Report Number: WUCS-89-46

1989-10-01

U.S. Tax Law as an Expert System

Karin M. Hartzell and Susan T. Miles

Tax law is a particular application well-suited for an expert system because of the large among of knowledge and the many variables involved. The following presents the results of implementing a small portion of tax code, with a brief look at whether or not it can be fully recreated in an accurate system with an acceptable response time. The actual prototype implementation has been done using the expert shell NEXPERT, making use of the IF/THEN rule construct which seems to fall naturally from the tax code.

U.S. TAX LAW AS AN EXPERT SYSTEM

Karin M. Hartzell
Susan T. Miles

WUCS-89-46

October 1989

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

Supmitted to the Fourteenth Annual Internaitonal Computer Software and Applications Conference to be held in Chicago, Illinois, October 1990. This project was completed in partial fulfillment of the requirements for the Graduate Certificate in Artificial Intelligence.

U.S. Tax Law as an Expert System

Karin M. Hartzell
Susan T. Miles

Department of Computer Science
Washington University
Saint Louis, Missouri*

ABSTRACT

Tax law is a particular application well-suited for an expert system because of the large amount of knowledge and the many variables involved. The following presents the results of implementing a small portion of tax code, with a brief look at whether or not it can be fully recreated in an accurate system with an acceptable response time. The actual prototype implementation has been done using the expert shell NEXPERT, making use of the IF/THEN rule construct which seems to fall naturally from the tax code.

INTRODUCTION

U.S. tax law as currently put forth by the Internal Revenue Service is quite complex and can be difficult for the average taxpayer to fully understand, especially in light of the continual revisions made to it. More people are having to turn to tax advisors for help in understanding exactly how the laws apply to them [1]. Normally such an advisor answers the questions by using knowledge stored in her memory, turning to abundant tax law books and pamphlets, or, in extreme cases, asking someone more knowledgeable. These methods can be time consuming and are prone to providing incorrect information; in 1988, the I.R.S. telephone assistance helpline was rated 'B+' in accuracy by [2], suggesting that there is room for improvement.

In the past, work has been done on systems which deal with specific angles of the law. Programs exist which solely compute taxes and provide printouts of completed forms. At a more sophisticated level, McCarty's TAXMAN was designed to evaluate the tax implications which might result from a corporate reorganization [3]. TAXADVISOR, which provides estate planning for the client with the goal of increasing client profit, relies upon rules of thumb supplied by experts in the field rather than strict implementation of the law [4]. Still, this is not all-encompassing. As opposed to singling out one subdomain of tax law, the current work being described is a prototype which will assist tax advisors in answering questions concerning all aspects of the tax code, quickly and accurately.

* Correspondence to: S. T. Miles, 9515 Antigo Drive, Saint Louis, Missouri 63123

CONCEPTUALIZATION

Tax law can be represented as a goal-driven IF/THEN rule-based system once some restructuring has taken place. When someone wants to know if he must file a tax return for the year, he asks his advisor, "Must I file?", and this becomes the tax advisor's goal. To establish or reject the goal, she must gather only the pertinent facts from the client. As it currently stands, tax law seems to contain ambiguities. For example, one criterion that a person must meet in order to be considered a dependent is the gross income test which states that his income must be less than $1,950.00. However, according to another section of tax law, a dependent age 65 or older must file a return if his gross income is greater than $3,750.00 [5]. The discrepancy occurs with the term *dependent.* In many instances these alleged ambiguities can be resolved, but only after in-depth research into appropriate Treasury regulations, relevant court cases, and I.R.S. promulgations; in those that are unresolvable, alternate solutions need to be made known [1]. Once the discrepancies have been satisfactorily eliminated, it becomes possible to write all of tax law as IF/THEN rules. For example, **if** a person is a U.S. citizen, single, under age 65, and has a gross income equal to or greater than $4950.00, **then** he must file a return for the tax year in question.

In a prototype, such as this is, it is impractical to implement every tax law and, in turn, have the system be able to respond to every possible question a taxpayer may have. For this reason, the realm of tax law this prototype covers was scoped down. In order to make the system readily expandable, instead of choosing one area of tax law and going into great detail, implementing every law that falls into that category, many areas of tax law were addressed, but without depth. The more general types of questions a person might have, such as "Must I file a return?," are covered, as opposed to very person-specific issues such as "Is my home equity loan fully deductible on my tax return?". In the future it will be easier to add more laws within an area by building on the current existing framework; this is like modular programming in a high level procedural language.

