LITES, an intelligent tutoring system for legal problem solving in the domain of Dutch Civil law

Georges Span

University of Limburg

Department of Metajuridica

P.O. Box 616

6200 MD Maastricht

Netherlands

tel. +31 43 883042 / 883020

email: Georges.Span@Metajur.RULimburg.NL

1. INTRODUCTION

In the early 1960s researchers started to develop educational computer programs. These programs can be categorized in: experimental environments [Papert, 1980; Abelson and diSessa, 1981]; games and simulations [Goldstein, 1982]; and computer-assisted instructions (CAI). The CAI-programs try, unlike the first two approaches, to stimulate and control learning process explicitly [Howe, 1973]. The early CAI-programs were not more than electronic "page turners", which printed predefined text, or drill-and-practice programs, printing problems and pre-stored responses [Barr and Atkinson, 1977]. Today's research focuses on the design of intelligent computer assisted learning programs (ICAL), which adapt itself to the students needs and weaknesses. Pioneers in this field are Carbonell and Collins with SCHOLAR, a geography tutor [Carbonell, 1970]; Brown and Burton with SOPHIE, an electronic troubleshooting tutor [Brown, Burton and de Kleer, 1982] as well as BUGGY, a program that can determine student's misconceptions (bugs) about basic arithmetic skills [Brown and Burton, 1978]; and Clancey with GUIDON, a program for teaching diagnostic problem solving [Clancey, 1983, 1984] using the rules of the MYCIN consultation system. However in the field of legal ICAL systems, there is only evidence of a few attempts, each with different goals and objectives. We can mention the work of Haft and Jones on the LEX-tutor, a program which gives students an open ended exploratory learning environment using a natural language dialogue [Haft ea., 1987]. What has become of this ambitious project is unclear [Jones, 1989]. We also got the work of Sherman, constructing a Prolog-program which is able to generate quiz-questions at random on the domain of the Canadian Income Tax Act. The aim of the system is to check whether the students understood the subject matter by giving them a preliminary examination [Sherman, 1989]. Furthermore we can mention the work of Ashley and Aleven on a tutoring system about reasoning with cases [Ashley and Aleven, 1991]. With the help of this system students will be taught how to argue about cases, by proposing arguments derived from other cases for or against a specific solution of a case. All these legal ICAL-programs work differently and have their own designs according to their objectives. For the same reason we developed our own ICAL-system, which is used for learning legal problem solving skills in the domain of Dutch Civil Law. Our program is, as we call it, an Intelligent Tutoring System-shell (ITS-shell) and is named LITES, standing for Legal Intelligent Tutoring/Expert System. LITES can be used as an ITS-developing tool and can be used for running an ITS. We developed the shell and made just one ITS to see whether the concept works [Span, 1992]. In this article we describe the ideas behind and the architecture of our ITS.

2. SOLVING LEGAL CASES

Our main objective is making an ITS for teaching students how to solve cases in a specific domain of law. The domain we implemented in our system deals with the gain of property by 'a third party in good faith'. This doctrine is based upon section 3:86 and 3:87 of the Dutch Civil Law code (BW). This statute became law in 1991 and therefore it incorporates the case law until then, which makes it easier to encode.

For making such a system we first had to develop a theory of how to solve cases in a domain of law. First we have to know what is a legal case? In our opinion a case is a description in natural language of a set of events with legal consequences. These events occur in a certain period of time and sequence. Furthermore in an event we can

distinguish specific actors, objects and actions, which are described in legal terms by some legal doctrine (qualification). So if we want to solve a case we have to know:

- the events, actors, objects and actions of the legal case;
- in what order do events occur;
- what is their legal qualification;
- what legal rules are applicable;
- what is the consequence of these rules;

Normally it is no problem to distinguish the events and the sequence of the events. The most difficult part in legal problem solving is making qualifications and deciding what rules are applicable. As these tasks are domain dependent, we have to make a model of the domain.

