A PROBLEM SOLVING CONSULTANT IN FINANCIAL ANALYSIS

Markus Lusti

1987

Markus Lusti, Institut für Infcrmatik, Universität Basel, Switzerland

Abstract

Intelligent Tutoring Systems try to cover aspects of domain and pedagogic competence of an ideal human tutor. A complete intelligent tutoring system can be described by the following components: expert model, student model, tutorial model and communication model.

The paper gives an account of FA-TUTOR, an intelligent tutoring system assisting students in the solution to problems in financial accounting (FA). FA-TUTOR consists of an authoring and a tutoring component. The authoring system helps the teacher to prepare problems. The tutoring component allows two styles of interaction: the first giving the student hints about the solution to a problem, the second displaying the right answer and allowing the student to traverse the corresponding solution tree.

The emphasis of FA-TUTOR is on the expert model. The implementation of the problem solving and explanation components is influenced by the methods of programming expert systems: problems are solved by reducing goals of Prolog clauses. A trace collects information relevant to the hints and explanations of the two interaction styles.

Keywords

Computer Assisted Learning, Intelligent Tutoring Systems, Financial Accounting, Knowledge Based Systems, Expert Systems, Prolog

* This paper is a summary of a research report about a system teaching financial accounting to Swiss students (Lusti 1987). Computer outputs have been translated from german.

1 Introduction

Intelligent Tutoring Systems try to cover aspects of domain and pedagogic competence of an ideal human tutor. The design and implementation of such programs is influenced by the methods and tools for developing knowledged based systems. An ideal intelligent tutoring system can be described by the following model (Figure 1).

Component	Tasks
expert model	presents information and problems solves problems explains solutions
student model	identifies bugs analyses bugs diagnoses abilities and motivation diagnoses interaction and learning styles
tutoring model	selects information and problems selects interaction and learning style
communication model	determines appropriate communication style

Figure 1: Functionality of an ideal intelligent tutoring system

Most intelligent tutoring systems implement only a subset of the tasks described in figure 1. The following sections describe FA-TUTOR, an intelligent tutoring system for financial analysis (FA concentrating on the expert and communication model (Figure 2).

Ideal tutoring system	Functionality of FA-TUTOR
expert model	elaborate
student model	rudimentary
tutorial model	rudimentary
communication model	elaborate

Figure 2: FA-TUTOR and the ideal intelligent tutoring system

2 Teaching objectives

2.1 Teaching Domain

FA-TUTOR's expert model covers some central aspects of financial accounting. In essence, financial accounting has three distinct areas of study:

2 the preparation of final accounts, e.g. of balance sheet, profit and loss statement and funds statement

3 the interpretation of final accounts.

2.2 Objectives

The main objective of FA-TUTOR is not to teach factual knowledge but to help students to apply previously learned material to problems presented by the program. The student is therefore supposed to have a basic knowledge of the following concepts:

1 doubly-entry bookkeeping

2 final accounts

3 cash flow

4 selected financial ratios.

By solving the generated problems the student learns

1 to record business transactions in a double-entry bookkeeping system

2 to prepare financial statements, namely

- balance sheet
- profit and loss account

3 to calculate selected financial ratios, for example

- liquidity ratios
- gearing ratios
- profitablity ratios
- credit control ratios
- stock control ratios
- cash flow statistics

3 User interface

FA-TUTOR consists of a tutoring and an authoring component (Figure 3). The authoring component helps the teacher to generate problems which are presented to the student in written form. If the student cannot solve the problem in a straightforward way she may consult the tutoring component for solution hints. If, on the other hand, she is proficient enough to tackle the problem without help, she can compare her final answer with FA-TUTOR's solution. If the solutions differ the user can ask for an explanation of the right solution. The depth of the explanation can be controlled by the student (see further on).

By enabling novel problems to be generated easily and by subsequently monitoring students as they solve these problems, FA-TUTOR assists the teacher in a very time consuming task. In particular, the manual generation of accounting problems can be laborious because the underlying business transactions must meet a variety of constraints. Such constraints may prevent, for example, non-negative assets in the final balance sheet or implausible financial ratios.

