CHIRON's use of a formal representation for its input and plans. They show the way in which the system's recommendations vary as the facts are varied along different dimensions, such as the age of the taxpayer or the amount of time the taxpayer has been away from home. And they illustrate the compare-and-contrast arguments generated by the system in support of its plans.

In the fourth example, the system recommends and combines two strategies, the rollover and the §121 exclusion. Its arguments supporting the §121 exclusion include a particularly good distinction of a previous case. The taxpayer in that case, Revenue Ruling 88-29, sold his rights to a New York rent-controlled apartment and attempted to exclude part of his gain under

§121. The IRS concluded that the rights to a rent-controlled apartment are more like a lease than the ownership interest required by the statute and rejected the taxpayer's claim. Similarly, CHIRON distinguished the case from the current taxpayer's house sale on the grounds that the two involved sales of different types of property.

Finally, in the fifth example, CHIRON rejects a plan that it should have recommended. Given the same input facts as a previous case (with minor changes), the system concludes that because the taxpayer in the previous case failed, the current plan should be rejected as well.

In the previous case, *Welch v. Commissioner*, the taxpayer sold a residence, bought land and had a house built, and occupied the new house twenty months after the sale of his old residence. The taxpayer in that case was not allowed to defer the gain on the sale of his old residence, because the statute at that time provided that the taxpayer in a rollover transaction must occupy his or her new residence within eighteen months of the sale of the old one.

Under current law, the taxpayer in *Welch* would succeed. The statute now gives taxpayers two years, rather than eighteen months, to occupy a new residence after selling the old one. *Welch* is not invalid; it can still be cited, for example, for the proposition that the deadline is fixed and cannot be extended. But capturing this dynamic relationship between changing statutes and cases is a difficult (and unsolved) problem.

Chapter 7

Evaluating CHIRON

7.1 Introduction

Throughout artificial intelligence, more attention is being given to validation and evaluation of programs [Cohen and Howe, 1988]. If we are to make progress, we must ask what our goals are and whether they have been achieved; and if we want others to use our systems, we must persuade them that the output of those systems will be useful and reliable.

In this thesis, I have described a design for a system that would construct plans using open-textured legal rules and cases. What are the requirements for such a system? I will start by placing the problem in context, then present an ideal model for a legal planning system and evaluate CHIRON's strengths and weaknesses in relation to that model.

7.2 Background

Suppose you are a lawyer of the future, with the ideal legal assistant program. The program would keep you from missing anything important. When you log in in the morning, it would alert you to new and proposed cases and rules in areas that you have specified. When you consult it with a specific question, its interface would be easy to use; it would query you interactively for the necessary information. It would combine analysis and planning: it would have the ability to analyze the current situation and suggest hypotheticals, recognize situations where a plan has been partially completed, and suggest plans for the future. It would reason with legal rules and cases in the same way that a lawyer does.

The system's output would include an analysis of the current situation, if appropriate, or

a selection of possible plans, with issues flagged, and citations to relevant rules and cases. It would prepare analyses of plans it had not already considered, on request. It would also monitor plan execution. For your clients who are in the middle of executing a plan (which could take months or years), the system would notify you of changes in the law that might provide opportunities or problems. It would be integrated with an information-retrieval system such as Lexis or Westlaw so you could simply mouse on one of the citations to see the full text on your screen. It would integrate new cases, statutory provisions, and domain knowledge automatically, and make it easy for you to add your own cases, with the appropriate annotations to supporting rules and cases. And finally, if you select a particular transaction, it would prepare the necessary forms for you.

Lawyers must keep up with a large volume of materials in a constantly-changing domain. Because of increasing competition among lawyers, there is constant pressure to work quickly and keep costs down. Because law is an adversarial domain, any mistakes make a lawyer's work vulnerable to challenge. A computer program that helped to solve these problems would find a receptive audience. During the past twenty or twenty-five years, most law offices have acquired word-processing systems, and most lawyers have become comfortable using information-retrieval systems such as Lexis and Westlaw.

Compare the system's role with that of an associate in a large law firm. Associates are typically recent graduates of law school. Like the partners in the law firm, they have been trained to analyze and reason with legal rules and cases. Their knowledge of the law is more current, but they have less practical experience than the partners. Their research and analysis supplements the practical experience of the more senior lawyers, while they are (hopefully) gaining the skills necessary to become partners themselves.

Unlike a human associate, the computer system would not become a partner. If successful, an associate gradually begins to share control over decisions and contact with clients, but a computer system would continue in a subordinate role. Like a human associate, the ideal computer system would be able to reason with legal rules and cases and provide current legal knowledge. The system would make associates more productive. It would be useful for lawyers in solo practice, who have no associates or partners. And it would orient lawyers in areas of the law with which they are unfamiliar.

7.3 The ideal planner

One of the modules of the ideal legal assistant program would be a planner. Its task would be to generate plans using open-textured rules and cases. Its input would include all of the necessary information about the current situation; another module would have queried the user for the relevant information and translated it into an internal representation. The planner would reason with internal representations of rules and cases, which would have been translated by another module. Its knowledge base would be static during the construction of a given plan. New rules and cases would be processed by a knowledge-acquisition module and integrated into the knowledge base by a learning module. Any ongoing plans affected by changes would be modified by a repair module, in cooperation with the user.

Because the hypothetical users of the system are experts, the ideal planner would give the user as much control as possible. It would let the user suggest strategies for it to consider, output several suggestions rather than just one, where there are several reasonable suggestions, and justify its suggestions with supporting citations and reasoning. No matter how reliable a system is, lawyers will want to be able to check its answers. A human associate would be expected to justify his or her answers in the same way. And in this respect, the computer system has the advantage that it is consistent: it always gives the same answer for the same input. Moreover, it is debuggable: its reasoning process can be reproduced and examined.

The ideal planner would recognize when plans had been partially executed. Often by the time a client consults a lawyer, before the lawyer uses a system, relevant actions would already have been performed. The ideal planner would take this into account in constructing its plans.

The ideal planner's output would have to be reliable. What would it mean for a system to be reliable, in an open-textured domain, where the difference between right and wrong answers is not always clear? Although some answers are more likely than others, most can be argued either way. Here, we will define a reliable system as one that gives all the answers that a good lawyer, experienced with a particular problem, would give.

The ideal planner's output would include plans, citations, and arguments for and against the success of each plan. To gain the trust of expert users, the ideal legal assistant must present solutions that seem reasonable and justify them with citations to appropriate statutes and cases. Its answers should be given in the customary form. A standard legal memorandum includes a statement of the question (including the facts of the current situation), a proposed solution or solutions, and a discussion of the strengths and weaknesses of the plan in comparison with previous cases. The output of the ideal planner would include all the information necessary

for such a memorandum.

In order to generate legal plans and analyze their strengths and weaknesses, the ideal system must be able to reason with legal rules and cases. It must be able to compare and contrast the current situation with previous cases and distinguish easy and hard cases. It should be familiar with the reasoning used in previous similar cases and able to manipulate both the facts of previous cases and abstractions from those cases.

To be sure of generating all the solutions a good lawyer would find, with the appropriate analysis, the system would need a complete knowledge base, including all the relevant cases, statutes, regulations, and information about prospective changes. It would need domain knowledge from treatises and other secondary information. It would be efficient, and convenient for the user, to include plans based on experience of similar situations. These plans could be re-derived from the rules, cases, and domain knowledge, however, so they are not strictly necessary. If present, they should be dated and accompanied by supporting citations. If there have been any new cases or statutory changes that might affect their success, the new information should also be noted.

Ease of use and quick response time are desirable, but less important, features. If the results (and the price) are good enough, lawyers will invest time in learning to use the system. They are accustomed to measuring research time in hours, so a sixty-second response time is not critical for this task.

In summary, the ideal planner would:

- allow the user as much control as possible;
- reason with representations of legal rules, cases, and domain knowledge; and
- generate the plans that a good, experienced lawyer would suggest, with supporting citations and arguments.

7.4 Discussion of CHIRON

No system has yet reached this ideal level of performance; but CHIRON is a step in the right direction. First, it gives the user a substantial amount of control. It could have been designed to select the single "best" combination of strategies, construct a plan, and print it out without comment. Instead, it gives the user a set of plans to choose from, rather than just one. In addition to suggesting plans for individual strategies, the system informs the user which combinations

of those strategies have succeeded in previous cases and allows the user to suggest additional strategies or combinations of strategies for it to work on. As long as the user continues to make requests, the system attempts to generate the desired plans. And each plan generated by the system is accompanied by arguments for and against its success in the current situation, so that the user can assess its merits.

