# Representing and reasoning about open-textured predicates

Kathryn E. Sanders
Department of Computer Science, Box 1910
Brown University, Providence, RI 02912 U.S.A.
ks@cs.brown.edu

April 29, 1991

## Abstract

In this paper, I will describe a method for representing and reasoning about open-textured predicates. This method is being implemented in CHIRON, a system I am developing in the domain of United States personal income tax planning.

## 1 Introduction

In this paper, I will describe a method for representing and reasoning about open-textured predicates. This method is being implemented in CHIRON, a system I am developing in the domain of United States personal income tax planning [Sanders, 1991].

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.

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. 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 CHIRON, therefore, I am examining the way in which plans are developed from statutes and cases.

Like other Anglo-American statutes, the United States Internal Revenue Code is open-textured. It contains phrases that are underspecified. For example, §1034 of the Internal Revenue Code provides that if an individual sells his "principal residence" and buys and uses another "principal residence" within a certain period of time, he can defer paying tax on the income (if any) from the sale. The phrase "principal residence" is not defined within the statute. Some information about the legal meaning of the phrase can be derived from the commonsense meaning of the words; additional information is obtained from cases.

The open-textured nature of the rules places a major constraint on planners in this domain. Somehow, starting with open-textured rules, the planner must generate a set of facts that satisfy (or avoid satisfying) those rules.

Each case makes a connection between the rules and one particular set of facts. Information about the taxpayer and his actions is given in the statement of facts; given those facts, the court determines whether certain statutory rules have been satisfied or not. If you look at the statement of facts as a plan (although possibly not a plan the taxpayer intended), cases can be seen as the evaluation of that plan by a court. Cases provide examples of plans that succeeded or failed.

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 construct plans that are similar to previous successful plans and different from unsuccessful ones.

For a legal reasoning system to perform this task, it must have some method for representing and reasoning about open-textured predicates. In Section 2 of this paper, I will survey various approaches used for reasoning about open-textured rules in related work. In Section 3, I will describe the method used in CHIRON, in Section 4, I will give a detailed example, and in Section 5, I will summarize the results of this paper.

## 2 Related Work

There is a long tradition of work on planning and design in artificial intelligence (see, e.g., [Allen *et al.*, 1990]). In terms of one recent framework, CHIRON can be classified as a system whose generic task is construction in a domain without a complete theory, where it is necessary to reason with weakly-defined concepts and the number of possible plans is so large that exhaustive search is, if not impossible, certainly impractical [Steels, 1990]. Planning in law shares many issues with planning in other domains; indeed, the problem of reasoning with weakly-defined, or open-textured, rules is not unique to law. 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," may suffer from the same problem.

The previous literature in planning has not addressed the problem of reasoning with open-textured rules, however. Perhaps because the issue is especially well illustrated in law, it has been given much more attention in the artificial intelligence and law literature. In this section, I will survey the approaches to reasoning about open-textured rules that are most closely related to my own.

### 2.1 McCarty

McCarty was perhaps the first to address the problem of representing and reasoning with open-textured predicates in the context of artificial intelligence and law. His first project was TAXMAN, a program that analyzed cases in the domain of corporate tax law. 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 [McCarty, 1977].

TAXMAN took as input the description of a corporate reorganization 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 backwards chaining rules to determine whether the expanded facts satisfied the 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 rules. This works fairly well for some simple inferences, such as determining that someone is a stockholder. 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.

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.

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, 1989].

Unfortunately, TAXMAN II was never implemented. As a result, McCarty offers no solution to algorithmic issues such as 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 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, no solution is proposed for the algorithmic issues such as how the prototypes are chosen, indexing, and search. And again, there is no explicit facility for representing or reasoning with legal cases.

## 2.2 Gardner

Gardner has also addressed the open-textured statutory predicate problem. Her system, GP,[1] like TAXMAN, takes a sequence of events as input and determines whether that sequence satisfies certain legal requirements [Gardner, 1987]. GP'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. First, GP 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, GP 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 GP 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 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. When analyzing an input case with regard to a particular open-textured statutory predicate, GP retrieves the cases involving that statutory predicate whose facts match the facts of the statutory predicate's case pattern.