[1] enumerates several criteria necessary for a successful tax advisor expert system. With Neuron Data's expert shell NEXPERT *OBJECT*'s ability to perform forward and backward chaining simultaneously, control the search strategy, list an "audit trail", and tell *how* a fact was deduced and *why* a fact was needed [6,7], NEXPERT meets MacRae's criteria. This, and its development environment, made NEXPERT running on a Macintosh II with 1200K of memory the tool of choice for this prototype.

The user interface needs to contribute to the goal of providing information quickly. Using the system should require minimal computer knowledge on the part of the user, the tax advisor; fact input should require minimal time-consuming typing; the user should be guided through a session; all of which leads to the answer being provided in less time than traditional methods.

IMPLEMENTATION

To begin implementation, various methods for communication between the tax advisor and the system were investigated. For ease-of-use and the desire to minimize typing, a multi-valued menu was chosen to drive the system. The top-level tax topics, such as "Filing Requirements",

"Income", "Standard Deductions", and "Credits" became the main menu. To get to the topic the

FILING REQUIREMENTS
- AGE REQUIREMENTS
- WHO MUST FILE IN GENERAL
- REQUIREMENTS FOR DEPENDENTS

EXEMPTIONS
- PERSONAL EXEMPTIONS
- DEPENDENTS

INCOME
- WAGES, SALARIES, AND OTHER EARNINGS
- INTEREST INCOME
- SOCIAL SECURITY INCOME
- ...
- SALE OF PROPERTY
  - Investment Property
  - Business Property
  - Rental Property
- STOCKS

STANDARD DEDUCTION

CREDITS
- CHILD & DEPENDENT CARE CREDIT
- CREDIT FOR THE ELDERLY OR DISABLED
- ...
- REGULATED INVESTMENT COMPANY CREDIT

Figure 1. Excerpts from the menu hierarchy.

question relates to is simply a matter of following the menu hierarchy (Figure 1). If the question posed to the tax advisor concerns Social Security income, she would select the category "Income" from the main menu. A second menu appears, the result of firing one of the front-end menu rules with the *category* object instantiated to "Income", and the tax advisor then selects "Social Security Income". If at any time in the front-end menu system the user decides she has selected the wrong category, it is possible to backtrack to the previous menu by selecting the NOTKNOWN box. The topic is quickly narrowed and NEXPERT begins inferencing based on the selected goal, requesting any pertinent information to establish or reject this goal. Once the goal has been reached, the answer to the client's question is presented in a text box.

Currently this menu system is implemented inside of NEXPERT. The rules for the top level of the menu hierarchy are tied to the rule conclusion, known as the *hypothesis* in NEXPERT terminology, "task_complete". The system is started and the main menu is displayed when the user begins a *Knowcess*-ing session after *suggesting* the hypothesis "task_complete". Unfortunately this method is ungainly and slows NEXPERT considerably. A future enhancement for the system is an external interface written in C which takes advantage of NEXPERT's callable interface and handlers.

NEXPERT makes use of classes and objects. Every variable referenced in the rules is automatically deemed an object by NEXPERT, however it is also possible to predefine objects with multi-valued attributes, similar to "struct" in C or "record" in Pascal. For instance, a *person* object was created since most questions asked relate in some way to the person asking the question: "Can **I** claim **my** daughter as a dependent?," "Must **my** son file a return?," and so on.

The information the person gives his tax consultant to help her answer his question becomes fields, or properties, in the *person* object, such as his age, marital status, any dependents, and earned income. In the rules, the person's age is then referenced as *person.age* and it will be consistent throughout the rule base; the structures are available for use when more tax laws are implemented.