Second, like the ideal planner, CHIRON reasons with representations of legal rules, cases, and domain knowledge. Its knowledge base includes a detailed representation of the facts of its cases and abstractions from those facts, domain knowledge about which features strengthen or weaken a case with respect to a particular strategy, the court deciding each case and the date of the decision.

CHIRON's knowledge base is incomplete, however. In order to measure up to the ideal planner, it will need more knowledge. It can't recommend strategies that it doesn't know about or reason with cases it doesn't have. It can't find similarities based on features or dimensions that are not represented. And it can't reason about the weaknesses of plans compared to the prototype or look for case support for those weaknesses, unless they are identified as weaknesses. Knowledge acquisition is a bottleneck for all legal reasoning systems, and it is particularly difficult for systems with detailed representations like CHIRON's. To be useful, a system will need to find some way of solving this problem.

The ideal system would also have the ability to make explicit use of the reasoning from previous cases. CHIRON does not. It is hard to incorporate this ability in a tax system, since many tax cases do not spell out their reasoning in detail. In theory, the reasoning is part of every court decision; but unlike, say, Supreme Court cases, whose opinions may be fifty or a hundred pages long, the typical tax case is only a couple of pages. The facts are detailed, but the reasoning often is not. Given CHIRON's current case base, comparing and contrasting the facts of the current situation with the facts of previous cases is more important than adapting the reasoning from those cases; as the case base is expanded, the ability to use the reasoning from previous cases should be incorporated as well.

A system that could adapt the reasoning used in previous cases would need to be able to make additional types of argument. Ideally, the system would be able to argue based on the legislative history of a provision, its purpose, the interpretation of occurrences of the same phrase in other provisions or statutes, and underlying theories expressed in treatises. Meta-knowledge about arguments, such as the standard form of "slippery slope" argument, would also be useful. In addition, as discussed in Chapter 6, the system should be able to reason about changes in the statutes over time. And a step beyond the ideal would be a system that

could generate creative arguments: for example, a system that could suggest that it might be possible for a house to be someone's principal residence before any cases had been decided permitting it.

Despite these limitations, CHIRON generates reasonable plans. It cannot find all of the plans that a good lawyer would suggest, but it does examine the strategies that an experienced lawyer would be likely to consider first: those that were used in cases involving the same kinds of transaction found in the current situation. Cases that share the same transaction type are similar in a relevant way, since the United States income tax is based on transfers of property and services. All of these strategies are worth considering. The system focuses the user's attention on these strategies, which are the most relevant to the current situation. If the user wants to consider additional strategies, the system will analyze other possibilities upon request.

CHIRON's approach to planning is a conservative one, consistent with much of tax planning. It filters strategies for the user, recommending only those that it considers to have sufficient case support. For each strategy recommended, there must be at least one most-on-point case with the desired result (i.e., in the tax domain, a case that held for the taxpayer); and for each fact where the current situation is weaker than the prototype case, there must be either a favorable case that is at least as weak, or a trend in that direction. If a strategy is deemed acceptable, CHIRON adapts it into a plan that is as close to the prototype as possible, within the boundaries set by previous cases.

When adapting plans, the system is flexible enough to recognize when a plan has already been partially executed and take that into account. For example, if the client has already sold a house and CHIRON recommends a plan based on the rollover strategy, the steps to be executed will include only the purchase of a second residence, although the plan would ordinarily include both a sale and a purchase.

Like the ideal planner's, CHIRON's output includes citations and arguments for and against the success of each of the plans it suggests. It does not make all of the arguments a good, experienced lawyer would make, but given its limited case base and repertoire of argument types, it does fairly well.

The system distinguishes between easy and hard cases. The current situation is an easy positive case if it does not vary significantly from the prototype, and an easy negative case if it varies so far that it has no adequate case support. It is a hard case if it has weaknesses for which there is at least some case support.

CHIRON retrieves the cases that a good lawyer would consider first when reasoning about a strategy: the cases where the taxpayer attempted, successfully or unsuccessfully, to execute

the same strategy. After comparing each of these cases with the current situation, CHIRON chooses those that are most on-point to the current situation as a basis for its arguments. As in HYPO and CABARET, the determination of which cases are most on-point is based on the facts they share with the current situation. CHIRON's detailed fact representation enables it to make particularly fine-grained distinctions in this respect.

The system's detailed representation is also helpful in comparing and contrasting the current situation with the most-on-point cases, the basis for CHIRON's arguments. The ideal system would have more knowledge about the significance of the comparisons it makes; for example, CHIRON reports the fact that one case involves a house with a tile bathroom and another does not in the same way that it notes that one case involves a house and another involves intangible property, even though the latter is likely to be a much more important distinction. Although the system cannot evaluate its own arguments, however, it does generate some arguments that are right on point, as illustrated in Chapter 6.

Finally, although quick response time is not part of our ideal model, as noted above, it is still a desirable feature. CHIRON's detailed representation is computationally expensive. To compare two cases, instead of comparing simple feature vectors, CHIRON must map the facts of one case onto the facts of the other. Since our representation can easily be translated into GREBE's semantic nets, the problem for CHIRON, as for GREBE, is essentially one of finding isomorphic subgraphs.[1] If our graphs were unlabelled, the problem would be NP-complete. [Garey and Johnson, 1979] But the edges are labelled. Each edge corresponds to a predicate and its endpoints correspond to parameters. Two edges match only if their labels are equal. This fact makes the problem substantially easier (c.f. discussion in [Branting, 1990a, page 42]). How much easier, depends on whether the labels are distinct. In practice, CHIRON's problem is closer to the unique-label problem than to full subgraph isomorphism. Most of its labels are distinct, but a few are duplicates. The average case has 44 facts, and there are often a couple of "sellings," or two or three "objects," or a couple of "sellers." So far, the computational cost of the richer representation seems to be justified by its expressive power.

As the case base grows, the cost of comparisons may present a problem; if so, we will explore the possibility of assigning weights to facts. CHIRON follows HYPO in delaying as far as possible an assignment of weights [Ashley and Rissland, 1988]. GREBE makes a similar decision [Branting, 1990a, page 41]. In general, assigning weights is a difficult problem in

[1] We select the mapping that causes the greatest number of facts in the current situation to be matched, while GREBE looks for the mapping that leaves the fewest facts in the precedent case unmatched. [Branting, 1990a] These criteria reflect a difference in emphasis, but are computationally the same.

this domain. Although lawyers know that certain factors are more important than others, such judgments are heavily dependent on context; facts that seem important in the abstract, may be overshadowed by a particular combination of other facts. Nevertheless, as we develop larger, more detailed case bases, it may be time to begin experimenting with some possible assignments of weights.

7.5 Summary

In this chapter, we have described the ideal legal planning system and compared CHIRON to that model. The ideal planner would function as one component of a larger legal assistant program. Other modules would handle such issues as the interface with the user, case acquisition, and learning new abstractions. The planner itself would need to satisfy the following criteria:

- allowing the user as much control as possible;
- reasoning with representations of legal rules, cases, and domain knowledge; and
- generating the plans that a good, experienced lawyer would suggest, with supporting citations and arguments.

CHIRON is not the ideal system, but it is a step in the right direction. It allows the user a substantial amount of control. It reasons with representations of legal rules, cases, and domain knowledge. It generates reasonable plans, annotated with supporting citations and arguments. It examines the strategies that an experienced lawyer would be likely to consider and recommends those that it considers to have sufficient case support. If a strategy is deemed acceptable, CHIRON adapts it into a plan that is as close to the prototype as possible, within the boundaries set by previous cases, following a conservative approach that is consistent with much of tax planning. When adapting plans, the system is flexible enough to recognize when a plan has already been partially executed and take that into account. CHIRON retrieves reasonable cases, distinguishes between easy and hard cases, and builds arguments for and against the success of its plans by finding similarities between the current situation and previous cases, distinguishing previous cases, indicating how central or peripheral the current situation is (i.e., how it varies from the prototype), and determining trends in a series of cases involving the current strategy.