In effect, each open-textured predicate corresponds to a set of cases. The relationships between the cases are not spelled out, however. There are no levels of abstraction between the facts and the statutory predicates. GP has no information about case transformations, such as those suggested by McCarty. As a result, it is hard for GP 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.

## 2.3 HYPO

HYPO illustrates a use of cases that is different from Gardner's [Ashley, 1988]. Like TAXMAN and GP, HYPO takes 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 attempt to solve the problem. 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 is stored using a fixed set of indices (termed "dimensions"). When the user inputs a legal situation, expressed in the legal case-frame language, HYPO uses this representation to calculate the dimension values for the situation, and then uses these values to index into the most similar cases. The system then constructs arguments for both plaintiff and defendant based on the dimensions each retrieved case shares (or fails to share) with the current situation.

HYPO does not incorporate rules explicitly, but it is addressing the same open-textured statutory predicate problem as TAXMAN II and GP. Like GP, HYPO uses a set of cases to correspond to an open-textured predicate. In effect, the whole system can be seen as using cases to interpret a single open-textured predicate, "trade secret violation." All the cases in the case base correspond to that predicate. Unlike GP, 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 GP could not have done. In GP, the open-textured predicate corresponds to an unstructured set of cases; in HYPO, the set of cases is structured by the dimensions.

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 a prototype, however.

## 2.4 CABARET

CABARET's solution to the problem of reasoning about open-textured rules is similar to HYPO's [Rissland and Skalak, 1991]. Like HYPO, CABARET is a legal analysis system. It takes a set of facts as input and generates an argument for or against the application of open-textured rules to those facts.

Unlike HYPO, CABARET applies explicit statutory rules to its input. Instead of the single implicit predicate "trade secrets violation," which was the subject of HYPO's arguments, CABARET interprets §280A of the United States Internal Revenue Code, the provision governing deductions for the cost of maintaining an office at home. To perform this task, CABARET integrates a case-based reasoner modelled on HYPO with a rule-based

[1] "Gardner's program." So called for convenience, since Gardner did not name her system.

reasoner.

The main contribution of this project is in the area of controlling mixed case-based and rule-based systems. CABARET interleaves case-based and rule-based reasoning using control knowledge isolated in a separate module. The use of cases and dimensions is essentially the same as HYPO's, and as in HYPO, there is no notion of a prototype.

## 2.5 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]. 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.

# 3 CHIRON's solution

CHIRON is a planner. It takes as input a transaction the client intends to perform and (optional) background information, and outputs a plan or plans for performing that transaction with favorable income tax consequences. In order to perform this task, it must represent and reason with the open-textured rules of the income tax law. Its solution to this problem combines rules, prototypes, cases, and dimensions, drawing on ideas from McCarty, Gardner, HYPO, and CABARET.

Like CABARET, CHIRON is a hybrid system, combining rule-based and case-based modules. Since control is not a primary focus of this project, however, CHIRON's control structure is much simpler than CABARET's. Planning proceeds in two stages: first, the rule-based planner constructs a partial plan based on the tax rules. This plan will contain open-textured subplans corresponding to the open-textured portions of the rules on which it is based. Second, the case-based planner takes the partial plan and uses it as the basis for a plan that can be executed by the user. In this paper, I will focus on the second stage; for further details about the construction of the partial plan, see [Sanders, 1991].

In the second stage of planning, CHIRON's case-based module uses the partial plan to index into its case base. It retrieves a prototype plan corresponding to the partial plan and a set of past cases where taxpayers attempted, successfully or unsuccessfully, to execute plans that also correspond to the partial plan.

The prototypes are based partly on the commonsense meaning of the statutory predicates. Terms such as "principal residence," although they are underspecified, do have some commonsense meaning. Additional information can be obtained from the cases. In any law case, there will be some easy questions which 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.