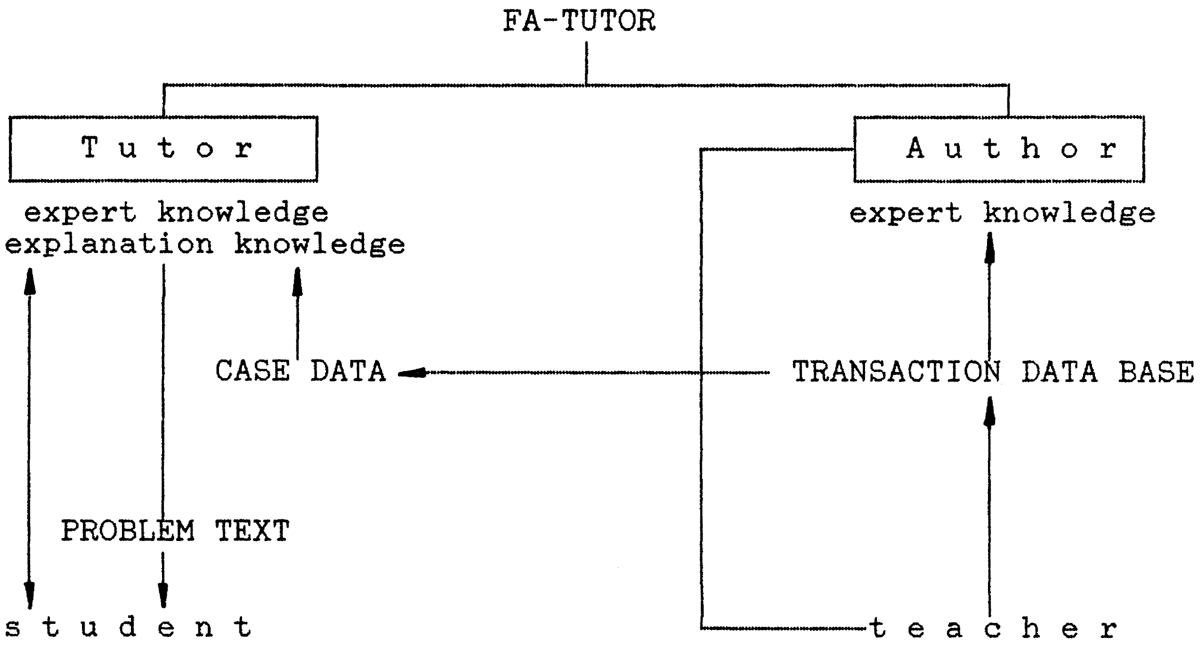


Figure 3: Architecture of FA-TUTOR

The authoring component (figure 4) accesses a data base of business transactions, which can be edited (i.e. the text and the amounts of transactions can be modified, added, or deleted by the teacher). A what-if facility allows the author to inspect the results of choosing different sets of business transactions and other problem data. Like in a spreadsheet, the effect of such variations on the final accounts and the financial ratios is displayed immediately. In calculating the results of different case data the authoring component uses the same expert knowledge as the tutoring component. After the teacher has composed a problem the case knowledge is asserted as a file of Prolog clauses, and a problem text for the student is printed.

The authoring component allows the teacher

1 to generate <u>problems</u>

- to select case data (especially business transactions)
- to modify the task (e.g. the kind and number of ratios)
- to experiment with different case/task combinations and view their effects (what-if facility)
- to format the problem text

2 to edit business <u>transactions</u>

- to modify texts and amounts for transactions
- to add transactions
- to delete transactions

3 to customize the default <u>help</u> information pertaining to

- the chart of accounts and the glossary
- the use of the program

Figure 4: The functionality of the authoring component

Some intelligent tutoring systems generate problems at run time, which allows the adaptation of problems to the state of a student's ability and motivation. But this is only feasable if the task can be parameterized. Task generating systems are possible in some domains of mathematics (see e.g. Brown/Burton 1978, Burton 1982, Sleeman 1985). More complex mathematical tasks, for example problems in SPIRIT (Barzilay 1984), or case texts, for example in Guidon (Clancey 1979), use a problem data base. FA-TUTOR's case data and tasks are too complex to be generated at run time. The ease of specifying tasks and choosing case data, however, should compensate for the lack of fully automated problem generation.

A typical <u>problem text</u> fills three A4 pages and is subdivided into

- a case part containing the description of all transactions and other data and
- a task part specifying what the student is supposed to do.

1 the opening balances of the balance sheet in alphabetic order
2 the names of the profit and loss accounts in alphabetic order
3 natural language descriptions of business transactions and other data.