On the other hand, the system does not suggest all the plans and arguments that a good lawyer would suggest. It is limited by its knowledge base. Knowledge acquisition is a

bottleneck for all legal reasoning systems, and it is particularly difficult for systems with detailed representations like CHIRON's. CHIRON cannot generate arguments based on cases that are not decided under the current provision, but interpret rules that contain some of the same phrases. It doesn't make explicit use of the reasoning from previous cases. It has no facility for constructing arguments based on the legislative history of the law, its purpose, and underlying theories expressed in treatises, or for reasoning about changes in statutes, as well as in the cases interpreting the statutes. It has no meta-knowledge of argument types, such as the standard "slippery slope" argument. Finally, it does not suggest the most aggressive plans. The ideal system would be more flexible, perhaps dependent on the preference of individual lawyers or their clients.

All of the legal reasoning systems discussed in this thesis made steps towards this ideal goal. CHIRON's contributions include the generation of plans based on rules and cases, the integration of planning and arguments, and the generation of arguments based on cases with a detailed fact representation. In addition, the description of the model is itself a contribution, since it provides a standard to measure our systems against.

Chapter 8

Conclusions and Future Work

In this thesis, I have investigated planning issues in law. Planning problems arise in law when an individual (or corporation) wants to perform a sequence of actions that raises legal issues. Using statutes and cases, lawyers construct plans to achieve the desired legal treatment for their clients.

Specifically, I have focused on tax planning. Almost every transaction has some tax aspect, so tax planning forms part of almost all legal planning; and the way in which the statutes and cases are used in this area is typical of general legal planning.

Statutes are rules that have been created formally, by legislation. No matter how detailed, they always contain some important provisions that are not defined in the statute itself. Instead, these phrases are defined partly by commonsense knowledge, partly by example. Each court case is an example — an application of the law to a particular set of facts.

This issue arises throughout legal reasoning, not just in tax. Indeed, it is part of a general natural language problem. Many ordinary categories, such as "tiger" or "cup," are surprisingly difficult to define. This indeterminacy has been studied in linguistics and philosophy, where it is labeled *open texture* [Waismann, 1965, Hart, 1961, Lakoff, 1987]. Any planning rule expressed in natural language, such as "be careful," "never get involved in a land war in Asia," or "buy low, sell high," suffers from the same problem.

The cases only partially define the rules. In general, defining an open-textured rule is not just a matter of inferring defining characteristics from a set of examples. There may not be any set of essential characteristics shared by all positive instances of the rule. Some examples are typical, and others more or less similar to them along various dimensions. Some cases clearly qualify, some clearly do not qualify, and there will always be some borderline cases that could

go either way.

Since the courts are bound by precedent, similar cases must be decided similarly. Thus, planners attempt to construct plans that are similar (or identical) to examples of previous successful plans. The facts of these examples can be used as the basis for new plans.

In constructing a plan based on previous cases, a planner can make use of the court's reasoning. This must be included in the case report along with the facts and result. In a case interpreting a statute, for example, the court will suggest intermediate rules connecting the facts of the case to the open-textured statutory rules in the statute.

These rules have some predictive value. The courts are likely to follow them in later cases. They are not required to do so, however. They are free to adopt new rules, as long as the new rules are consistent with the results in all previous cases. Thus, the courts' reasoning in previous cases is useful in constructing plans, but does not make the success of a plan certain.

The use of open-textured rules and examples has a pervasive effect on legal reasoning. It makes the legal system flexible. The fact that rules are underspecified means that the courts can respond to changing circumstances. On the other hand, because the system is flexible, it is also uncertain. Because there is uncertainty, there is room for argument. Lawyers are trained to find support for different conclusions in a given set of cases. A large part of a lawyer's training involves learning to make arguments for and against the application of some statutory rule, and for and against the similarity of a previous case to the current one. Although one result may be more likely than another, it is generally possible to make these arguments in both directions.

Tax planning is no exception. Here, the adversaries are the taxpayers and the government. Taxpayers seek to exclude or defer items of income and deduct expenses, and the government seeks to include income and disallow deductions. Taxpayers must construct plans that are similar to favorable precedents and different from unfavorable ones in some relevant way. If the similarity is too distant, or the differences are too small, the plans will be vulnerable to challenge by the government.

The open-textured rules and examples interact in interesting ways. Being reminded of a similar case directs the lawyer's attention to the rules applied in that case; being reminded of a potentially applicable rule directs his or her attention to the cases interpreting that rule. In computational terms, rules limit the search through the case base, and cases limit the search through the statutory rules. If you know what statutory rule you are interested in, it is easy to retrieve exactly those cases interpreting that particular rule. Conversely, if you know of a case similar to your own, the case report will indicate which rules were applied in that case.

In my thesis, I have tested this theory of open-textured planning by designing and implementing CHIRON, a system to solve simple problems in the domain of personal income tax planning. Specifically, it solves a cluster of problems having to do with buying, selling, renting, and owning residential housing.

CHIRON is designed to be a lawyer's assistant. The hypothetical user of the system, a lawyer, begins by entering information about a taxpayer's goals and tax situation. Given this input, the system generates plans by which the taxpayer can reduce his or her taxes, annotated with citations to relevant cases and statutes, and arguments for and against the success of each strategy in the current situation.

A legal planning system should reason with representations of legal rules and cases, and to use the cases to define, extend, and limit the rules. Accordingly, in CHIRON, I have combined hierarchical and case-based planners in a hybrid system. The hierarchical planner reasons with representations of open-textured rules; it also reasons with representations of facts. The case-based planner bridges the gap between the two, as legal cases bridge the gap between open-textured rules and facts.

First, the hierarchical planner takes the taxpayer's goals and background information, adds the general system goal (reduce taxes), and constructs a partial plan based on the statutory tax rules. There are a number of plans it could consider; for guidance, it asks the case-based module for a list of plans tried in previous similar cases.

When the hierarchical planner has constructed a partial plan, the case-based planner retrieves a set of cases associated with that plan (cases in which previous taxpayers attempted, with or without success, to execute the same plan). To determine whether the current situation fits within the open-textured rules, the case-based planner compares and contrasts it with previous cases and generates arguments for and against the success of the plan in the current situation, based on the current situation's similarity or dissimilarity to the facts of those previous cases. If the arguments in favor of the plan are sufficient, the case-based planner generates a plan and returns it to the hierarchical planner. Finally, the hierarchical planner reduces the case-based plan to primitive actions.

Because law is an adversarial domain, it is useful to have a safe interpretation of the rules, and also to have some measure of how far you can deviate from that safe interpretation. Accordingly, CHIRON associates a prototype "safe-harbor" plan with each partial plan and uses the previous cases as guidelines to show how far the prototype can be adapted.

Any legal reasoning system's ability to compare and contrast cases, generate arguments, and adapt plans is constrained by its case representation. In order to reason about the relationship

between open-textured rules and cases, a system must be able to:

- distinguish between two cases;
- show similarities between two or more cases;
- indicate which cases are central and which are peripheral; and
- determine trends in a series of cases illustrating a given rule.

A trend in the cases may make it possible to predict the result in the current situation. Comparisons with previous cases may help to predict the result in the current situation, suggest a plan, or highlight problems. The more similar the current plan is to previous cases that satisfy (or fail to satisfy) an open-textured rule, the more likely it is to have the same result. The more it can be distinguished from previous cases, the more likely it is to avoid the result in those cases. The more central a case is, the more likely it is to satisfy a rule.

To support these tasks, what information must be included in a case representation? All of these tasks require both detailed information and abstractions. First, a representation of the facts stated in the official case report is desirable. They are readily available, and all of them are potentially useful in comparing and contrasting cases. Second, abstractions from those facts should also be included, both those used by the court in reasoning from the facts to the relevant open-textured rules, and others derived from general domain knowledge. The more abstractions are included, the more similarities a system can find between cases. To distinguish between central and peripheral cases with regard to a particular rule, information about which features strengthen or weaken a case should also be included. The more central a plan is, the more likely it is to succeed. And finally, the court deciding each case and the date of the decision should be included to make it possible to examine trends in the case law.

CHIRON's case representation includes all of these types of information. The case structure itself has fields for the name (the official legal citation), short-name (an abbreviated form of the citation for use in text), court, date, facts, strategies involved, and holdings of the case, as well as various indices. Information about which features strengthen or weaken a case with regard to a particular strategy is stored separately.