After retrieving the prototype and a set of past cases, the case-based planner then tests the prototype plan to determine whether it satisfies the constraints input by the user. If so, it outputs the prototype as a suggested plan. If not, it has three choices: adapt the prototype so that it will satisfy the constraints, relax the violated constraints, or abandon the plan.

The ways in which a prototype plan can be adapted are the *dimensions*; they are stored explicitly in a separate domain knowledge module. The dimensions are suggested by cases. For example, the cases interpreting principal residence consider whether a house is a 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 not lived in the house for years and do not choose to move back, etc. The corresponding dimensions are: number of the taxpayer's residences, amount of time since the taxpayer occupied this house, strength of reason for not returning to the house.

The prototypes are related to the actual cases by the dimensions. 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.

CHIRON's domain knowledge module also contains information about which types of constraints can be relaxed. If the client is not living in his house, for example, CHIRON may advise him to move in; but if the client is 22 years old, CHIRON will not suggest that he become 55.

If the case-based planner succeeds, it outputs a plan

annotated with citations to supporting rules and cases. After outputting a plan, CHIRON will query the user to determine whether to continue. If the case-based planner succeeds and the user requests another plan, the case-based planner returns control to the rule-based planner, which then attempts to construct another plan for the same transaction. Similarly, if the case-based planner is forced to abandon a plan, it returns control to the rule-based planner, which continues and attempts to construct another plan. Plans are output roughly in order of their desirability from a tax point of view, but rather than attempting to construct a single "best" plan, CHIRON leaves the final choice of plan up to the user. The system will halt when the user stops requesting new plans, or when it is unable to find another possible plan. A diagram of this architecture is given in Figure 1.

Like TAXMAN II, CHIRON uses prototypes and transformations (dimensions). CHIRON attempts to provide answers to some of the questions left unanswered by TAXMAN II: how to choose the prototypes, how to index them, how to search the space of prototypes, how to choose and retrieve transformations, and so forth. In addition, CHIRON explicitly incorporates cases into this process. Like GP, CHIRON uses both rules and cases, but as in HYPO and CABARET, the cases are related to each other by dimensions. Unlike GP, HYPO, or CABARET, CHIRON uses prototypes. Unlike GP, HYPO, or CABARET, CHIRON is a planner. As a result, it must generate a set of facts rather than recognizing one. Since there will usually be a range of possibilities corresponding to any open-textured plan, it is useful to have a default plan, which can be chosen unless there is some reason to do otherwise, and the prototype serves as such a default.

Another way of looking at this is that in CHIRON, the court cases partially define a space of possible plans, with the prototype at the origin. CHIRON's goal is to construct a plan that falls within that space of possibilities. Specifically, it will construct a plan as similar to the prototype as possible. The prototype is a conservative plan, what a tax lawyer would call a "safe harbor." Thus, CHIRON's strategy gives plans a conservative bias, which is consistent with much of tax planning; it could be altered if desired to obtain a more aggressive planner.

CHIRON's approach resembles the propose-and-revise method used by many construction systems [Steels, 1990]. These systems can typically be broken down into three modules: generate a partial solution, test the solution to determine whether the input constraints have been satisfied, and if necessary, adapt the partial solution so that the constraints are satisfied. The partial solution corresponds to CHIRON's prototype plan; and the constraints correspond to the transaction and background information input by the user. CHIRON differs from these systems in that it provides an explicit role for previous cases and relates them to the partial solution (prototype). In addition, it allows for the possibility of relaxing the constraints in certain cases.