When attempting to establish a rule conclusion, NEXPERT prompts the user for values needed to confirm facts on the left hand side of a rule in the agenda. These facts, which are objects or object properties, can be of type boolean, integer, float, string, date, time, or special; the objects currently in the system are all of the first four types listed. In providing the value for a boolean-type object, the user is permitted to select one of three available answers: TRUE, FALSE, and NOTKNOWN. For an integer or float, she is given both a blank line on which she can type the value from the keyboard and a NOTKNOWN box. To supply a value for a string, the user has three options: typing the value on the blank line, using the mouse to click the appropriate answer from a list NEXPERT provides, or selecting the NOTKNOWN box. NEXPERT gets the strings in the list by searching the rule base for all rule conditions of the form "IS *object string*" and "ISNOT *object string*", where *object* is the object for which a value is being requested, and *string* is a specific instantiation of that object. If a value is entered which is not known to NEXPERT as a possible value for the string, all rules which require a known value fail. For all object types, NOTKNOWN is always a possible value. When it is chosen as the value for a particular object, NEXPERT assumes that the condition has been satisfied and proceeds as such. This can lead to erroneous results, especially in an area like tax law. If the client says that the value of his gross income is NOTKNOWN and the system assumes then that the condition "*< person.gross_income 400*" is true when in reality the person's gross income is $100,000, the answer the system gives the client may not at all be relevant; for instance, the amount of tax the client owes on $100,000 is expected to be much more than what he would owe if his income were indeed $400, and vice versa; the government considers it to be tax evasion when a person pays taxes on $400 worth of income and lets the other $99,600 go. Unfortunately, there is no way to remove the presence of this NOTKNOWN box, which leaves several options. The selection of that box can be tied to an action, such as setting the value of some variable or bringing up another screen. For instance, in the menu system, the NOTKNOWN box is tied to the previous menu screen. Another approach is to have a text box appear stating that the requested value for the item must be given in order for inferencing to continue, otherwise the session will be terminated.

NEXPERT formats its prompts for object values based on the name of the object. In general, this looks like, "What is the Value of *object.property*?". This can be awkward for the user, especially when the object and property names are unclear. Because the tax advisor is using this system from within the development environment, all the prompts were modified so that they appear in a more natural manner.

CONCLUSION

As discussed above, a very small percentage of tax code has been implemented in this tax advising system. For system verification, two experts from a local tax firm were brought in to experiment. The results were quite positive; they learned the mechanics of the system quickly, repeatedly commented on its ease-of-use, and were impressed with the consistently correct answers.

However, they stressed the fact that tax law is a huge area and that this prototype scrapes only the tip of the iceberg.

Considering the amount of tax law covered, its knowledge base counterpart was huge in comparison, consisting of over 300 rules. This created difficulties in loading and processing. It quickly became necessary to use a dedicated machine, as other processes running in the background prevented enough memory from being available to perform loading. Even on a dedicated machine, though, the response time during a "consulting" session increased noticeably with the increase of rules. The hardware necessary then to store and run a complete system would need to be quite large. Another problem with tax law is its dynamic nature. With the constant modification of laws, those implemented quickly become obsolete. Keeping the system updated as the tax laws change becomes a full-time effort. Politically, a major change in legal attitudes is required to have new laws written in a form which is easily implemented without restructuring.

Although MacRae's criteria for the application of rule-based systems to tax problem solving have been realized in this prototype, the above difficulties indicate that an expert system implementing the complete I.R.S. tax code would require enormous effort at this time.

ACKNOWLEDGEMENTS

The help of S. B. Cousins, Department of Computer Science and Department of Internal Medicine, Washington University, in preparing this paper is gratefully acknowledged.

The assistance of N. Curl and W. Holmes in the development of this system is gratefully acknowledged.

The advice and resources provided by our tax experts is gratefully acknowledged.

REFERENCES

[1] C. D. MacRae, "Tax Problem Solving with an If-then System," *Computing Power and Legal Reasoning*, St. Paul: West Publishing Company, 1985, pp. 595-620.

[2] P. P. Storer, "IRS Telephone Answers Rate a B-Plus," 39 Tax Notes No. 3, 393, 1988.

[3] L. T. McCarty, "Reflections on TAXMAN: An Experiment in AI and Legal Reasoning," *Harvard Law Review*, March 1977, pp. 837-893.

[4] R. Michaelsen and D. Michie, "Expert Systems in Business," *Datamation*, vol. 29, issue 11, November 1983, pp. 240-246.

[5] *Your Federal Income Tax*, Department of the Treasury Internal Revenue Service, Publication 17, Revision November 1988.

[6] *NEXPERT OBJECT Fundamentals*, California: Neuron Data Inc., 1988.

[7] P. Harmon, R. Maus, and W. Morrissey, *Expert Systems Tools and Applications*, New York: John Wiley & Sons, Inc., 1988.