Legal knowledge based systems
JURIX '95
Telecommunication and AI & Law

The Foundation for Legal Knowledge Systems
Editors:
J.C. Hage
T.J.M. Bench-Capon
M.J. Cohen
H.J. van den Herik

F. Centinia, T. Routen, A. Hartmann and C. Hegarty, STATUTOR: Too intelligent by half?, in: J.C. Hage, T.J.M. Bench-Capon, M.J. Cohen, H.J. van den Herik (eds.), *Legal knowledge based systems JURIX '95: Telecommunication and AI & Law*, Lelystad: Koninklijke Vermande, 1995, 121-132, ISBN 90 5458 252 9.

More information about the JURIX foundation and its activities can be obtained by contacting the JURIX secretariat:

Mr. C.N.J. de Vey Mestdagh
University of Groningen, Faculty of Law
Oude Kijk in 't Jatstraat 26
P.O. Box 716
9700 AS Groningen
Tel: +31 50 3635790/5433
Fax: +31 50 3635603
Email: sesam@rechten.rug.nl

 http://jurix.bsk.utwente.nl/

STATUTOR: TOO INTELLIGENT BY HALF?

F. Centinia*, T. Routen*, A. Hartmann* and C. Hegarty‡
**Department of Computer Science, ‡School of Law*
De Montfort University, The Gateway, Leicester LE1 9BH, United Kingdom
Tel: 0116 2551551 Ext. 8495, fax: 0116 2541891, email: twr@dmu.ac.uk

Abstract

This paper describes some of the issues that have arisen from the development of an Intelligent Tutoring System to educate students in the statute-law domain. The originality of the system consists of a graphical environment, in which the student can represent valid legal arguments by constructing complex graphical structures. A brief description of the system is given, mainly the graphical environment or tutorial module, the authoring module and the expert-system module. The three modules share the same knowledge base, that is the rule-based representation of statute law, providing an interesting example of reusability of declarative knowledge. The rule-based representation of the statute is discussed, with the main difficulties encountered in catching the real meaning of the statute, while at the same time providing a suitable representation format for the graphical display. The system provides assistance to the student both during the process of constructing an argument and at the end of the exercise; the latter consisting of the overall assessment of the student performance for the given exercise, and the display of the correct answer as generated by the system. This assessment is supported by a dynamic student-modelling approach, which is based on the comparison of the student's proof-tree and the correct proof-tree. The results of an interim evaluation of the system are also provided in this paper.

1 Introduction

Intelligent Tutoring Systems (ITSs) are computer programs which are designed to provide students with individualised, dedicated tutoring. This can only be achieved if ITSs know who they teach, what they teach and how to teach it. These three main types of knowledge are traditionally referred to in ITSs systems as the expert knowledge, the student diagnostic knowledge and the instructional or curricular knowledge (Burns and Capps, 1988). ITSs involve artificial-intelligence techniques, such as knowledge representation, problem-solving approaches, dynamic student modelling, human cognition, intelligent user interfaces (Frasson and Gauthier, 1990). ITSs need to be generalised from applications too closely linked to a particular domain and moved toward general purpose tools (Lawler and Yazdani, 1987). Those general purpose tools for tutoring, or ITS *shells*, are intended to be applicable to teach different domains in the way an expert system shell is used to implement expert systems in different domains. The use of an ITS shell provides several advantages. A practical advantage is that the user of the shell will have only to implement a knowledge base for the new domain and not a complete new system (Sleeman, 1987). A second, theoretical advantage is that the main knowledge of the shell can be oriented toward general theories and strategies of teaching, at the same time providing a good means for testing their generality in different domains. An important role in ITSs is played by the instructional environment. The instructional environment defines the kinds of problems the student is to solve and the tools available for solving them. The environment module may make explicit properties of the domain that were previously hidden or implicit (Burton, 1988). For example, the graphical display in the Geometry Tutor shows that geometry proofs are not linear, but tree-structured (Anderson *et al.*, 1985).