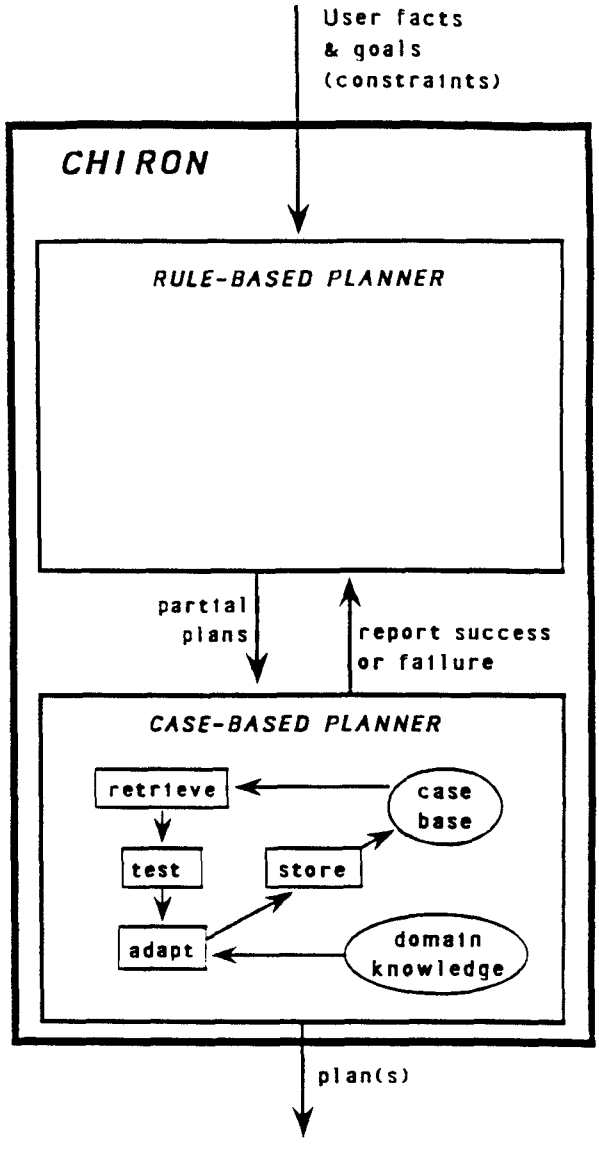


Figure 1: CHIRON's architecture.

## 4 Example

Suppose you have a client who wants to sell a house. Under United States income tax law, some possible plans are:

**§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 2: §1034(a) of the Internal Revenue Code.

- sell the house and pay tax on the income;
- sell the house at a loss;
- occupy the house until the time of sale (to make it a principal residence) and buy and occupy another house within the statutory time limit (currently two years).
- occupy the house until the time of sale, and, if the taxpayer is over 55, take a one-time exclusion of a portion of the gain;
- occupy the house until the time of sale, take the over-55 exclusion, and invest the remainder of the gain in another house;
- rent the house to tenants and exchange it for another house that is also rented; and
- sell the house and donate the income to charity.

Suppose you input the information that the client has the goal of selling a house, but no additional background information. CHIRON will then attempt to construct a plan that includes selling the house, and in addition, satisfies the system goal of reducing the income tax due on this transaction. First, CHIRON's rule-based module will construct a partial plan satisfying these goals. Suppose it chooses the plan suggested by §1034 of the Internal Revenue Code: show that the house is the client's principal residence at the time of sale, and buy and use another principal residence within two years before or after the sale. A relevant portion of §1034 is given in Figure 2.

This plan is still too abstract. "Principal residence" is an open-textured predicate. So CHIRON retrieves the prototype for this plan and all the cases from its case base where a taxpayer attempted to execute the plan. The prototype is: physically occupy the old house until the date of sale or the date of purchase of a new house, whichever comes first; and after you buy the new house, move there and physically occupy that house. ("Physically occupy" is a term used to include sleeping in a house, keeping your possessions there, eating your meals there, and so forth.)

There are a number of cases interpreting this partial plan; one of the most interesting is *Trisko v. Commissioner*, 29 T.C. 515 (1957). In this case, the taxpayer sold a house that he hadn't lived in for three years. The issue in the case was whether the house should be considered his principal residence.