I have attempted to represent the facts as given in the case report as precisely as possible, plus abstractions from those facts that seem useful, based on the reasoning in the particular case, other cases, or general domain knowledge. CHIRON's fact representation language is a modal temporal logic with a formal syntax and semantics. The system's algorithms for comparing and

contrasting cases and building arguments, based on the algorithms used in HYPO and CABARET, exploit this rich representation.

Besides the open texture of rules, another important feature of CHIRON's task environment is the fact that the hypothetical users of the system are experts. In general, experts are much more likely to use a program that offers advice than one that seems to take over part of their expertise; and lawyers are no exception to this rule. Since CHIRON is designed as a lawyer's assistant, it seemed best to give as much control as possible to the user.

CHIRON could have been designed to select the single "best" combination of strategies, construct a plan, and print it out without comment. It has a number of tax-reduction strategies, including, for example, selling a house and buying another to get the §1034 rollover, making a charitable donation, and selling a house and taking the §121 exclusion. These strategies can generally be used either individually or in combination. All of these strategies and all of their possible combinations are candidate plans.

Instead, the user is given a set of plans to choose from, rather than just one. In addition to suggesting individual strategies, the system informs the user which combinations of the suggested strategies have succeeded in previous cases. The user is allowed to suggest plans or combinations of plans for the system to work on. The system attempts to construct a plan combining all the strategies requested by the user, and repeats this process as long as the user continues to request combinations. Finally, each plan constructed by the system is accompanied by arguments for and against its success in the current situation, so that the user can assess its merits. Since the system chooses strategies to analyze and constructs the individual and combined plans, the theoretical problems involved in legal planning are addressed, but without restricting the user any more than necessary.

The contributions of this thesis include:

- developing and implementing a method for generating plans that satisfy open-textured rules;
- designing and implementing a representation of open-textured rules using a prototype, a set of adaptations, and a set of cases;
- combining hierarchical and case-based planners in a hybrid system; and
- designing and implementing a planner that generates plans with supporting arguments for and against the success of each plan, based on comparing and contrasting the current situation with previous cases.

There are a number of directions for future work in this area. One interesting possibility would be to integrate this planner within a general financial planning system, whose goal would be to maximize the client's net worth, rather than simply reducing taxes. Such a system could be extended still further, by incorporating estate planning concepts, to maximize a family's net worth rather than the individual's. Alternatively, in the opposite direction, it would be interesting to look for an area of law where the system could be implemented and tested in a practical context. Finally, the ideal legal assistant program described in Chapter 7 suggests many interesting problems, including case acquisition, user interface design, and learning.

Appendix A

A sample case representation:

Hughston v. Commissioner

```
(make-case
 :name "Hughston v. Commissioner, T.C.M. (P-H) 50,188 (1950)"
 :short-name "Hughston"
 :court "Tax Court"
 :date 1950
 :facts '((occurs (individual-return return2) (1947) (1947))
          (occurs (taxpayer return2 Hughston) (1947) (1947))
          (occurs (house house1) (October 1945) (December 1947))
          (occurs (real-property house1)
                                (October 1945) (December 1947))
          (occurs (location house1 Houston)
                                (October 1945) (December 1947))
          (occurs (location house1 Texas)
                                (October 1945) (December 1947))
          (occurs (location house1 USA)
                                (October 1945) (December 1947))
          (occurs (area house1 (2400 square-feet)) (1947) (1947))
          (occurs (spatial-part room1 house1) (1947) (1947))
          (occurs (bathroom room1) (1947) (1947))
          (occurs (room room1) (1947) (1947))
          (occurs (real-property room1) (1947) (1947))
          (occurs (spatial-part room2 house1) (1947) (1947))
          (occurs (bathroom room2) (1947) (1947))
```

```
(occurs (room room2) (1947) (1947))
(occurs (real-property room2) (1947) (1947))
(occurs (spatial-part room3 house1) (1947) (1947))
(occurs (bathroom room3) (1947) (1947))
(occurs (room room3) (1947) (1947))
(occurs (real-property room3) (1947) (1947))
(occurs (spatial-part room4 house1) (1947) (1947))
(occurs (half-bath room4) (1947) (1947))
(occurs (room room4) (1947) (1947))
(occurs (real-property room4) (1947) (1947))
(occurs (spatial-part room5 house1) (1947) (1947))
(occurs (kitchen room5) (1947) (1947))
(occurs (room room5) (1947) (1947))
(occurs (real-property room5) (1947) (1947))
(occurs (floortype room1 tile) (1947) (1947))
(occurs (floortype room2 tile) (1947) (1947))
(occurs (floortype room3 tile) (1947) (1947))
(occurs (floortype room4 tile) (1947) (1947))
(occurs (floortype room5 tile) (1947) (1947))
(occurs (house house2) (October 1945) (December 1947))
(occurs (real-property house2)
                          (October 1945) (December 1947))
(occurs (location house2 Midland)
                          (October 1945) (December 1947))
(occurs (location house2 Texas)
                          (October 1945) (December 1947))
(occurs (location house2 USA)
                          (October 1945) (December 1947))
(occurs (area house2 (1350 square-feet)) (1947) (1947))
(occurs (spatial-part room6 house2) (1947) (1947))
(occurs (bathroom room6) (1947) (1947))
(occurs (room room6) (1947) (1947))
(occurs (real-property room6) (1947) (1947))
(occurs (spatial-part room7 house2) (1947) (1947))
(occurs (kitchen room7) (1947) (1947))
(occurs (room room7) (1947) (1947))
(occurs (real-property room7) (1947) (1947))
(occurs (not (floortype room6 tile)) (1947) (1947))
```

```
(occurs (not (floortype room7 tile)) (1947) (1947))
(occurs (number-of-bathrooms house1 3.5) (1947) (1947))
(occurs (number-of-bathrooms house2 1) (1947) (1947))
(occurs (physically-occupy Hughston house1)
                      (October 1945) (February 27 1947))
(occurs (physically-occupy Hughston house2)
                               (February 28 1947) (1951))
(occurs (employment employment3) (1946) (1950))
(occurs (employer employment3 Shell-Oil-Company)
                                          (1946) (1950))
(occurs (corporation Shell-Oil-Company) (1946) (1951))
(occurs (employee employment3 Hughston) (1946) (1951))
(occurs (services employment3 task1) (1946) (1951))
(occurs (practice-of-law task1) (1946) (1951))
(occurs (location employment3 Houston)
                               (1946) (February 1947))
(occurs (location employment3 Texas)
                               (1946) (February 1947))
(occurs (location employment3 USA) (1946) (February 1947))
(occurs (location employment3 Midland)
                                     (March 1947) (1950))
(occurs (location employment3 Texas) (March 1947) (1950))
(occurs (location employment3 USA)   (March 1947) (1950))
(occurs (distance Houston Midland (500 miles))
                                  (1946) (1951))
(occurs (lawyer Hughston) (1946) (1951))
(occurs (parent Hughston child1) (1947) (1947))
(occurs (individual child1) (1947) (1947))
(occurs (parent Hughston child2) (1947) (1947))
(occurs (individual child2) (1947) (1947))
(occurs (parent Hughston child3) (1947) (1947))
(occurs (individual child3) (1947) (1947))
(occurs (older child1 child2) (1947) (1947))
(occurs (older child2 child3) (1947) (1947))
(occurs (age child1 10) (1947) (1947))
(occurs (selling selling1)
                  (February 27 1947) (February 27 1947))
(occurs (seller selling1 Hughston)
```

```
                        (February 27 1947) (February 27 1947))
        (occurs (object selling1 house1)
                        (February 27 1947) (February 27 1947))
        (occurs (selling selling2)
                        (February 28 1947) (February 28 1947))
        (occurs (buyer selling2 Hughston)
                        (February 28 1947) (February 28 1947))
        (occurs (object selling2 house2)
                        (February 28 1947) (February 28 1947))

        (occurs (duration-of-occupancy Hughston house1 (1 4 0))
                            (October 1945) (February 27 1947))
        (occurs (duration-of-occupancy Hughston house2 (4 0 0))
                                    (February 28 1947) (1951))
        (occurs (number-of-residences Hughston 1) (1946) (1951))
        (occurs (absence-before-sale Hughston selling1 (0 0 0))
                                                (1947) (1947))
        (occurs (sale-to-purchase-time selling1 selling2 (0 0 1))
                        (February 27 1947) (February 28 1947)) )
:action-types '(:sale)
:property-transferred '(:real-property :cash)
:property-received '(:cash :real-property)
:transfer-types
      '((:sale :real-property :cash)(:sale :cash :real-property))
:strategies '(:like-kind-exchange)
:holdings '((:like-kind-exchange :government)))
```

Appendix B

Text of *Hughston v. Commissioner*, T.C.M. (P-H) 50,188 (1950)

Maurice Harris Hughston; Richard Lively Hughston. Docket Nos. 21922, 21923. 7-25-50.
Richard L. Hughston, Esq., pro se.
Donald P. Chehock, Esq., for the respondent.