(Routen, 1992) described a prototype ITS shell called STATUTOR (Statute-Tutor) since it was designed to support, amongst other things, the reuse of formalisations of statute law in the presentation of exercises analogous to the familiar case analysis exercises of traditional legal education. In 1994, the Joint Information Systems Committee of the UK Higher Education Funding Councils, under its New Technologies Initiative, awarded a support for a two-year project aimed at developing the prototype and producing a finished system which could be made available to law schools. This paper reports on this project, identifying some of the problems which arose and some of the issues raised by the development.

2 Realistic knowledge base

STATUTOR attempts to educate students by requiring of them active learning in the construction of complex structures, such as proof-trees representing a simple kind of legal argument. Students are asked to demonstrate their understanding of the logical structure of a piece of legislation by constructing an argument which shows how the legal consequence of a number of *case facts* comes about. The argument consists of a number of inferences, each inference represented by a graphical link from a number of facts to a conclusion, which in turn may be then linked to establish further conclusions. The result of this process is a graphical tree-like structure which serves as a proof-tree for the conclusion.

STATUTOR was developed and demonstrated with only toy knowledge bases, and one aim of this project was to provide a realistic knowledge base and associated exercises to enable an effective evaluation of the potential of the system within the law curriculum. To this end, in collaboration with a colleague in the school of law, the Data Protection Act 1984 (sections 1 and 21), was chosen to form the basis for the development of a knowledge base which seeks to reflect the provisions of this law.

In developing the realistic knowledge base, several problems came to light. Firstly, it became apparent that the system had to provide the flexibility for negated conditions to appear in arguments. Secondly, it also became apparent that, although the main idea behind STATUTOR was to treat the system's ability to construct a proof-tree as analogous to answering the case analysis exercise, students could not be expected to provide their answer to the level of detail necessary for the system. That is, the system should not require of students that they present 'commonsense' calculations such as '1992 > 1990' as part of their arguments. Thirdly, we noted that it is possible that an argument may want the same condition to play a role in establishing more than one intermediate conclusion, which would necessitate the student duplicating part of his answer. This problem had to be solved.

2.1 Negative facts

In Figure 1 we can see a typical exercise as it is presented to the student. The conclusion the student is required to prove is shown in a box at the centre top of the figure (text-box in bold characters). The student needs to select from, and connect in the appropriate way, the conditions shown in the set of boxes lined underneath the conclusion. These conditions can be of three types: *facts* (text-boxes in underlined characters), *negated facts* (as facts but with a NOT flag), and *normal conditions* which have to be proved by using facts, negative facts or indeed other normal conditions. (The real system makes full use of colours to distinguish among the different types of conditions). There are also *red herrings*, that is conditions which are not required in the specific argument. The student constructs his argument by using the tools on the left hand side of the figure. The open book icon represents a *source tool*, this tool is used to label each inference in the proof-tree with the portion of statute which warrants that particular inference.