Since one of the learning objectives is to teach the ordering of financial statements the names of accounts are given in alphabetic order. Figure 5 shows some examples of business transactions appearing in case sections.

- We purchase goods on credit for $3000.
- 30 September 1987: We pay a $2000 cheque for the rent. The ren' is paid half-yearly in arrears on 31 March and 30 September.
- At the end of the accounting period the stock is evaluated at $62 000.
- The proportion of investements replacing depreciated assets is 60%.

Figure 5: Some examples of business transactions and other case data

The task section asks the student

1 to prepare financial statements for the current accounting period
2 to compute selected financial ratios.

Figure 6 summarizes the functionality of the tutoring component. The following sections illustrate the first aspect of the tutoring subsystem, its two interaction styles.

The tutoring component offers

1 two interaction styles, allowing the student
   - to ask for hints during problem solving
   - to request an explanation of the final solution after he has completed a task
2 context sensitive help answering the following questions:
   - Where am I?
   - How shall I use the program?
   - How are the main domain concepts defined?
3 user-friendly interaction characterised by
   - little typing
   - short response times
   - windowing, colour, semigraphics and selective printing

Figure 6: Functionality of the tutoring component

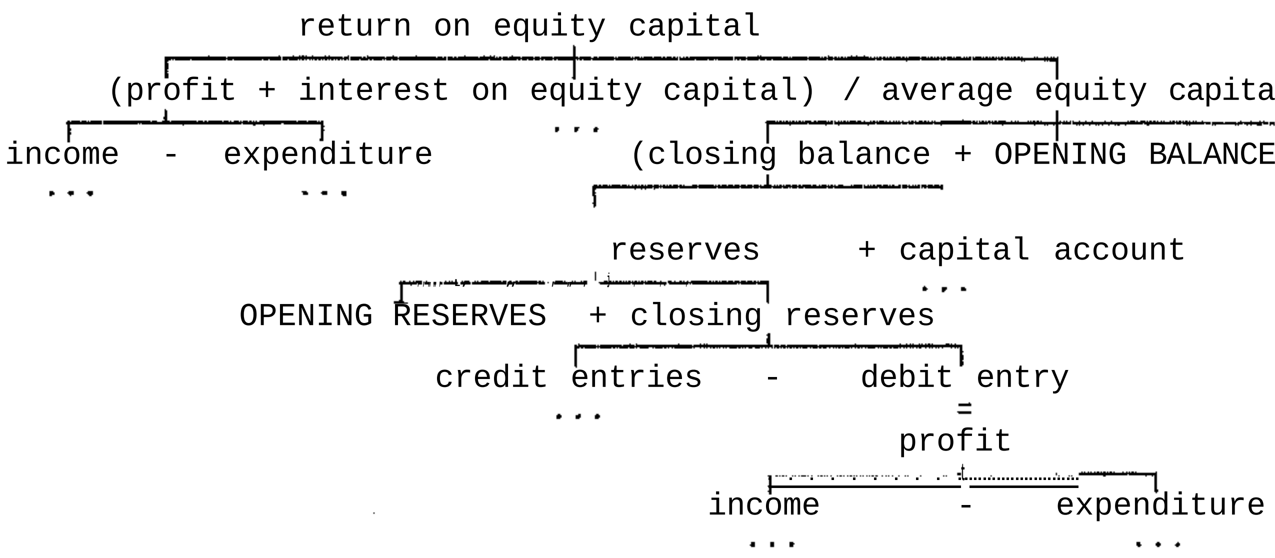
As an example task, we show the calculation of a selected financial ratio. The ratio "return on equity capital" is the quotient of (profit + interest) and the average equity capital (Figure 7). Both the numerator and the denominator can be reduced to other intermediate terms. The stepwise reduction of a term to more basic terms is typical for many problems in financial accounting. Similar problems occur in mathematics teaching; well known examples are symbolic differentiation and integretation (see e.g. Kimball 1973 and Genesereth 1979).