MEMORANDUM OPINION

Johnson, Judge:

Respondent determined deficiencies in income tax for the calendar year 1947 in the amounts of $364.74 for petitioner Maurice Harris Hughston, and $260.24 for petitioner Richard Lively Hughston. Petitioners are wife and husband, and the income involved is community income. The husband, Richard Lively Hughston, is hereinafter referred to as petitioner. The sole issue is whether petitioners realized a long-term capital gain in 1947 from the sale of their personal residence. The proceedings were consolidated for hearing. The facts were stipulated.

Petitioner is employed as an attorney by the Shell Oil Company, Inc., and was so employed during 1947 and for some time prior thereto. Petitioners have three children, the oldest of whom was ten years of age on March 1, 1947.

Early in 1947 petitioner and others were transferred by their employer, the Shell Oil Company, Inc., from Houston, Texas to Midland, Texas. Midland is a distance of approximately 500 miles from Houston. Since such transfer, petitioners and family have remained residents of Midland. On or about February 27, 1947, petitioners sold their personal residence at Houston and immediately purchased another at Midland. The Midland home cost about $12,500. Approximately $10,000 in improvements were immediately added by petitioners. The Houston property had some 2,400 square footage in it, with two and a half baths, with tile baths and kitchen; it also had a servant's room with a bath. The

Midland property, as acquired, had only one bath; the kitchen and bath were not tiled. Petitioners added a bedroom and bath, and it now has about 1,700 square feet.

The Houston residence property of petitioners sold in 1947 was previously purchased by petitioners in October, 1945, at a cost of $19,375. The sale price of said property in 1947 was $27,500. The cost to petitioners of improvements, together with the expense of sale, was $2,148.30.

On March 4, 1948, petitioners filed individual community income tax returns for the year 1947 with the collector of internal revenue for the second district of Texas, at Dallas, Texas. The returns so filed did not include in income as a long-term capital gain the difference between the sale price of the Houston residence and the cost of said property. Attached to the community 1947 returns so filed was a letter of the petitioner reading as follows:

March 3, 1948

Collector of Internal Revenue
Dallas, Texas.

Dear Sir:

Herewith the 1947 income tax returns for my wife, Maurice Harris Hughston, and me are enclosed. In the spring of 1947 I was transferred by my employer, Shell Oil Company, Incorporated, from Houston, Texas to Midland, Texas, and as a result was forced to sell my home in Houston and to purchase one in Midland. The Houston place was sold at an increase over its cost price, but we have not included the difference as income because we actually realized no income therefrom, the Midland place being bought on a market more badly inflated than was the Houston one at the time we sold the property there.

We think that the only fair basis for handling such a situation is to carry the cost of the Houston home into the cost of the Midland home. In the event that this cannot be done, we shall be glad to discuss the matter with you.

Very truly yours,
(Signed) Richard L. Hughston

The respondent determined each of the petitioners realized taxable long-term capital gain of $1,494.17 from the sale of the personal residence in Houston, computed as follows:

Gross Sales Price		$27.500.00
Less:	Cost	$19,375.00
	Expense of sale (and improvements)	$2,148.30
		$21,523.30
Total profit		$5,976.70
Held over six months, therefore 50 per cent reportable		$2,988.35
One-half share in community profit (taxable to each petitioner)		$1,494.17

Petitioners maintain that, even though they sold their Houston residence at an increase over its cost, they should not be held to have realized taxable gain on the sale. Their argument is:

> Looking at the substance of the transaction and not at the form thereof, how it can be said that one selling one home and buying another simultaneously in the same generally inflated market receives any beneficial income therefrom cannot be plausibly explained. ... The residence must be replaced on some sort of basis and in the same market a comparable residence would cost as much.

They argue that since they bought a smaller house, they merely realized savings, not gain.

These arguments cannot be seriously entertained. The transaction here in issue was a sale of $27,500 cash of petitioners' Houston residence, which has a cost of $19,375, with expenses of sale and improvements of $2,148.30. The total profit was, therefore, $5,976.70, of which 50 per cent was reportable, as determined by respondent, the property having been held for over six months. Under section 22(a), I.R.C., gross income includes gains from the sale of property. Under section 111(a), I.R.C., the gain from the sale of property is the excess of the amount realized therefrom over the adjusted basis. This was not an exchange solely in kind within the meaning of section 112(b)(1), on which no gain or loss is recognized. It was a sale for cash. What petitioners did with the proceeds of the sale of the Houston house is, as to the tax consequences of that sale, immaterial. We sympathize with petitioners' feeling that it is "inequitable and a hardship on those being transferred during inflationary times to be taxed on the difference between the cost and sale prices of the home they were leaving." An inflation causes inequities that, to some extent, affect us all. Dollars invested one year may be paid off in a later year in dollars worth, by comparison, fifty cents, so that apparent profits are illusory. But the tax law taxes income measured in dollars and cannot take cognizance of the fluctuations in the value of the dollar. Presumably such fluctuations bring as many tax bonanzas as they do tax hardships. For instance, if a sudden period of deflation had followed petitioners' sale of their Houston house and they had been able to purchase a house in Midland of twice the size and desirability for what they had received for their Houston house, the Commissioner would certainly not be heard to say that the difference in value of the two houses represented gain from the Houston sale, which is the substance of petitioners' argument here, but only the amount he has determined here, i.e., the excess of sales price of the Houston house over cost. By the same token, petitioners cannot prevail in their argument here. Accordingly,

Decisions will be entered for the respondent.

Appendix C

CHIRON's cases

1. *Austin v. Commissioner*, T.C.M. (P-H) 77,434 (1977)
2. *Dowd v. Commissioner*, 37 T.C. 399 (1961)
3. *Duke v. Commissioner*, T.C.M. (P-H) 76,038 (1976)
4. *Erdelt v. United States*, 715 F.Supp. 278 (D.N.D. 1989)
5. *Felton v. Commissioner*, T.C.M. (P-H) 82,011 (1982)
6. *Goldman v. Commissioner*, 497 F.2d 382 (6th Cir. 1974)
7. *Hantzis v. Commissioner*, 638 F.2d 248 (1st Cir. 1981)
8. *Hjalmarson v. Commissioner*, T.C.M. (P-H) 81,342 (1981)
9. *Hughston v. Commissioner*, T.C.M. (P-H) 50,188 (1950)
10. *Commissioner v. Janss*, 260 F.2d 99 (8th Cir. 1958)
11. *Johnson v. Commissioner*, T.C.M. (P-H) 88,177 (1988)
12. *Liang v. Commissioner*, T.C.M. (P-H) 75,297 (1975)
13. *Mazzotta v. Commissioner*, T.C.M. (P-H) 71,227 (1971)
14. *Rauchwerger v. Commissioner*, T.C.M. (P-H) 78,177 (1978)
15. *Rev. Rul. 68-12*, 1968-1 C.B. 96
16. *Rev. Rul. 74-242*, 1974-1 C.B. 69
17. *Rev. Rul. 88-29*, 1988-1 C.B. 75
18. *Rosenspan v. United States*, 438 F.2d 905 (2d Cir. 1971)
19. *Sayre v. United States*, 163 F.Supp. 495 (S.D.W.Va. 1958)

20. *Schweighardt v. Commissioner*, 54 T.C. 1273 (1970)
21. *Trisko v. Commissioner*, 29 T.C. 515 (1957)
22. *Tyler v. Commissioner*, T.C.M. (P-H) 82,557 (1982)
23. *Welch v. Commissioner*, T.C.M. (P-H) 79,010 (1979)
24. *Winchell v. United States*, 564 F.Supp. 131 (D.N.D. 1983)