Trisko bought the house in 1941. Except for a couple of years in the army during the war, he lived there with his family until 1948. At that time he was working for the Foreign Service, which offered him a temporary position in Europe. He kept the house because he intended to return to it when he was transferred back to the United States. In the meanwhile, he rented it to a series of tenants who were responsible for maintaining the property. When he and his family returned to the United States in 1951, however, there was a serious housing shortage and strict rent control laws had been imposed near Washington, D.C., where his house was located. As a result, he was legally prevented from terminating the lease and moving back into his house. After four or five months, he sold the house subject to the lease and bought another one nearby. The court held that under the circumstances, even though he had not lived in the house for three years, it was his principal residence at the time of sale.

Cases are represented using a structure much like HYPO's legal case frames: both are based on the "squibs" or case summaries written by law students. My representation includes fields for the official citation, the court, the date of the decision, the facts, and the holdings. The description language I use for the facts is an extension of the temporal logic developed in [Shoham, 1988], modified to incorporate the three modal operators 'want,' 'know,' and 'believe,' as well as the deontic predicates suggested by McCarty [McCarty, 1983]. As a description language, this representation is very similar to McCarty's LLD [McCarty, 1989b], but its inference rules are simpler. A student's summary would also include some discussion of the court's reasoning; instead, I use a very simplified representation, a list of "links." This list indicates whether a proposition or event makes another proposition or event more or less likely. If it is necessary to adapt the facts of a case in constructing a new plan, these links indicate which of the results in the case may be affected.

The representation of the facts of a case corresponds closely to the facts as given in the official case report. It is not exact — for example, the taxpayer in this case owned the house jointly with his wife and was jointly liable for the taxes. For simplicity, the case is represented as if he were the sole owner. In general, however, the goal is to include as much detail as possible.

Suppose for simplicity that *Trisko* is the only case retrieved in our example. In comparing it with the prototype, the relevant dimension is time, the amount of time that the owner has spent away from his first house before the earlier of the sale date and the date of purchase of the second house. In the prototype, this amount of time is zero; in *Trisko*, roughly 3 years.

Next, the system will compare the current situation with the prototype. In this example, the prototype satisfies the input: the prototype plan involves selling the house. No further information was input, so CHIRON assumes that the prototype plan will satisfy the user's requirements. There is no need to consider *Trisko*. CHIRON will instantiate the plan, output it with a citation to §1034, and ask the user if he or she wants any more suggestions.

Suppose the current situation is somewhat different. Suppose, for example, that your client has not been occupying the house. Instead, he has been away for two years in business school. He intended to return to his home at the end of the (two-year) program, but was unable to find a job near his house. This case is stronger than *Trisko* along the time dimension, so the system will construct the §1034 plan and cite *Trisko* as support. It will suggest strengthening the client's reasons for leaving home and failing to move back. Arguably Trisko's justification was stronger than this client's, but CHIRON did not retrieve any unsuccessful case with the same weakness, so the weakness is not fatal. CHIRON will also suggest, as an alternative plan, that the client move back into the house (that is, relax the constraint imposed by the input information).

Finally, suppose that your client has been away from home for six years. If TRISKO is the only case retrieved, CHIRON will extrapolate along the time-away-from-home dimension and suggest the same two plans. On the other hand, if the case base also includes a case where the taxpayer was away from home for five years and lost, instead of adapting the prototype, CHIRON will suggest that the client move back into the house, or if that is not possible, CHIRON will abandon the §1034 plan and look for another one.

# 5 Conclusions

In this paper, I have presented a method for representing and reasoning about open-textured predicates. This method is being implemented in CHIRON, a system I am developing in the domain of United States personal income tax planning [Sanders, 1991].

CHIRON is a planner. It takes as input the representation of a transaction the client intends to perform and (optional) background information, and outputs a plan or plans for performing that transaction with favorable income tax consequences. In order to perform this task, it must represent and reason with the open-textured rules of the income tax law. Its solution to this problem combines rules, prototypes, cases, and dimensions, drawing on ideas from McCarty, Gardner, HYPO, and CABARET. Each of these ideas is found in one or more of the earlier systems; but none of the earlier systems integrates them all.