Figure 7 shows a partial solution tree for the calculation of the ratio "return on equity capital[11]. A solution tree contains the following node types:

1 root node (goal) =
  problem answer, e.g. the value of a finacial ratio

2 intermediate nodes =
  terms defining the goal, in particular financial ratios and results from final accounts needed to answer the problem

3 leaves (terminal nodes) =
  business transactions required to determine the intermediate nodes.


return on equity capital
(profit + interest on equity capital) / average equity capita
income - expenditure
...
(closing balance + OPENING BALANCE
reserves + capital account
OPENING RESERVES + closing reserves
credit entries - debit entry
=
profit
income - expenditure

Legend

uppercase letters: leaves
lowercase letters: nodes (except leaves)
... : subtree (nodes not shown)

Figure 7: Partial solution tree for a selected financial ratio

When the student asks for a hint or for the solution to a problem, the system solves the problem, protocols the solution path and offers the student a user-friendly way of traversing the protocol. The student can traverse it breadth-first (left to right and right to left) or depth-first (downwards and upwards).

The kind of node information depends on the selected interaction style. Under the <u>interaction style HINT</u> the student gets hints on how to solve a subtask. Imagine a student who learned the following definition of "quick-assets ratio":

quick-assets ratio =
(cash + debtors) / current liablities due within three months.

Although she knows the top formula of the solution tree, she is unable to remember the subtree for "cash". She therefore expands the corresponding node and gets the following advice:

"cash = sum of

(1) cash in hand
(2) cash at bank
(3) cash at post
(4) realisable investments"

By typing a number from 1 to 4 she can request further information about the elements of "cash".

Working under the <u>interaction style SOLUTION</u> the system presents the full answer to a subproblem and explains it. Suppose a student has found an answer of her own for the "quick-assets ratio". He can then compare his solution to the teacher's solution by asking the system about its solution path. The display for the top level of the tree might be:

"The quick-assets ratio is 1.84, because

(1) 1.84 = (21 900+ 20 800) / 2 320
(2) 21 900 = cash
(3) 20 800 = debtors
(4) 2 320 = current liabilities due within three months"

Realizing that her answer is not equal to the displayed answer because she did not get 20 800, the student follows the solution path for "debtors". At a later node, she might learn that the ultimate reason for the wrong result was the incorrect recording of the second transaction in the problem text:

"The sum of all credit entries in "debtors" is 10 500 and is the result of the following transactions:

transaction	**debit account**	**credit account**	**amount**
2	bank	debtors	7 500
5	post	debtors	3 000"

By typing the number of a transaction the student gets additional information about how to debit and credit the accounts concerned.

1 Expand any node on the current tree level. In the above "cash" node the student can for instance request hints for all four child nodes of "cash".

2 Go back to the predecessor node.

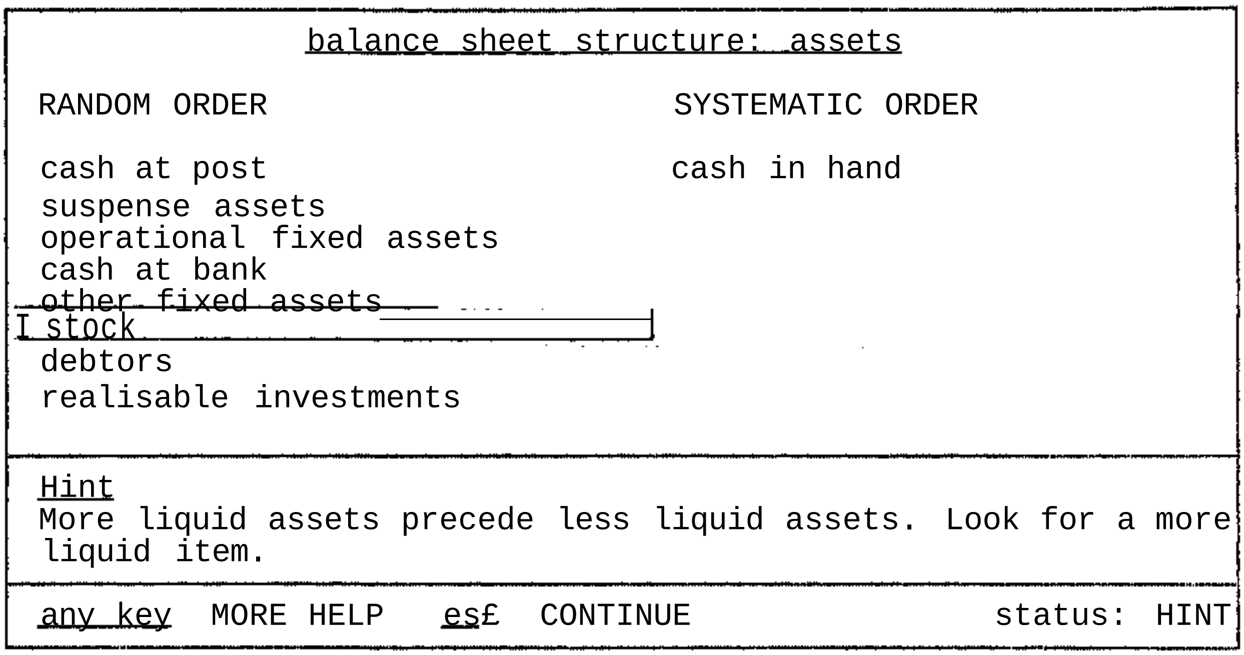
3 Go back to the root node (goal).