Appendix D

CHIRON's factual predicates

absence-before-sale	age	apartment
area	bank	bathroom
building	buyer	car
cash	condominium	cooperative
corporation	distance	dormitory
duration-of-occupancy	duration-of-ownership	duration-of-rental
121-elections-before-sale	elementary-school	employment
employer	employee	farm
files	financial-part	floortype
government	half-bath	house
individual	individual-return	intangible-personal-property
joint-return	kitchen	labor
laboratory	lawyer	lessee
lessor	location	maintain
near	number-of-bathrooms	number-of-residences
object	121-occupancy-time	121-ownership-time
older	owns	parent
payment	physically-occupy	private-school
public-school	real-property	rent-controlled
renting	research	room
sale-to-occupancy-time	sale-to-purchase-time	seller
selling	services	signs
spatial-part	student	tangible-personal-property
tax-exempt	taxpayer	tenants-rights
trailer	university	war-veteran

Appendix E

CHIRON's property-type hierarchy and other axioms

These rules are fired when the input is asserted to the knowledge base; previous cases already have the appropriate corresponding information included in their list of facts.

```
;; the hierarchy of property types
(-> (occurs (apartment ?x) ?t1 ?t2)
    (occurs (real-property ?x) ?t1 ?t2))
(-> (occurs (building ?x) ?t1 ?t2)
    (occurs (real-property ?x) ?t1 ?t2))
(-> (occurs (condominium ?x) ?t1 ?t2)
    (occurs (real-property ?x) ?t1 ?t2))
(-> (occurs (cooperative ?x) ?t1 ?t2)
    (occurs (real-property ?x) ?t1 ?t2))
(-> (occurs (dormitory ?x) ?t1 ?t2)
    (occurs (real-property ?x) ?t1 ?t2))
(-> (occurs (land ?x) ?t1 ?t2)
    (occurs (real-property ?x) ?t1 ?t2))
(-> (occurs (house ?x) ?t1 ?t2)
    (occurs (real-property ?x) ?t1 ?t2))
(-> (occurs (room ?x) ?t1 ?t2)
    (occurs (real-property ?x) ?t1 ?t2))
(-> (occurs (trailer ?x) ?t1 ?t2)
    (occurs (real-property ?x) ?t1 ?t2))
(-> (and
     (occurs (real-property ?x) ?t1 ?t2)
```

```
     (occurs (spatial-part ?y ?x) ?t3 ?t4))
    (occurs (real-property ?y) ?t3 ?t4))

(-> (occurs (kitchen ?x) ?t1 ?t2)
    (occurs (room ?x) ?t1 ?t2))
(-> (occurs (bathroom ?x) ?t1 ?t2)
    (occurs (room ?x) ?t1 ?t2))
(-> (occurs (half-bath ?x) ?t1 ?t2)
    (occurs (room ?x) ?t1 ?t2))
(-> (occurs (laboratory ?x) ?t1 ?t2)
    (occurs (room ?x) ?t1 ?t2))

(-> (occurs (car ?x) ?t1 ?t2)
    (occurs (tangible-personal-property ?x) ?t1 ?t2))

(-> (occurs (stock ?x) ?t1 ?t2)
    (occurs (intangible-personal-property ?x) ?t1 ?t2))
(-> (occurs (cash ?x) ?t1 ?t2)
    (occurs (intangible-personal-property ?x) ?t1 ?t2))
(-> (occurs (tenants-rights ?renting ?x) ?t1 ?t2)
    (occurs (intangible-personal-property ?x) ?t1 ?t2))

(-> (occurs (real-property ?x) ?t1 ?t2)
    (occurs (property ?x) ?t1 ?t2))
(-> (occurs (tangible-personal-property ?x) ?t1 ?t2)
    (occurs (property ?x) ?t1 ?t2))
(-> (occurs (intangible-personal-property ?x) ?t1 ?t2)
    (occurs (property ?x) ?t1 ?t2))


;; a simple event hierarchy
(-> (occurs (selling ?x) ?t1 ?t2)
    (occurs (transaction ?x) ?t1 ?t2))
(-> (occurs (renting ?x) ?t1 ?t2)
    (occurs (transaction ?x) ?t1 ?t2))
(-> (occurs (lending ?x) ?t1 ?t2)
    (occurs (transaction ?x) ?t1 ?t2))
(-> (occurs (giving ?x) ?t1 ?t2)
```

```
    (occurs (transaction ?x) ?t1 ?t2))
(-> (occurs (employment ?x) ?t1 ?t2)
    (occurs (transaction ?x) ?t1 ?t2))
(-> (occurs (volunteering ?x) ?t1 ?t2)
    (occurs (transaction ?x) ?t1 ?t2))

;; a hierarchy of types of work
(-> (occurs (research ?x) ?t1 ?t2)
    (occurs (labor ?x) ?t1 ?t2))
(-> (occurs (practice-of-law ?x) ?t1 ?t2)
    (occurs (labor ?x) ?t1 ?t2))

;; a hierarchy of employer types
(-> (occurs (elementary-school ?place) ?t1 ?t2)
    (occurs (school ?place) ?t1 ?t2))
(-> (occurs (university ?place) ?t1 ?t2)
    (occurs (school ?place) ?t1 ?t2))
(-> (occurs (private-school ?place) ?t1 ?t2)
    (occurs (school ?place) ?t1 ?t2))
(-> (occurs (public-school ?place) ?t1 ?t2)
    (occurs (school ?place) ?t1 ?t2))
(-> (occurs (public-school ?place) ?t1 ?t2)
    (occurs (government ?place) ?t1 ?t2))
(-> (occurs (government ?place) ?t1 ?t2)
    (occurs (tax-exempt ?place) ?t1 ?t2))
(-> (occurs (bank ?place) ?t1 ?t2)
    (occurs (for-profit ?place) ?t1 ?t2))
(-> (occurs (corporation ?place) ?t1 ?t2)
    (occurs (for-profit ?place) ?t1 ?t2))
(-> (occurs (farm ?place) ?t1 ?t2)
    (occurs (for-profit ?place) ?t1 ?t2))
```

Bibliography

[Alterman, 1986] Alterman, Richard 1986. An adaptive planner. In *Proceedings of the Fifth National Conference on Artificial Intelligence,* Philadelphia, Pennsylvania. (reprinted in *Proceedings of a Workshop on Case-Based Reasoning*, 1988, and *Readings in Planning*, edited by James Allan, James Hendler, and Austin Tate, Morgan Kaufman, 1990).

[Ashley and Rissland, 1987] Ashley, Kevin D. and Rissland, Edwina L. 1987. Compare and contrast, a test of expertise. In *Proceedings of the Sixth National Conference on Artificial Intelligence,* Seattle, Washington. 273–278.

[Ashley and Rissland, 1988] Ashley, Kevin D. and Rissland, Edwina L. 1988. Waiting on weighting: a symbolic least commitment approach. In *Proceedings of the Seventh National Conference on Artificial Intelligence,* Minneapolis, Minnesota. 239–244.

[Ashley, 1991] Ashley, Kevin D. 1991. *Modelling legal argument: reasoning with cases and hypotheticals*. MIT Press. (Revised version of PhD thesis, originally University of Massachusetts at Amherst Department of Computer and Information Science Technical Report 88-01.).

[Bareiss, 1988] Bareiss, Ray 1988. Protos: a unified approach to concept representation, classification, and learning. Technical Report AI88-83, University of Texas at Austin. (PhD Thesis).

[Bench-Capon and Sergot, 1988] Bench-Capon, Trevor and Sergot, Marek J. 1988. Towards a rule-based representation of open texture in law. In *Computer power and legal language*. Quorum Books, New York. 39–60.

[Berman and Hafner, 1991] Berman, Donald H. and Hafner, Carole D. 1991. Incorporating procedural context into a model of case-based legal reasoning. In *Proceedings of the Third International Conference on Artificial Intelligence and Law,* Oxford. 12–20.

[Branting, 1988] Branting, L. Karl 1988. The role of explanations in reasoning from legal precedents. In *Proceedings of a Workshop on Case-Based Reasoning*. 94–103.

[Branting, 1989a] Branting, L. Karl 1989a. Integrating generalizations with exemplar-based reasoning. In *Proceedings of a Workshop on Case-Based Reasoning*. 224–229.