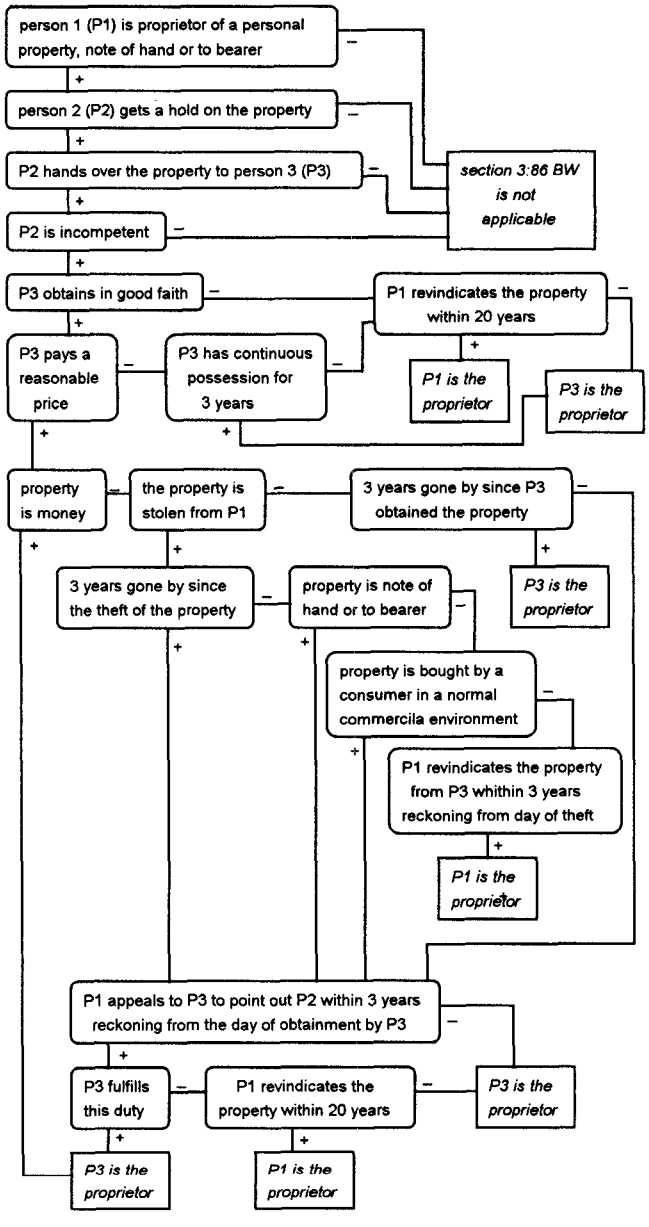
3. THE DOMAIN MODEL

One of the major functions of a tutor for a particular domain should be to communicate the ideal problem-solving structure of that domain [Anderson ea., 1990]. To do so we have analyzed the subject mater and made a schematic representation or flow-chart of the doctrine (figure 1). Such a diagram shows how cases are solved under the doctrine. This model links up with the aim of the system, i.e. providing insight into the structure of the doctrine and the connections between its components. The diagram is the expert interpretation of the doctrine and represents the mental model the expert has of the subject matter [de Kleer and Brown, 1979]. We choose the mental model because we believe that students learn most effectively when they are actively applying their knowledge to solving problems [White and Frederiksen, 1990]. To support such a learning-by-problem-solving the computer has to have a knowledge structure that is complete enough to actually solve the problems itself. Because the diagram shows us which actors, objects, actions and events are relevant and in what order the events have to take place we belief that we can build a learning model on top of this mental model.

So we represented this diagram in a formal knowledge representation language. The knowledge representation language we used is called PL+ and was developed by us for representing legal material in the form of production rules. Because time is essential in legal doctrines the language has the facilities to make time-calculations. The inference mechanism works straightforward, it uses backward chaining, depth first search with a left to right evaluation of the production rules.

One of our main objectives in developing PL+ was that law teachers, who do not have a lot of programming experience can use this language to formalize a legal domain. Therefore PL+ uses Dutch language elements and deals with all user interface problems. The teacher can use two types of rules: inference rules and question rules. If a question rule is used the system puts the question on the screen in the right format and waits for an answer of the student. When the input is restricted or of a certian format the system waits until the right type of input is given. An adventitious circumstance of using such an easy to learn language is that it is not only easy to make knowledge representations but also easy to update them.