4 Change the intQrac1;4,pn style.

For most learning objectives, the traversal of solution trees underlies the interaction between system and student. There are, however, some domains which are less amenable to deeply structured deductive explanations. Two important examples are the classification of accounts and the structuring of financial statements. Figure 8 shows an example interaction from the latter domain.

The items of the balance sheet the student is supposed to prepare are given in random order. Her task is to order the items according to accounting standards. She points at the item she thinks should come next. The system then applies its ordering rules to the answer, and, if the answer is right, the item is moved to the column titled "systematic order". Contrary to most other FA-TUTOR subjects, problems in this domain may have more than one solution.

In figure 8, the student's first answer ("cash in hand") was correct. Her second choice ("stocks"), however, contradicts the following rule: "The assets of a balance sheet must be ordered by liquidity". The system therefore opens a help window and displays advice on how to correct the mistake. If the student needs more help she can press any key to get information about the accounts concerned.



balance sheet structure: assets

RANDOM ORDER

cash at post
suspense assets
operational fixed assets
cash at bank
other fixed assets
I stock
debtors
realisable investments

SYSTEMATIC ORDER

cash in hand

Hint
More liquid assets precede less liquid assets. Look for a more liquid item.

any key MORE HELP es£ CONTINUE status: HINT

Figure 8: Example from the learning domain "Ordering of financial statements

3 Architecture

Section 2 looked at FA-TUTOR from a user's perspective. The following sections examine the system's architecture from the implementor's point of view.

3.1 Expert Model

The expert component of an intelligent tutoring system tries to model the domain competence of a human teacher. Domain knowledge enables an ideal intelligent tutoring system to solve and to generate problems. An "articulate expert" is also able to explain solutions. The program not only knows one or more solution procedures but also explains its solution in a user-friendly way.

The difficulty of designing an expert model depends on the complexity of the subject. Algorithmic domains are easier to model than domains with more or less vague heuristics. For the first case, numerous examples exist in mathematics and related domains (for reviews see e.g. Park 1987 and Lusti 1986, 1987). An example using heuristic knowledge is GUIDON (Clancey 1979, Richer/Clancey 1987). It extends the heuristic knowledge of MYCIN by tutoring rules and additional facts and rules about how to explain MYCIN's results to students.

expert model of FA-TUTOR solve problems by processing Prolog clauses organised in an easily explainable way. Running through the clauses, the program collects all the information the explanation component needs under both interaction styles. Such explanations may be classified by the following criteria:

1 Comprehensiveness: The explanation component may output all traced clauses including all instantiated variables, or it only displays a subset of the clauses and variables. FA-TUTOR uses both kinds of explanations.

2 Depth: Explanations may rely entirely on the protocol of the traced clauses, or may also explain implicitly-used knowledge and solution strategies (like the implicit depth-first strategy of a Prolog interpreter). FA-TUTOR's explanations essentially use protocols. Some of them, however, display additional information not used by the problem solving component (cf. figure 8).

3 Extent of user control: Most protocols contain a wealth of information. In order not to overload the user's receptivity an explanation component may offer powerful traversal functions.

4 Language generation: Algorithms or templates translate the internal representation of the clauses into a user-friendly external representation. FA-TUTOR uses templates to translate protocols into natural language text.

5 Implementation: Two main factors determine the implementation of an explanation component in Prolog:

   (a) **protocol data structure**: The solution protocol may be built by a meta-interpreter, by an additional argument of the clauses concerned, or can be asserted as a set of clauses. In the first two cases, the trace develops locally (without side effects), in the last case, it is generated globally.