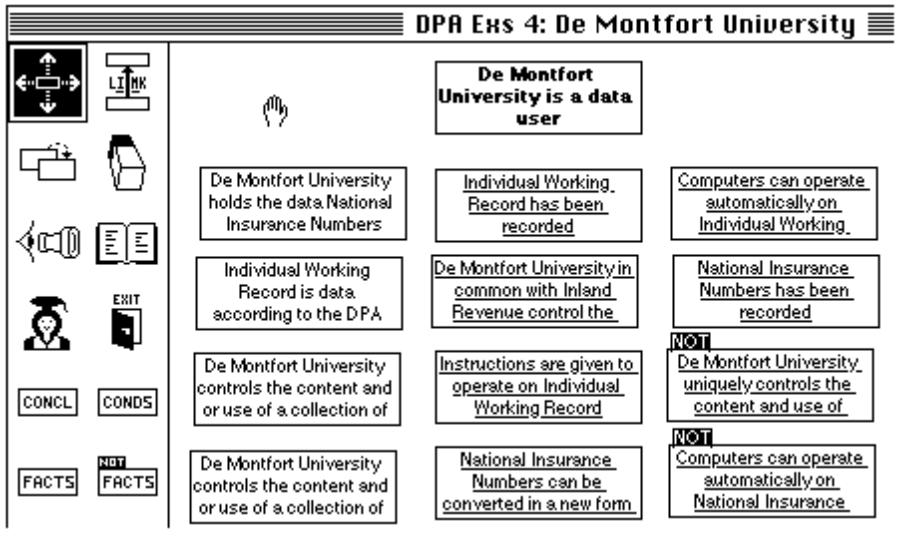

Figure 1: The user interface of a typical exercise

In Figure 2, the student has drawn two inferences in order to prove the conclusion. In the bottom-level inference the student states that three conditions are required to prove the intermediate condition, and that this rule originates from the Data Protection Act, section 1, subsection (5), paragraph (h) and subparagraph (ii). In the top-level inference the student believes that a single condition is required to prove the conclusion, and that according to DPA 1(5).

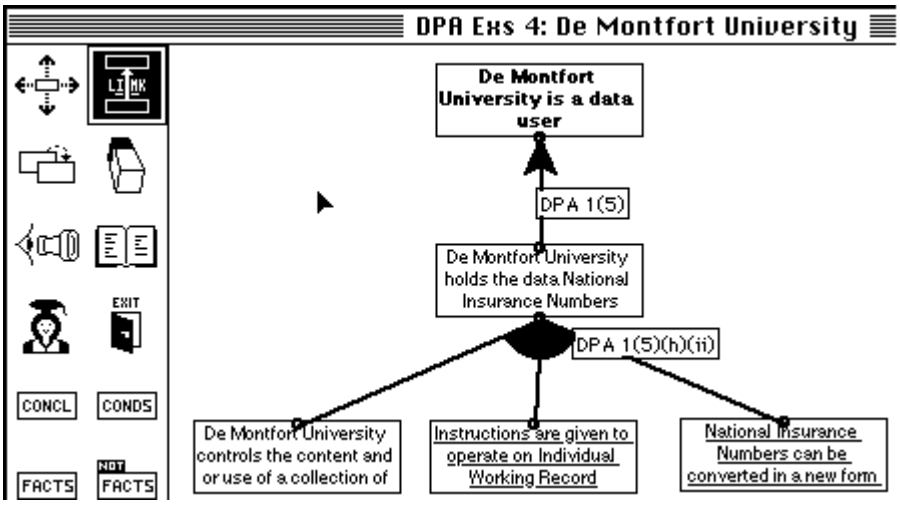

Figure 2: Results of interaction with a student

2.2 Evaluable conditions

Here follows the representation of *data holder* as defined in DPA 1(5)(c):

```
srule(['DPA', '1', '(5)', '(c)'],
   holds_data(USER, DATA),
   [                                          /*rule conclusion*/
      controls_content_and_or_use_of(USER, DATA),
                                              /*normal condition*/
      convertible(DATA, CONVERTED_DATA),           /*fact*/
      different(DATA, CONVERTED_DATA),        /*evaluable condition*/
      data(CONVERTED_DATA)]).                      /*normal
   condition*/
```

The above rule contains two *normal conditions*, which are conditions requiring to be proved themselves, a *fact*, which can only be negated or asserted, and an *evaluable condition. Evaluable conditions* are conditions that are evaluated internally by the system. Students do not have to provide them in their construction of the legal argument, though they must make sure that the remaining conditions are such that the *evaluable conditions* are satisfied. The *evaluable conditions* are required in two main circumstances. One is when the calculations involved are too complex, and we do not want the student to perform them but only to concentrate on the conditions for their successful termination. The other circumstance is when the facts stated in the *evaluable conditions* are obvious from considering the other conditions, and therefore would be quite futile questions to be asked to the student. *Evaluable conditions* are represented as small circles in the correct proof-tree provided by the system (Figure 3).