[Branting, 1989b] Branting, L. Karl 1989b. Integrating generalizations with exemplar-based reasoning. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society,* Ann Arbor, Michigan. 139–146.

[Branting, 1989c] Branting, L. Karl 1989c. Representing and reusing explanations of legal precedents. In *Proceedings of the Second International Conference on Artificial Intelligence and Law,* Vancouver, British Columbia. 103–110.

[Branting, 1990a] Branting, L. Karl 1990a. Integrating rules and precedents for classification and explanation: automating legal analysis. Technical Report AI90-146, Artificial Intelligence Laboratory, Department of Computer Sciences, University of Texas at Austin. (PhD Thesis).

[Branting, 1990b] Branting, L. Karl 1990b. (personal communication).

[Branting, 1994] Branting, L. Karl 1994. (personal communication).

[Chirelstein, 1988] Chirelstein, Marvin A. 1988. *Federal Income Taxation: a law student's guide to the leading cases and concepts*. Foundation Press, Westbury, NY.

[Cohen and Howe, 1988] Cohen, Paul R. and Howe, Adele E. 1988. How evaluation guides AI research. *Artificial Intelligence Magazine* 9:35–43.

[Cohen, 1987] Cohen, Robin 1987. Interpreting clues in conjunction with processing restrictions in arguments and discourse. In *Proceedings of the Sixth National Conference on Artificial Intelligence,* Seattle, Washington. 528–533.

[Cuthill and McCartney, 1990] Cuthill, Barbara and McCartney, Robert 1990. Issue spotting in legal cases. In *Proceedings of the Fourth International Conference on Artificial Intelligence and Law,* Amsterdam. 245–253.

[Cuthill, 1992] Cuthill, Barbara 1992. Situation analysis, precedent retrieval, and cross-context reminding in case-based reasoning. Technical Report CSE-TR-92-3, University of Connecticut Department of Computer Science and Engineering. (PhD Thesis).

[Davis, 1990] Davis, Ernest 1990. *Representations of commonsense knowledge*. Morgan Kaufman, San Mateo, California.

[Fikes and Nilsson, 1971] Fikes, Richard and Nilsson, Nils J. 1971. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

[Gardner, 1987] Gardner, Anne v.d.L. 1987. *An artificial intelligence approach to legal reasoning*. MIT Press, Cambridge, Massachusetts.

[Garey and Johnson, 1979] Garey, Michael R. and Johnson, David S. 1979. *Computers and intractability*. W. H. Freeman and Company, New York.

[Ghosh *et al.*, 1991] Ghosh, Subrata; Hendler, James; Kambhampati, Subbarao; and Kettler, Brian 1991. *Common Lisp Implementation of NONLIN USER MANUAL.* Computer Science Department, University of Maryland, College Park, Maryland 20742.

[Golding and Rosenbloom, 1991] Golding, Andrew R. and Rosenbloom, Paul S. 1991. Improving rule-based systems through case-based reasoning. In *Proceedings of the Ninth National Conference on Artificial Intelligence,* Anaheim, California. 22–27.

[Gordon, 1993] Gordon, Thomas F. 1993. The pleadings game: an artificial intelligence model of procedural justice. Technical report, Fachbereich Informatik der Technischen Hochschule Darmstadt. (PhD Thesis).

[Hafner, 1978] Hafner, Carole D. 1978. *An information retrieval system based on a computer model of legal knowledge*. Ph.D. Dissertation, University of Michigan.

[Hammond, 1986] Hammond, Kristian J. 1986. Case-based planning: an integrated theory of planning, learning, and memory. Technical Report YALEU/CSD/RR 488, Yale University Department of Computer Science. (PhD Thesis).

[Hammond, 1989] Hammond, Kristian J. 1989. On functionally motivated vocabularies: an apologia. In *Proceedings of a Workshop on Case-based Reasoning*. 52–56.

[Hanks and Weld, 1992] Hanks, Steven and Weld, Daniel S. 1992. Systematic adaptation for case-based planning. In *Proceedings of the First International Conference on Artificial Intelligence Planning Systems*. 96–105.

[Hart, 1961] Hart, H. L. A. 1961. *The concept of law*. Clarendon Press, Oxford.

[Hinrichs and Kolodner, 1991] Hinrichs, Thomas and Kolodner, Janet 1991. The roles of adaptation in case-based design. In *Proceedings of the Ninth National Conference on Artificial Intelligence,* Anaheim, California.

[Hinrichs, 1991] Hinrichs, Thomas Ryland 1991. Problem solving in open worlds: a case study in design. Technical Report GIT-CC-91/36, Georgia Institute of Technology. (PhD Thesis).

[Hughes and Cresswell, 1968] Hughes, G.E. and Cresswell, M.J. 1968. *An Introduction to Modal Logic*. Methuen, London.

[Kambhampati and Hendler, 1989] Kambhampati, Subbarao and Hendler, James A. 1989. Control of refitting during plan reuse. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* Detroit, Michigan. 943–948.

[Klein and Calderwood, 1988] Klein, Gary R. and Calderwood, Roberta 1988. How do people use analogues to make decisions? In *Proceedings of a Workshop on Case-based Reasoning*. 209–223.

[Lakatos, 1976] Lakatos, Imre 1976. *Proofs and refutations: the logic of mathematical discovery*. Cambridge University Press, Cambridge, England.

[Lakoff, 1987] Lakoff, George 1987. *Women, fire, and dangerous things: what categories reveal about the mind*. University of Chicago Press, Chicago, Illinois.

[Levi, 1949] Levi, Edward H. 1949. *An introduction to legal reasoning*. University of Chicago Press, Chicago, IL.

[Lifschitz, 1987] Lifschitz, Vladimir 1987. Formal theories of action. In *Proceedings of the 1987 Workshop on the Frame Problem in Artificial Intelligence*.

[Llewellyn, 1930] Llewellyn, Karl N. 1930. *The Bramble Bush: our law and its study*. Oceana Publications, Dobbs Ferry, New York.

[Loui *et al.*, 1993] Loui, Ronald P.; Norman, Jeff; Olson, Jon; and Merrill, Andrew 1993. A design for reasoning with policies, precedents, and rationales. In *Proceedings of the Fourth International Conference on Artificial Intelligence and Law,* Amsterdam. 202–211.

[McCartney and Sanders, 1990] McCartney, Robert and Sanders, Kathryn E. 1990. The case for cases: a call for purity in case-based reasoning. In *Proceedings of the AAAI Symposium on Case-based Reasoning*. 12–16.

[McCartney and Wurst, 1991] McCartney, Robert and Wurst, Karl R. 1991. DEFARGE: a real-time execution monitor for a case-based planner. In *Proceedings of the Workshop on Case-Based Reasoning*. 233–244.

[McCartney, 1993] McCartney, Robert 1993. Planning from partial and multiple episodes in a case-based planner. In *Proceedings of the 1993 AAAI Workshop on Case-Based Reasoning,* Washington, D.C. 94–100.

[McCarty and Sridharan, 1982] McCarty, L. Thorne and Sridharan, N.S. 1982. A computational theory of legal argument. Technical Report LRP-TR-13, Laboratory for Computer Science Research, Rutgers University.

[McCarty, 1977] McCarty, L. Thorne 1977. Reflections on TAXMAN: an experiment in artificial intelligence and legal reasoning. *Harvard Law Review* 90:837–893.

[McCarty, 1980] McCarty, L. Thorne 1980. The TAXMAN project: towards a cognitive theory of legal argument. In Niblett, Bryan, editor 1980, *Computer Science and Law*. Cambridge University Press, Cambridge, England. 23–43.

[McCarty, 1983] McCarty, L. Thorne 1983. Permissions and obligations. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* Karlsruhe, West Germany. 287–294.

[McCarty, 1986] McCarty, L. Thorne 1986. Permissions and obligations: an informal introduction. Technical Report LRP-TR-19, Laboratory for Computer Science Research, Rutgers University.

[McCarty, 1989a] McCarty, L. Thorne 1989a. Computing with prototypes (preliminary report). In *Proceedings of the Bar-Ilan Symposium on the Foundations of Artificial Intelligence*.