   (b) **Interdependence of problem solving and explanation**: Is it possible to solve a problem without generating the overhead of a protocol?

The following example illustrates how those explanations which are based on a trace of FA-TUTOR's solution path are implemented. Figure 9 shows a Prolog rule defining the top level of the solution tree of figure 7.

```
ratio("return on equity capital",
       Result,
       trace(ratio("return on equity capital", Result),
             [trace(plusdiv(Result, Profit, Interest, Capital),
                    []
                   ),
              Trace1,
              Trace2,
              Trace3,
             ]
            )
     )
IF
  profit_loss_account("profit", Profit, Trace1) AND
  interest_on_equity_capital(Interest, Trace2) AND
  average("equity capital", Capital, Trace3) AND
  Result = (Profit + Interest) / Capital.
```

Figure 9: Prolog rule illustrating the generation of a solution protocol in FA-TUTOR

Ratio rules are structured as follows: The first argument of the rule head contains the name of the financial ratio whereas the second argument is the result of the ratio. The third argument is a data structure containing the explanation tree. Its first argument holds the name and value of the ratio whose calculation is traced. While the first argument records rule head information the second lists rule body information being of interest for the explanation. The variables Trace1, Trace2 and Trace3 in figure 9 contain the protocols of the first three antecedents. plusdiv/4 contributes to the explanation of the fourth antecedent. The elements of the protocol list have the same structure as trace/2. Figure 10 shows the recursive nature of trace/2. The explanation tree can reach a considerable size. Great care has therefore been taken to ensure a user-friendly traversal of the protocol.

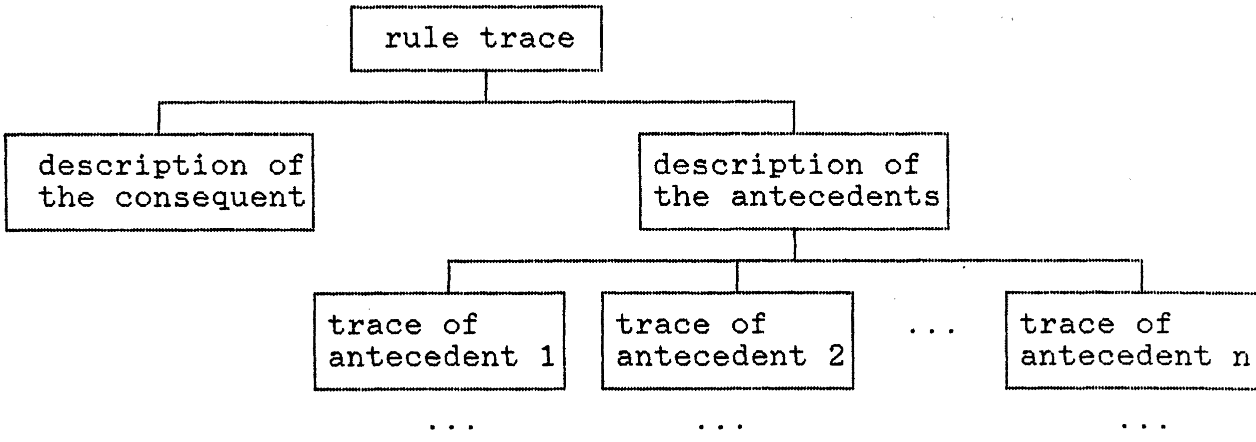


Figure 10: The recursivity of the protocol argument

suggested by Clark/McCabe (1982). Figures 9 and 10 present a modified variant, which has the following advantages and disadvantages:

Advantages

- Explanations are handled locally (i.e. by parameters). This avoids the notorious drawbacks of manipulating global data by assert and retract predicates.
- Tree manipulation (building, traversing, deleting) can be easily described by recursive data structures and algorithms.

Disadvantages

Problem solving and explanation components are not modular enough. It is not possible, for example, to evaluate a goal without accepting the ensuing protocol overhead. And, what is worse, it is difficult to develop a shell. A shell enables the teacher to add learning material (problem solving and explanation knowledge) without having to know about the internals of the program.