Our domain model has however a disadvantage, because the diagram is the interpretation of both statute and case law, it is impossible to see clearly where the underlying rules are based upon. So if we want to justify our model of problem solving we have to connect the law sections with the problem solving rules. This connection will be made in our learning model.



Schematic reproduction of section 3:86 jo 3:87 BW
Figure 1

4. THE LEARNING MODEL

The learning model consists of a set of assumptions about how the student's knowledge state changes after each time the student solves a case within the domain. We believe that each different case within the domain represents a knowledge state. The student has to understand the subject matter to solve a case, so by correctly solving a case we take the view that the student understands the subject matter so far. To make our learning model we must distinguish learning levels in the domain knowledge. In other words we must distinguish the cases in degrees of difficulty in problem solving. If we look at the diagram in figure 1 we can see that this is not easy. We can not value the cases just by looking at the number of inferences made by the computer while solving a case. For instance, the case where person 3 does not pay a reasonable price for the personal property, but still has continuous possession of the property for three years and therefore becomes the proprietor, is easier to solve than the case where person 3 does pay a reasonable price for the property and because the property is money he will be the proprietor of the money. Both cases can be solved in equal inferences but differ in cognitive level. Some cases are more difficult to understand, because they express an exception within the legal domain. Therefore we had to value the cases by hand. With the expertise of teachers, who could point out in the diagram where students normally have great difficulty in understanding, we distinguished 11 learning levels.

Having these learning levels we had to formulate learning tasks to support the learning of legal problem solving. Considering our previous description of what the student has to know for legal problem solving, we formulate the following *learning tasks*:

1. learning what events, actors, objects and actions are of interest for solving a particular case within the domain;
2. learning how the law qualifies these events, actors, objects and actions;
3. learning what legal rules are applicable and why;
4. learning the individual argument steps used for solving a case within the domain;
5. learning the order of these individual argument steps;

At this point we know *what* we want to teach the student, now we must figure out *how* to do this. As we stated before, we want to teach how to solve legal cases within a particular domain of law. This has to be done in a natural way. In a normal learning environment we will try to activate the student's knowledge about the subject matter. We do so by asking questions about the subject matter in order to find out what the student knows already. In our way of exploring the student's state of mind we try to find the boundaries of the student's knowledge about the domain. If we find these boundaries we want to shift them. In fact our primary goal is to constitute progression in the student's knowledge of the domain. We try to imitate this process by giving the student a legal case (of some level) and the means to solve this case. If the student solves the case correctly he will get a new case of a higher level, otherwise he will get some explanatory text and will be confronted with a new case of the same or a lower level. This process will continue until the student solves the case of the highest level. So if a student solves a case correctly we assume the student has at least the same knowledge state for solving this case as the computer has.

Remains the question *'how can the student reveal all his problem solving skills'*? In an answer to this question we developed three didactic strategies:

1. let the student point out his knowledge about what actors, objects and actions are relevant for the case and how they interrelate;
2. let the student label the actors, objects, actions and events in terms of legal qualifications;
3. let the student make all the necessary inference steps for solving the case.

How these didactic strategies work and what knowledge they use will be revealed in the following sections.

4.1. RELEVANCY OF CASE FACTS

To test whether the student knows what actors, objects, actions and events are relevant for solving a particular case, we let the student question the computer about the case. This approach resembles the way lawyers question their clients in order to find out what the case is about. For asking the right questions the student must have knowledge about how certain information influences other information and knowledge about what is to be known to solve a case. However there is a big difference. Unlike the lawyer, the student does know that the case to be solved lays within a specific domain. But there is another difference, the student can't define his own questions, because we can not analyze natural language input. Instead the student can select a question from a set of predefined questions, so formulated that only a student with knowledge about the domain can select the proper question. Moreover the student has to make inferences on the answers to the questions, making some other questions superfluous. For example: when a student knows that person 2 has stolen the property from person 1, the question 'Was person 2 authorized to deliver the property to person 3?' is not a wrong question but the answer is already known by the fact that person 2 stole the property and the system will disapprove the selection of this question. On the other hand when the student knows for instance that person 2 borrowed the property from person 1 then this question is a valid one. So only if you have a clear insight into the subject matter you can choose your questions right.