[McCarty, 1989b] McCarty, L. Thorne 1989b. A language for legal discourse: I. basic features. In *Proceedings of the Second International Conference on Artificial Intelligence and Law,* Vancouver, British Columbia. 180–189.

[McDermott, 1985] McDermott, Drew 1985. The DUCK manual. Technical Report YALEU/CSD/RR 399, Yale University Department of Computer Science.

[Merryman, 1969] Merryman, John Henry 1969. *The civil law tradition*. Stanford University Press, Palo Alto, CA.

[Oskamp *et al.*, 1989] Oskamp, A.; Walker, R.F.; Schrickx, J.A.; and Berg, P.H.van den 1989. Prolexs, divide and rule: a legal application. In *Proceedings of the Second International Conference on Artificial Intelligence and Law,* Vancouver, British Columbia. 54–62.

[Redmond, 1990] Redmond, Michael 1990. Distributed cases for case-based reasoning; facilitating use of multiple cases. In *Proceedings of the Eighth National Conference on Artificial Intelligence,* Boston, Massachusetts. 304–309.

[Riesbeck and Schank, 1989] Riesbeck, Christopher K. and Schank, Roger C. 1989. *Inside Case-based Reasoning*. Lawrence Erlbaum Associates, Hillsdale, NJ.

[Rissland and Skalak, 1989a] Rissland, Edwina L. and Skalak, David B. 1989a. Case-based reasoning in a rule-governed domain. In *Proceedings of the Fifth IEEE Conference on Artificial Intelligence Applications,* Miami, Florida. 45–53.

[Rissland and Skalak, 1989b] Rissland, Edwina L. and Skalak, David B. 1989b. Combining case-based and rule-based reasoning: a heuristic approach. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* Detroit, Michigan. 524–530.

[Rissland and Skalak, 1989c] Rissland, Edwina L. and Skalak, David B. 1989c. Interpreting statutory predicates. In *Proceedings of the Second International Conference on Artificial Intelligence and Law,* Vancouver, British Columbia. 46–53.

[Rissland and Skalak, 1991] Rissland, Edwina L. and Skalak, David B. 1991. CABARET: rule interpretation in a hybrid architecture. *International Journal of Man-Machine Studies* 839–887.

[Rissland and Soloway, 1980] Rissland, Edwina L. and Soloway, Elliot M. 1980. Overview of an example generation system. In *Proceedings of the First National Conference on Artificial Intelligence*. 256–258.

[Rissland *et al.*, 1993a] Rissland, Edwina L.; Daniels, Jody J.; Rubinstein, Zachary B.; and Skalak, David B. 1993a. Case-based diagnostic analysis in a blackboard architecture. In *Proceedings of the Eleventh National Conference on Artificial Intelligence,* Washington, D.C.

[Rissland *et al.*, 1993b] Rissland, Edwina L.; Daniels, Jody J.; Rubinstein, Zachary B.; and Skalak, David B. 1993b. Diagnostic case retrieval guided by evaluation and feedback. In *Proceedings of the AAAI Workshop on Case-Based Reasoning,* Washington, D.C.

[Rissland *et al.*, 1994] Rissland, Edwina L.; Skalak, David B.; and Friedman, M. Timur 1994. Heuristic harvesting of information for case-based argument. In *Proceedings of the Twelfth National Conference on Artificial Intelligence,* Seattle, Washington. 36–43.

[Rissland (Michener), 1978] Rissland (Michener), Edwina 1978. Epistemology, representation, understanding and interactive exploration of mathematical theories. Technical report, Department of Mathematics, Massachusetts Institute of Technology. (PhD Thesis).

[Rissland, 1982] Rissland, Edwina L. 1982. Examples in the legal domain: hypotheticals in contract law. In *Proceedings of the Fourth Annual Conference on Cognitive Science,* Ann Arbor, Michigan.

[Rosch and Mervis, 1975] Rosch, Eleanor and Mervis, C.B. 1975. Family resemblance: studies in the internal structure of categories. *Cognitive Psychology* 7:573–605.

[Sacerdoti, 1974] Sacerdoti, Earl 1974. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence* 7:231–272.

[Sanders, 1989a] Sanders, Kathryn E. 1989a. A logic for emotions: a basis for reasoning about commonsense psychological knowledge. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society,* Ann Arbor, Michigan.

[Sanders, 1989b] Sanders, Kathryn E. 1989b. A logic for emotions: a basis for reasoning about commonsense psychological knowledge. Technical Report 89-23, Brown University Computer Science Department.

[Sanders, 1990] Sanders, Kathryn E. 1990. The truth, the whole truth, and nothing but the truth: case representations for legal reasoning systems. In *Proceedings of the AAAI Workshop on Legal Reasoning,* Boston, Massachusetts.

[Sanders, 1991a] Sanders, Kathryn E. 1991a. Representing and reasoning about open-textured predicates. In *Proceedings of the Third International Conference on Artificial Intelligence and Law,* Oxford, England. 137–144.

[Sanders, 1991b] Sanders, Kathryn E. 1991b. Within the letter of the law: planning among multiple cases. In *Proceedings of the DARPA Workshop on Case-Based Reasoning,* Washington, D.C.

[Sanders, 1993] Sanders, Kathryn E. 1993. If it worked last time, it should work again: justifying and validating case-based plans. In *Proceedings of the AAAI Workshop on Case-Based Reasoning*.

[Schlobohm and McCarty, 1989] Schlobohm, Dean and McCarty, L. Thorne 1989. EPS II: Estate planning with prototypes. In *Proceedings of the Second International Conference on Artificial Intelligence and Law,* Vancouver, British Columbia. 1–10.

[Sergot *et al.*, 1986] Sergot, M. J.; Sadri, F.; Kowalski, R. A.; Kriwaczek, F.; Hammond, P.; and Cory, H. T. 1986. The british nationality act as a logic program. *Communications of the Association for Computing Machinery* 29(5):370–386.

[Shoham, 1988] Shoham, Yoav 1988. *Reasoning about change: time and causation from the standpoint of artificial intelligence*. MIT Press, Cambridge, Massachusetts.

[Skalak and Rissland, 1992] Skalak, David B. and Rissland, Edwina L. 1992. Arguments and cases: an inevitable intertwining. *Artificial Intelligence and Law: an international journal* 1(1):3–44.

[Skalak, 1989a] Skalak, David B. 1989a. Options for controlling mixed paradigm systems. In *Proceedings of a Workshop on Case-Based Reasoning*. 318–323.

[Skalak, 1989b] Skalak, David B. 1989b. Taking advantage of models for legal classification. In *Proceedings of the Second International Conference on Artificial Intelligence and Law,* Vancouver, British Columbia. 234–241.

[Slator and Riesbeck, 1992] Slator, Brian M. and Riesbeck, Christopher K. 1992. Taxops: a case-based advisor. *International Journal of Expert Systems* 4(2):117–140.

[Statsky and Wernet, Jr., 1989] Statsky, William P. and Wernet, Jr., R. John 1989. *Case analysis and fundamentals of legal writing*. West Publishing Company, St. Paul, Minnesota.

[Tate, 1976] Tate, Austin 1976. Project planning using a hierarchic non-linear planner. Technical Report 245, Department of Artificial Intelligence, University of Edinburgh.

[Tate, 1977] Tate, Austin 1977. Generating project networks. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*. 888–893.

[Veloso, 1992] Veloso, Manuela M. 1992. Learning by analogical reasoning in general problem solving. Technical Report CMU-CS-92-174, School of Computer Science, Carnegie Mellon University. (PhD Thesis).

[Waismann, 1965] Waismann, Friedrich 1965. Verifiability. In Flew, Antony, editor 1965, *Logic and language: first and second series*. Anchor Books, Garden City, New Jersey. 122–151. (first published in *Proceedings of the Aristotelian Society,* 119-150 (1945).).

[Waltz, 1989] Waltz, David L. 1989. Is indexing used for retrieval? In *Proceedings of a Workshop on Case-based Reasoning*. 41–44.

[Waterman, 1986] Waterman, Donald A. 1986. *A guide to expert systems*. Addison-Wesley Publishing Company.

[Wright, 1951] Wright, G. H. von 1951. *An Essay in Modal Logic*. North-Holland.

[Zito-Wolf and Alterman, 1992] Zito-Wolf, Roland and Alterman, Richard 1992. Multicases: a case-based representation for procedural knowledge. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*.