Meta-interpreters allow a better separation of problem solving and explanation (Sterling 1986). By using, for example, the built-in predicate "clause(RuleHead, RuleBody)", a meta-interpreter "solve(Goal, Protocol)" collects all explanation-relevant rules in a protocol tree. The protocol is handed over to a predicate "traverse(Protocol)" which offers user-friendly ways of looking at the protocol. Contrary to the method used by FA-TUTOR, the information is not contained in additional rule arguments but collected by using the metaprogramming predicate "clause/2". If explanations are not needed, a goal can be evaluated without meta-interpreter, i.e. without incurring any protocol overhead. Moreover, rules can be added without further protocol arguments.

For efficiency reasons, the explanation component of FA-TUTOR does not use a meta-interpreter. First, because it is implemented in a very restricted (but fast) quasi-Prolog (Borland 1986) which has only few metaprogramming facilities. Second, because meta-interpreters put a heavy load on system resources. Performance considerations are particularily important for an intelligent tutoring system supposed to run on widely available and inexpensive personal computers. The disadvantage of incurring protocol overhead has been alleviated by lemma generation. The authoring component saves some computations as lemmas. These are used when the recording of protocols is not necessary.

With the advent of commercial expert system building tools, it has been suggested they might be used for the implementation of teaching aids (Harmon 1987). Figure 11 shows a commonly accepted but crude model of an expert system. A comparison with figure 3 reveals the similarity between the architecture of FA-TUTOR and a simple expert system model. To conclude that an intelligent tutoring system can easily be implemented by expert system building tools available on PC's is, however, hasty. Many tools are not flexible enough to implement all components of an intelligent tutoring system. This is particularily true for the student model, the communication model and the explanation component. It may even be difficult or impossible to implement a knowledge acquisition component if it deviates so much from a standard knowledge-base interface as FA-TUTOR does. There are several other reasons why expert system generators may not be suitable for the development of an intelligent tutoring system (Wallach 1987):

- The performance of applications created by generators tends to be slow. This applies particularly if many external routines are accessed.
- Porting applications to different computers may be more difficult.
- Language and licensing costs are high.

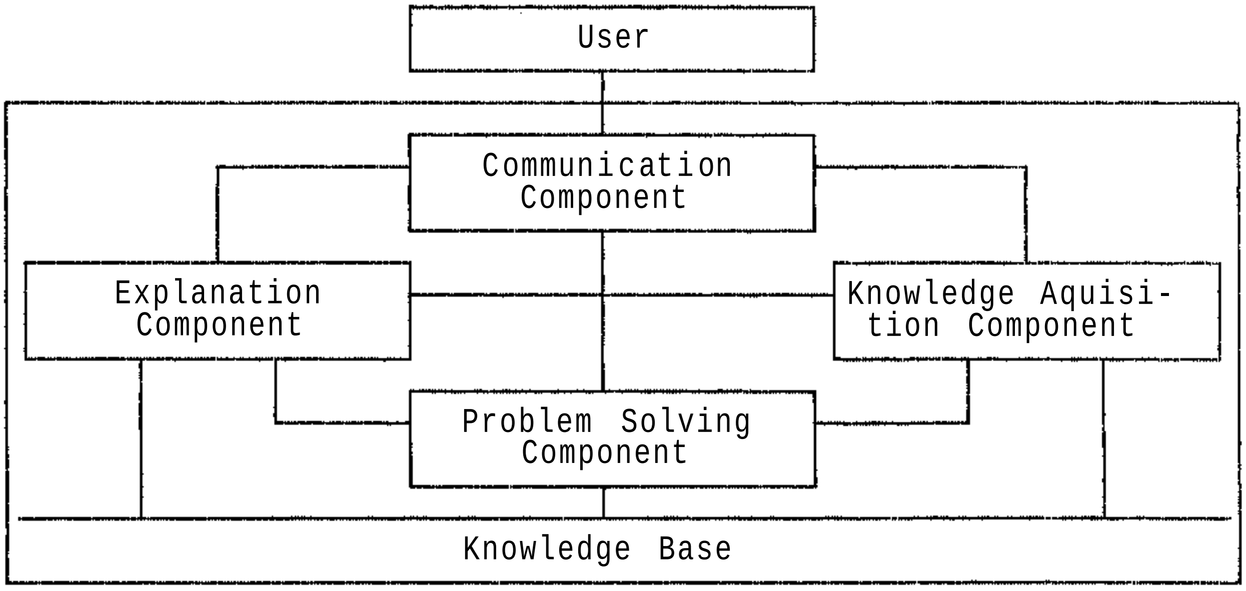


Figure 11: FA-TUTOR and Expert Systems

3.2 Tutoring Model and Student Model

A student model monitors the user's problem solving and diagnoses his ability, motivation and interaction style. The tutorial model starts from this diagnosis and devises a therapy for improving the student's problem solving behaviour.