4.2. QUALIFICATION OF CASE FACTS

To test whether the student can translate a natural language description of a legal case within the domain into legal concepts (qualifications) as used in legal doctrine, jurisprudence and statutes, we let the student answer questions about these qualifications. The student gets a natural language description of the case and the computer asks the student whether a certain qualification is appropriate according to the case-description. To answer this question the user has to know what is meant by the qualification. If he does not know this the system provides him with additional information in which the terminology is placed in a broader context and he will learn what is meant by this qualification according to the doctrine, law and jurisprudence. After reading this text he should be able to answer the question. If the student makes a mistake he will get case-sensitive help which makes references to the sections of the law. After qualifying all case facts correctly, we can assume that any error in solving a case will come from not knowing how to apply the rules.

4.3. SOLVING A CASE STEP BY STEP

Solving a case step by step gives the student the opportunity to show whether he can solve the problem correctly. The student can select its problem solving steps out of a predefined list of possible steps. The student has not only the task of selecting a valid step but also, where this is important, to select the steps in the right order. For instance imagine a case where person 2 stole money from person 1 and bought with this money something from person 3. Now the student has to select the argument that the property is money instead of the argument that the property is stolen, because if the property is money the fact that the property is stolen is not relevant anymore for solving the case (see figure 1). In this way the student has to show that he completely understands how the case facts are related to each other and what problem-solving steps can be made. If the student makes an error the system will give him a new opportunity to select the right step or otherwise he will be shown the right answer and the reason why it is the right answer (this includes mentioning and explaining the underlying legal rules).

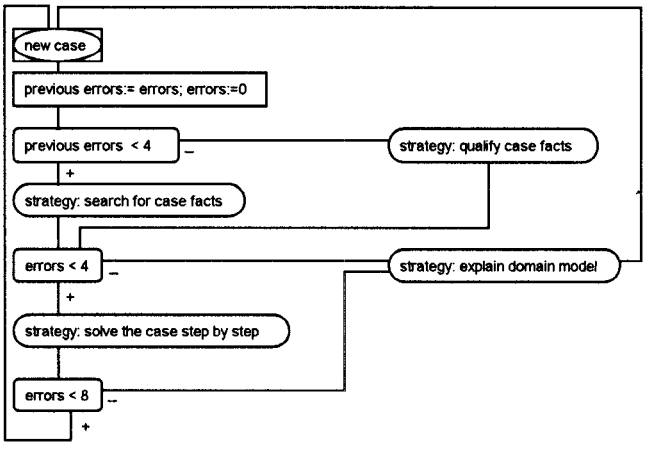
5. THE STUDENT MODEL

If a student makes a mistake it will be corrected immediately by means of explanatory texts. But this is not enough. The system has to be sure the student understood the correction and can apply this newly gained knowledge. Therefor the system has to check whether or not the student makes the same mistake again. To do this we keep track of the places where the student made mistakes while working through the three didactic modules. These places can be typified in terms of legal concepts. So eventually we end up with the following data:

1. The level of the cases the student has tried to solve;
2. Per case the amount of errors made by the student, split up to the didactic module they were made in;
3. Per case the amount of errors made by the student expressed in terms of legal concepts.

On basis of these data the system will decide of what level the new case will be, and on what points it has to vary case facts according to the previous case. Deciding the new level of the case is done by a very simple algorithm taking into account the total amount of errors during the solving of the previous case. If the student made no mistakes the case level will increase by 2, if the student made less than 5 mistakes the case level will increase by 1, otherwise if the student made less than 9 mistakes the level will decrease by 1 and if the student made more than 8 mistakes in solving a case the case level will be 1. When the case level is calculated we can vary in the case facts according to the errors made in the previous case. So for instance when in the previous case the student made an error concerning the nature of the personal property, we will change this nature in the next case (if this is supported by the new case level).

Not only does the system decide of what level the new case will be, but it will also decide on basis of these data what didactic strategies will be used (figure 2). Normally one session in which the student is presented a case the student will be confronted with two didactic strategies. The first strategy depends on the errors the student made during the previous session and can be either searching for case facts (less than 4 errors) or qualifying case facts (4 or more errors). The second strategy will always be solving the case step by step. If the student makes more than 4 errors during the first strategy, it is obvious that he has too little knowledge of the subject matter, and therefore the system presents a text in which the subject matter is explained in detail, after which the student can try again.