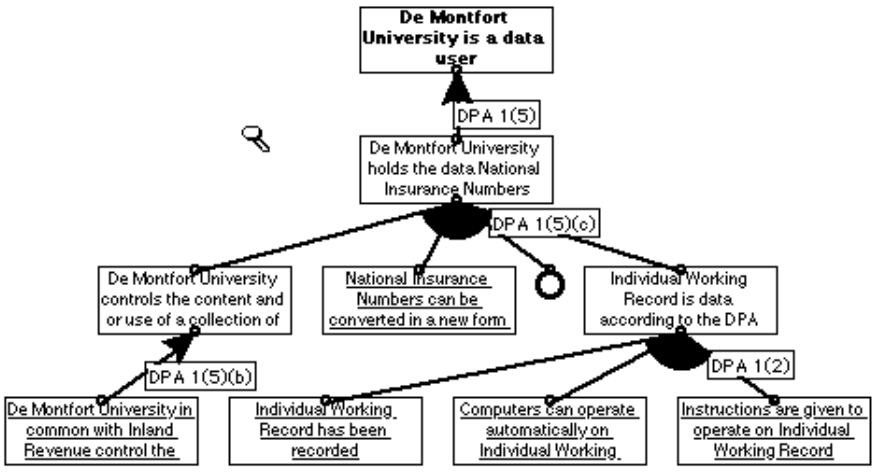

Figure 3: A correct proof-tree with some evaluable condition

Students can click on the small circle to see the content of the *evaluable condition* , which in this case is clearly an obvious derivation of the other conditions (Figure 4).

In Figure 3 we can see the graphical representation of the rule just described. This rule is represented between the second and third level in the proof-tree structure.

Students can also access the on-line source text of the statute by clicking, in the proof-tree provided by the system, on the labels referring to the applied rules. Clicking on the label DPA 1(5)(c) in Figure 3 for example, will present the following section of the statute, with paragraph (c) highlighted, which represents the piece of legislation formalised by the srule(['DPA', '1', '(5)', '(c)'], _) previously shown.

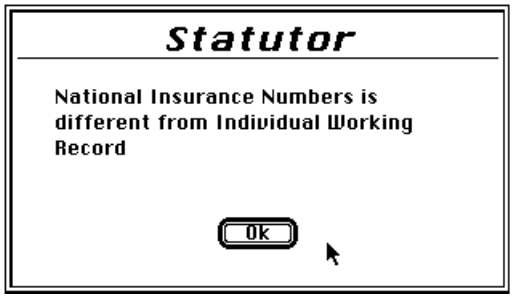

Figure 4: A sample evaluable condition

```
(5) "Data user" means a person who holds data, and a person "holds"
data if —
(a) the data form part of a collection of data processed or
    intended to be processed by or on behalf of that person as
    mentioned in subsection (2) above; and
(b) that person (either alone or jointly or in common with other
    persons) controls the contents and use of the data comprised in
    the collection; and
(c) the data are in the form in which they have been or are
    intended to be processed as mentioned in paragraph (a) above or
    (though not for the time being in that form) in a form into
    which they have been converted after being so processed and
    with a view to being further so processed on a subsequent
    occasion.
```

2.3 Duplication of sub-trees

One of the problems caused by a fully modular rule-based representation is that it may cause the same rule to be expanded several times in the same proof-tree. In fact, if each rule is complete on its own, it means that almost each of them will require the basic definitions to be proved. In the DPA this can be for example the *Definition of data.* Therefore if a higher level rule calls a number of rules most of which contain the same *Definition of data,* the student will be faced with the tedious task of repeating the same proof for *Definition of data* several times. This problem can be solved by requiring the student to supply the proof for the same statement only once, leaving the system to check that this has been done at least once in the overall proof-tree. Another solution to this problem is provided in the curriculum strategy (see dynamic student modelling and curriculum). Negative conditions in the rule-based representation are explicitly represented in the graphical display, so that the rule can be represented in the graphical display in its completeness. At the moment we have dealt only with negated facts, which can be interpreted either as negation by failure (we do not know whether they are true or false) according to the closed world assumption, or as facts that have effectively been negated. The system is to include the possibility of representing negative conditions when they refer to a conclusion of another rule (normal conditions).