CHIRON is a hybrid system, combining rule-based and case-based modules. It constructs plans in two stages: first, the rule-based planner constructs a partial plan based on the tax rules. This plan will contain open-textured subplans corresponding to the open-textured portions of the rules on which it is based. Second, the case-based planner takes the partial plan and uses it as the basis for a plan that can be executed by the user. It uses the partial plan to index into its case base, and retrieves a prototype plan corresponding to the partial plan and a set of past cases where taxpayers attempted, successfully or unsuccessfully, to execute similar plans.

After retrieving the prototype and a set of past cases, the case-based planner then tests the prototype plan to determine whether it satisfies the constraints input by the user. If so, it outputs the prototype as a suggested plan. If not, it has three choices: adapt the prototype so that it will satisfy the constraints, relax the violated constraints, or abandon the plan.

The ways in which a prototype plan can be adapted are the *dimensions*; they are stored explicitly in a separate module. The dimensions are suggested by cases; the dimensions along which the cases differ from the prototype are also dimensions along which the prototype can be adapted in constructing a new plan. Cases indicate the dimensions along which the prototype can be adapted; they also limit their extent. Negative cases limit the degree to which a plan can vary from the prototype.

CHIRON also has information about which types of constraints can be relaxed. If the client is not living in his house, for example, CHIRON may advise him to move in; but if the client is 22 years old, CHIRON will not suggest that he become 55.

If the case-based planner succeeds, it outputs a plan annotated with citations to supporting rules and cases. After outputting a plan, CHIRON will query the user to determine whether to continue. If the case-based planner succeeds and the user requests another plan, the case-based planner returns control to the rule-based planner, which then attempts to construct another plan for the same transaction. Similarly, if the case-based planner is forced to abandon a plan, it returns control to the rule-based planner, which continues and attempts to construct another plan. Plans are output roughly

in order of their desirability from a tax point of view, but rather than attempting to construct a single "best" plan, CHIRON leaves the final choice of plan up to the user. The system will halt when the user stops requesting new plans, or when it is unable to find another possible plan.

In CHIRON, the court cases partially define a space of possible plans, with the prototype at the origin. CHIRON's goal is to construct a plan that falls within that space of possibilities. Specifically, it will construct a plan as similar to the prototype as possible. The prototype is a conservative plan, what a tax lawyer would call a "safe harbor." Thus, CHIRON's strategy gives plans a conservative bias, which is consistent with much of tax planning; it could be altered if desired to obtain a more aggressive planner.

## Acknowledgements

This research has been supported by IBM contracts 17290066, 17291066, 17292066, and 17293066, and in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601 with matching funds from IBM, by the Advanced Research Projects Agency of the Department of Defense monitored by the Air Force Office of Scientific Research under Contract No. F49620-88-C-0132, and by the National Science Foundation in conjunction with the Advanced Research Projects Agency of the Department of Defense under Contract No. IRI-8905436.

The author gratefully acknowledges the criticisms and encouragement of Mark Boddy, Karl Branting, Eugene Charniak, Tom Dean, Robert McCartney, Leora Morgenstern, Edwina Rissland, and David Skalak.

## References

[Allen *et al.*, 1990] Allen, James; Hendler, James; and Tate, Austin 1990. *Readings in Planning.* Morgan Kaufman, San Mateo, California.

[Ashley, 1988] Ashley, Kevin D. 1988. Modelling legal argument: reasoning with cases and hypotheticals. Technical Report 88-01, University of Massachusetts, Amherst, Department of Computer and Information Science. (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.

[Gardner, 1987] Gardner, Anne v.d.L. 1987. *An artificial intelligence approach to legal reasoning.* MIT Press, Cambridge, Massachusetts.

[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, 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.

[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.* (to appear.).

[Sanders, 1991] Sanders, Kathryn E. 1991. Planning in an open-textured domain: a thesis proposal. Technical Report 91-08, Brown University.

[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.

[Shoham, 1988] Shoham, Yoav 1988. *Reasoning about change: time and causation from the standpoint of artificial intelligence.* MIT Press, Cambridge, Massachusetts.

[Steels, 1990] Steels, Luc 1990. Components of expertise. *Artificial Intelligence Magazine* 11:30–49.