Decision model of what didactic strategy to use
Figure 2

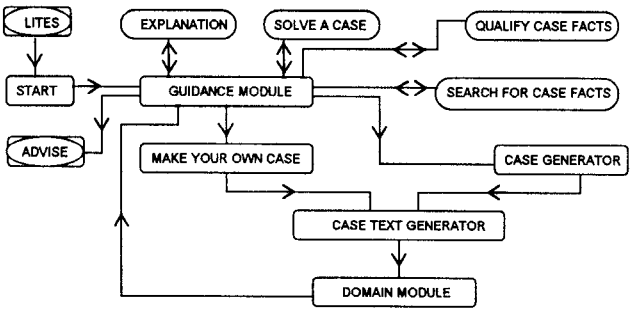
These data form our *student model* and give insight into:

1. the student's level of experience;
2. the student's knowledge progression;
3. what didactic strategies the student masters;
4. the legal concepts the student has insufficient knowledge of.

When the student has reached the highest level of experience or has solved 10 cases the system will evaluate the student's performance. First the system gives an overview of the level of the cases the student has mastered. In this overview the student can see what progress he made. Then the system gives an overview per case of the amount of mistakes the student has made (divided into didactic strategies). Furthermore the system gives an overview of the concept-related errors per case and gives an overview of the concepts that need more attention in the future.

6. THE INTEGRATION OF COMPONENTS

We have seen the basic components of our tutoring system. Now it is time to show how these components interrelate. In the previous section we saw that the decision on what didactic strategy is used, is based on a simple algorithm. This algorithm is incorporated in a knowledge base that gives guidance to the system. We call this the *guidance module*. In fact, the whole system is build out of knowledge modules. The primary module is the *domain module*, all didactic modules are based upon the knowledge of this domain module. In the *didactic modules* we added more domain and task specific knowledge and the knowledge in these modules is structured according to these tasks. Furthermore we had to design a *case generator* and a *case text generator*. These generators work straightforward. On bases of the level of a case the case generator knows what concepts are of interest and with regard to the errors the student made in the previous session it generates a composition of facts which form the case (the case facts). According to these case facts, the case text generator dynamically derives the proper descriptions in natural language of the actors, actions, objects and events. So every case-description has more or less the same structure but they differ on the described facts. All these modules work separately, they only exchange data. A schematic overview of the system is given in figure 3. In this figure the LITES-box stands for the point where the student can choose the domain. The START-box gives the student the opportunity to identify him- or herself, so the system can use the student history. If the student is unknown to the system a new student profile is made. In the MAKE YOUR OWN CASE-box the student can define his own case, he wants to solve. The GUIDANCE MODULE-box is the center of the system. At this point the system updates the student profile, selects the new didactic strategy, calculates the level of a new case, writes data to files and initiates knowledge modules who use these data files. Most knowledge modules activate, when they are finished, the guidance module. In the ADVISE-box the system computes the advise as described above and this forms at the same time the end of a learning session. The other modules speak for itself or are described above.



The integration of components in LITES
Figure 3

7. CONCLUSIONS

We constructed an ITS for learning legal problem solving in the domain of Dutch Civil Law. This ITS has a modular structure, which is of an advantage because it can be extended with other modules. On the other hand because these modules work more or less autonomous they have to be updated separately when the law changes, which is just more work. Our philosophy of building an ITS for legal problem solving is that we have to construct an image of how experts solve problems in a specific domain. This so called domain model forms the basis of our learning model and its didactic strategies. The actors, objects, actions and events as described by our domain model form also the basis of the student model. All the student's errors will be typified in terms of legal concepts and are recorded according to the case and the didactic module they are made in. Because of this we can adapt not only the level of teaching, that is the level of the case that has to be solved, but also the case facts within the new case and the didactic strategy the system will use. Moreover we can identify the problem areas in the student's conception of the domain. Our tutoring system is not extensively tested yet but as far as students worked with the system they found it revealing. However there has to be done some fine tuning. Eventually remains the question whether this approach is suitable for other domains. We believe as this domain does not differ in essence of other legal domains that LITES is suitable for constructing other ITSs, according to the same structure. Nevertheless this new ITS has to be designed from scratch, one can however use the techniques of the previous ITS to a certain extend but most of the work has to be done again. This is a major disadvantage of our system. On the other hand the fact that our system works on personal computers with the MS-DOS disk operating system and is easy to use (or learn) brings the making of ITSs within the reach of legal teachers. Besides the students can use these systems at home and are less dependent on the facilities of the universities anymore.