3 Interface developments

3.1 The forwards only exercise dialogue

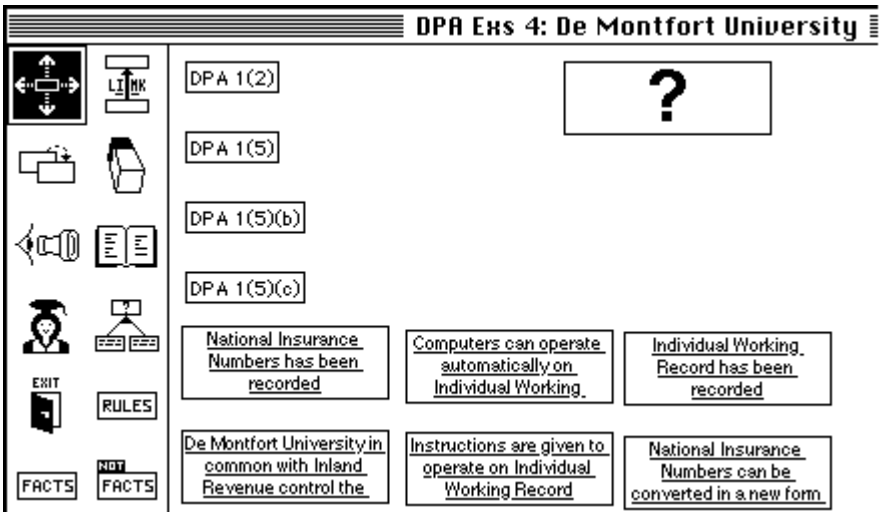

Figure 5: A sample of a forwards reasoning dialogue

In the exercise dialogue seen in Figure 1, the student can proceed in both forwards and backwards reasoning, as he is given all facts, conditions and conclusion, which he is asked to link appropriately in the construction of the proof-tree. In the following dialogue the student cannot reason backwards. The student is provided with only facts and a set of rules (Figure 5).

The student links a set of facts and their appropriate rule to the unknown conclusion, and then asks the system to produce the conclusion (Figure 6). If the student's inference is correct, the unknown conclusion will be changed into the conclusion derived by the inference.

Once a correct inference is made, a new unknown conclusion is generated by the system, so that the student can continue the forward-reasoning process until the final conclusion is reached (Figure 7). When the final conclusion is reached the unknown

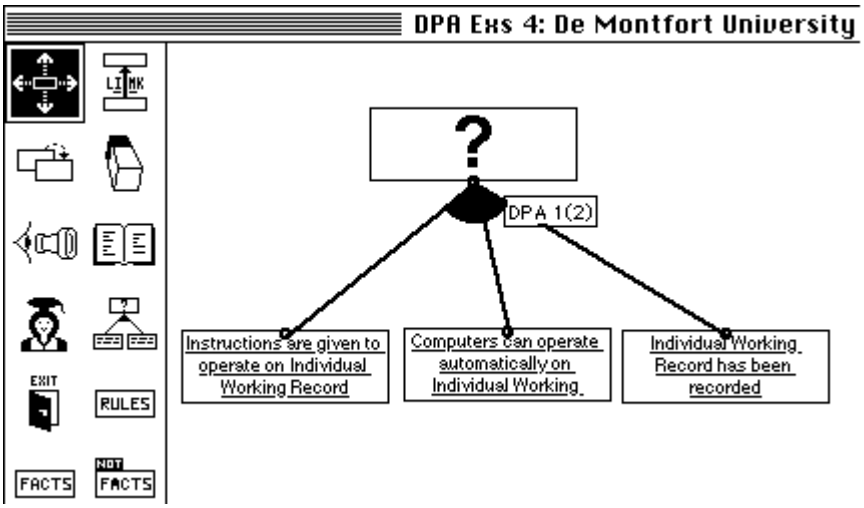

Figure 6: How the system may produce a conclusion

conclusion is not generated, meaning that the exercise was successfully completed.

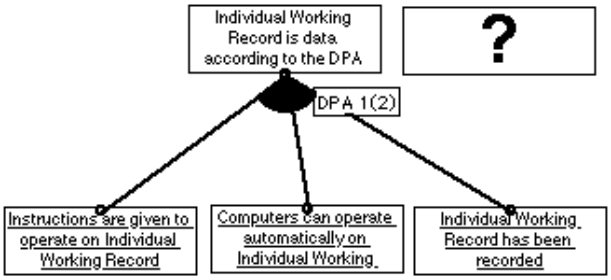

Figure 7: Repeating the final conclusion

3.2 Node feedback

The node feedback tool provides students with immediate help when they find themselves in difficulties. If students are not certain of their proof regarding the intermediate node, they may ask for some help or hint on how to prove that particular node. The feedback is based on each singular node or rules to solve that particular node Feedback based on providing the premises, conclusion and applied-rule (the reference to the portion of STATUTOR applied) for each rule appears to be the most effective for the student (McKendree, 1990).

The variety of different kinds of feedback available by the student includes referencing the primary source material (the statute); the rules which apply in the construction of the particular node, in addition to an immediate, graphical diagnosis of their current answer. The graphical feedback on the particular node would look as in Figure 8, where students are hinted that two of the conditions they provided are correct, one condition is wrong (double crossed box), and one condition is missing (shadowed box). A diagnosis is also made on the label used to refer to the statute applied, by using symbolic characters in place of Wrong, Missing or Surplus references.

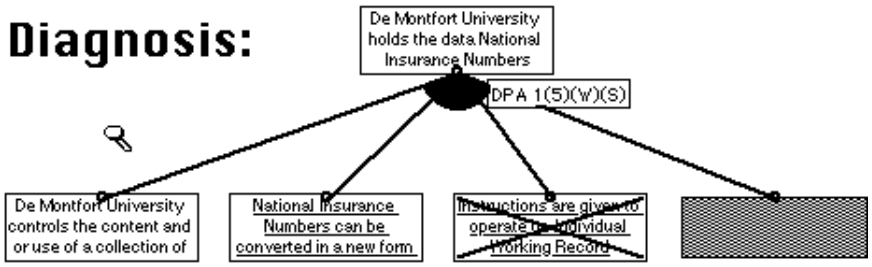

Figure 8: Four conditions distinguished by class

The reference diagnosis can be seen more explicitly by clicking on the symbolised diagnosis, as in Figure 9.

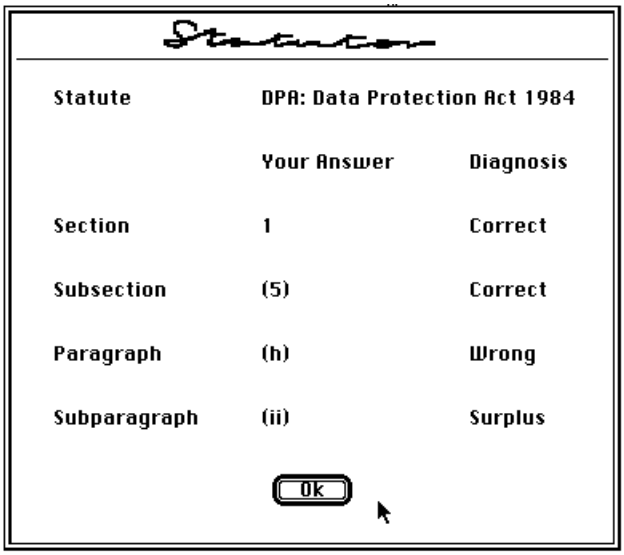

Figure 9: The references, answers and diagnosis

The feedback described so far is student-controlled feedback. The student decides if and when feedback is required. The system also provides some kind of system-controlled feedback. For example, the student is informed of this mistake when attempting to justify a fact by pointing an arrow link toward it (a fact does not require to be justified).