ACKNOWLEDGEMENTS

The research for this paper was partly financed by the Foundation for Knowledge Based Systems (SKBS), which seeks to improve the level of expertise in the Netherlands in the field of knowledge based systems, and to promote the transfer of knowledge in this field between universities and business companies.

REFERENCES

Abelson, H., diSessa, A. (1981): *Turtle geometry: The computer as a medium for exploring mathematics.* MIT Press, Cambridge, Mass.

Anderson, J.R., Boyle, C.F., Corbett, A.T., Lewis, M.W. (1990): Cognitive Modelling and Intelligent Tutoring. *Artificial Intelligence 42*, p. 7-49.

Ashley, K.D., Aleven, V.: Towards an Intelligent Tutoring System for teaching law students to argue with cases, in *Proceedings of the Third International Conference on Artificial Intelligence & Law*, Oxford, England, p. 42-52.

Barr, A., Atkinson, R.C. (1977): Adaptive instructional strategies, in: H. Spada and W.F. Kempf (Eds.), *Structural models of thinking and learning.* Hans Huber, Bern.

Brown, J.S., Burton, R.R., de Kleer, J.: Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II, III; in D. Sleeman and J.S. Brown (Eds.), *Intelligent Tutoring Systems.* Academic Press, London, p. 227-282.

Carbonell, J.R. (1970): AI in CAI: An Artificial Intelligence approach to computerassisted instruction. *IEEE Transactions on Man-Machine Systems*, Vol. 11, p. 190-202.

Clancey, W.J. (1983): GUIDON, *Journal of Computer-Based Instruction,* Vol. 10, p. 8-15.

Clancey, W.J. (1984): Use of Mycin Rules for Tutoring; in: B.G. Buchanan and E.H. Shortliffe (Eds), *Rule Based Expert Systems*, Addision Wesley.

Goldstein, I.P. (1982): The genetic graph: a representation for the evolution of procedural knowledge, in D. Sleeman and J.S. Brown (Eds.), *Intelligent Tutoring Systems.* Academic Press, London, p. 51-77.

Haft, F., Jones, R.P., Wetter, TH. (1987): A natural Language Based Legal Expert System for Consultation and Tutoring - The LEX Project, in proceedings of *The First International Conference on Artificial Intelligence and Law*, Boston, Massachusetts, p. 75-83.

Howe, J.A.M. (1973): Individualizing computer assisted instruction, in: A. Elithorn and D. Jones (Eds.), *Artificial and human thinking*, Elsevier, Amsterdam. p. 94-101.

Jones, R.P. (1989): Knowledge bases for computer assisted legal instruction, in *Preproceedings of the III international conference on logica, informatica diritto*, Legal Expert systems, Vol. 1, Florence, p. 359-379.

de Kleer, J., Brown, J.S. (1983): Assumptions and ambiguities in mechanistic mental models, in: D. Gentner and A. Stevens (Eds.), *Mental Models.* Erlbaum, Hillsdale, NJ.

Papert, S. (1980): *Mindstorms: Childeren, computers, and powerful ideas.* Basic Books, New York.

Sherman, D.M. (1987): Expert Systems and ICAI in Tax Law: Killing Two Birds with One AI Stone, in *Proceedings of the Second International Conference on Artificial Intelligence and Law*, Vancouver, BC Canada, p. 74-80.

Span, G.P.J. (1992): *LITES: een intelligent tutorsysteem voor juridisch onderwijs.* Ph.D.-thesis. University Press Maastricht.

White, B.Y., Frederiksen, J.R. (1990): Causal Model Progressions as a Foundation for Intelligent Learning Environments. *Artificial Intelligence* 42, p. 99-157.