In fact, by recording the various diagnoses made during the student's problem-solving activity, the system can evaluate the student not only on the basis of the present state of the exercise, but considering also the process through which the present state was reached. The mechanism also provides a good means of controlling the type of feedback we can provide to the student. In contrast with immediate feedback (Anderson *et al.*, 1990) where the system intervenes after each error made by the student, and final feedback, where feedback is given only at the end of the exercise without considering the intermediate steps, this mechanism gives us the possibility of intervening at any stage of the student's problem-solving activity, and allowing the student to make, for a given situation, a more significant number of errors before intervening.

Students requiring a node feedback show some lack of knowledge on the rule used in that particular node, and according to the degree of assistance they require we can make assumptions about how much they know about the applicability of that particular rule and therefore of the portion of statute associated with it. By requiring assistance students have nevertheless shown some doubts about their knowledge of the domain, which reduces the degree of error in the final proof-tree. This kind of student diagnosis can be used in selecting the next exercises to be given to the student and indeed the format in which these exercises should be presented. Students lack of knowledge of the use of a particular rule will lead to exercises where that rule will be likely to reappear. On the other hand when the student appears to have mastered the use of a particular rule, the following exercises may not deal with the same rule, or, if they do, the rules will not be required to be expanded (the conclusion of the rule may be provided as a fact). Collapsing the expansion of a mastered rule into a simple fact can be done either dynamically or by the human tutor when using the authoring mode. In fact the human tutor can decide before constructing the new exercise which rules are to be expanded and which are not. Another beneficial result of collapsing mastered rules into simple facts lies in avoiding the

repetition of the same rule expansion within a single proof-tree, as it is likely that the most mastered rules are the ones at the lowest level.

4 Evaluation

A formative evaluation of the system has been recently performed whereby a small number of subjects were guided in a rather formal manner through exercises with the system, and subsequently questioned about their impressions and ideas concerning the applicability of the system. Furthermore, the system was shown to a number of law tutors for the purpose of eliciting in an informal manner perspectives of experienced educators.

The general impression of all the subject trials was fairly consistent. All test subjects understood the presentation of the legal argument with the help of the graphical representation used. They enjoyed using the system and developed new perspectives how to structure arguments logically . Therefore it can be concluded that the graphical presentation chosen is a useful and successful one. All the subjects considered the system to be useful as an introduction to statute law and the Data Protection Act, supported and supervised by a human tutor, but restricted to an introductory stage.

The subjects managed to complete the given exercises in a reasonable time. Subjects with no previous exposure to the Data Protection Act concentrated on the graphical feedback and solved the exercises mainly with graphical help, without consulting the statute text. The students who had previous experience with the Data Protection Act attempted to solve as much as possible of the exercises by relying solely on their previous knowledge and the statute text. These students consulted the graphical feedback only to verify and correct their solution. This supports the conclusion that the students should be given an introduction into the statute text prior to the exercise. The system would help them further to understand the statute law, rather than give them an introduction into it. The graphical diagnosis as well as the graphical correct solution were understood very well by the subjects and they enjoyed using them. They liked the graphical presentation to such an extent that most of the students would like to refer to a similar tool allowing them to freely construct their own exercises. In summary, students' logical perception of the arguments actually improved, and students not previously exposed to the domain managed to solve the paper-exercise given to them after the trials surprisingly well.

The main criticism of the system, as expressed by both students and tutor, concerned the rule-based representation of the statute law. Students did not make much use of, and did not understand, the feedback providing them with the rule-format of the node they were attempting to construct. Students and tutors would like to see the argument's text being phrased in a more articulate way, and not being restricted to the rigid format required by the expert system. Furthermore, both students and teachers found the domain too limited as it is restricted to only two sections of the DPA.

During these evaluations, we came upon a valuable but paradoxical insight: that by making the system simpler, it could be made more useful. We are used to expecting that injecting a little intelligence into the system will enhance its capabilities, but in the case of the present system, for the domain of statute law at least, it may well be that it is possible to decompose the elements of the system which law tutors have responded positively to, from those elements of the system which seem to restrict its applicability. In short, as long as we maintain the interface but dispense with the knowledge-based aspects, then we may well be able to have more impact in presenting technological assistance for law tutors.

As a result of this insight, we are investigating the possibility of a new system, implemented in Visual Basic, which would apply the same principles as the current system, but without requiring a rule-based representation of the law domain. In the new system, the authoring tool permits the human tutor freely to edit both the argument's texts and structure. In brief, the tutor will construct his own graphical proof-tree, editing

the text boxes and structuring the argument as he wishes. The only drawback of this approach is that the human tutor will not be provided with the argument structure, but will have to construct it himself. Much of the functionality of the system as discussed in this paper, such as feedback, tree-comparison and tutorial action, can then be applied by using the graphical argument structure as constructed by the human tutor. With this approach human tutors can build structured arguments related to any piece of law, importantly including case law, without the requirements of the law having been previously formalised.

5 Conclusion

This project was predicated on the idea that the ITS shell is of interest to law tutors, but it was also predicated on the idea that it could be of interest in other subject domains too. The reason why the project existed in the first place was because of positive responses from law tutors to presentations of STATUTOR. However, the hypothesis now under consideration is that those positive responses were not related to the fact that the system could answer its own questions and had an expert system component too, but rather that they were related to the interaction which the system provided. We hope to be in a position in the near future, with two well-developed systems, one knowledge-based and the other based on a much simpler technology, to be able to provide a conclusive answer to this question.

References

Anderson, J.R., Boyle, C.F., and Yost, G. (1985). The Geometry Tutor. *Proceedings of International Joint Conference on Artificial Intelligence.*

Anderson, J.R., Patterson, E.G., and Corbett, A.T. (1990). Student Modelling and Tutoring Flexibility in the Lisp Intelligent Tutoring System. *Intelligent Tutoring Systems (At the Cross-roads of Artificial Intelligence and Education)* (eds. C. Frasson and G. Gauthier). Ablex Publishing Corporation, Norwood, New Jersey.

Burns, H.L., and Capps, C.G. (1988). Foundations of Intelligent Tutoring Systems: An Introduction. *Foundations of Intelligent Tutoring Systems* (eds. M.C. Polson and J.J. Richardson). Lawrence Erlbaum Associates, Hillsdale, N.J.

Burton, R.R. (1982). Diagnosing bugs in a simple procedural skill. *Intelligent Tutoring Systems* (eds. D. Sleeman and J.S. Brown). Academic Press, London.

Burton, R.R. (1988). The Environment Module of Intelligent Tutoring Systems. *Foundations of Intelligent Tutoring Systems* (eds. M.C. Polson and J.J. Richardson). Lawrence Erlbaum Associates, Hillsdale, N.J.

Clancey, W.J. (1982). Tutoring rules for guiding a case method dialogue. *Intelligent Tutoring Systems* (eds. D. Sleeman and J.S. Brown). Academic Press, London.

Frasson, C., and Gauthier, G. (1990). Introduction. *Intelligent Tutoring Systems (At the Cross-roads of Artificial Intelligence and Education)* (eds. C. Frasson and G. Gauthier). Ablex Publishing Corporation, Norwood, New Jersey.

Lawler, R.W., and Yazdani, M. (1987). Introduction. *Artificial Intelligence and Education. Volume One* (eds. R.W. Lawler and M. Yazdani). Ablex Publishing, Norwood, New Jersey.

McKendree, J. (1990). Effective Feedback Content for Tutoring Complex Skills. *Human-Computer Interaction*, Vol. 5, pp. 381-413. Lawrence Erlbaum Associates.

Routen, T. (1991). Complex Input: A Practical Way of Increasing the Bandwidth for Feedback and Student Modelling in a Statute-Based Tutoring System. *Proceeding of the Third International Conference on Artificial Intelligence and Law.* Oxford, England.

Routen, T. (1992). Reusing formalisation of legislation in a tutoring system. *Artificial Intelligence Review* 6, pp. 145-159. Kluwer Academic Publishers.

Sleeman, D. (1987). Pixie: a shell for developing intelligent tutoring systems. *Artificial Intelligence and Education. Volume One* (eds. R.W. Lawler and M. Yazdani). Ablex Publishing, Norwood, New Jersey.