FA-TUTOR puts the emphasis on the expert and communication model. It assists with problems which most students with previous (conventionally acquired) knowledge can answer without close guidanthe problem and the learning objectives to the user's ability and motivation:

1 The teacher may tailor the problem text to the ability of a group of students by choosing suitable task data (e.g. ratios) and case data (e.g. transactions).

2 The authoring system allows the teacher to readily adapt the main help texts to the students needs.

3 The student can choose freely the learning domain she wants to cover. In this respect, FA-TUTOR resembles much more an unobtrusive consultancy system than a teaching system actively guiding the student.

4 The student can at any time choose between two interaction styles.

5 The student can easily reach every node of the solution trees.

3.3 Communication Model

FA-TUTOR is essentially driven by pop-up menus. Free input, for example the text of business transactions, can be typed in by a full-fledged editor. The explanation component generates text by substituting instantiated Prolog variables to template placeholders.

A natural-language user interface is an important part of some intelligent tutoring systems. As most answers FA-TUTOR expects are out of a small number of alternatives, a natural language input format would not improve user-friendliness.

The communication interface of FA-TUTOR can be characterised by the following features:

1 extensive help facilities

At any moment, the student can ask for context sensitive information about her position in the menu hierarchy and the use of FA-TUTOR. In most contexts, she can also request information about the structure and contents of the accounting chart and about the definition of concepts used in financial accounting.

2 user-friendly input/output

Pop-up menus, prompts, defaults, editing facilities, plausibility tests, short response times, visual and acoustic clues and selective printing contribute to the ease of learning and use.

4 Implementation Issues

Only a few intelligent tutoring systems have been implemented in Prolog (see e.g. Davies 1985, Begg 1987). FA-TUTOR uses Prolog as the implementation language for the following reasons:

development and modification of the expert model

2 In FA-TUTOR's domain, Prolog's control strategy simulates effectively the top-down strategy of human problem solvers

3 The built-in pattern matching facility greatly helps with the user-friendly output of solution protocols by templates.

The size of FA-TUTOR's source code is about 200K (1000 clauses, object code: 390K). Most of the code concerns the user interface. In this domain, the advantages of logic programming hardly show. Instead, Prolog's shortcomings for procedural tasks become noticeable. Non-backtrackable input/output predicates with side effects force the programmer to deviate from good programming style. This is particularly true for windowing and cursor manipulation.

As an implementation dialect, Borland's quasi-Prolog was chosen. The main reasons are good run-time efficiency, easy access to IBM PC hardware and system software and a comfortable development environment. The choice of widely used hardware (IBM PC compatibles) and a compiler generating fast code is motivated by the desire to use the program in schools. Turbo Prolog, however, is weak in some areas which are commonly regarded as essential for Prolog. Typed variables and rudimentary meta programming facilities can make programming very awkward.

5 Possible Extensions

The following extensions would be possible without altering the basic implementation of FA-TUTOR:

1 A teacher-definable accounting chart and additonal rules for preparing final accounts would allow for more complex transactions. Examples are limited company accounts, taxation, business expansion and the inclusion of legal procedures for the handling of certain accounting procedures.

2 FA-TUTOR's analysis of final accounts is based on internal figures. The form and minimum contents of external (published) accounts are determined by legislation. Since these accounts are open to inspection by the general public, who will include competitors, only the legal minimum is disclosed. Analyzing published accounts, therefore, requires an allowance for window-dressing. Such adjustments may be interpreted as business transactions, which means that the basic logic of FA-TUTOR wouldn't change.

3 A funds statement identifies movements in assets, liabilities and capital during a period, and the net effect on net liquid assets (the funds). Moreover, it explains how operations of a business have been financed, and how its financial resources have been used. A funds statement could be prepared by relying on the same case data FA-TUTOR uses to prepare its final accounts.

4 To include teacher-definable ratios, the knowledge-acquisition component would convert ratio definitions into clause form and save them in a external database. Such a modification would, however, be easier using full Prolog. External databases in Turbo Prolog can only contain facts.

Summary

There are other subjects than financial accounting which are suitable for such tutoring systems (see Lusti (1987) for an examination of some other domains, e.g. law and operations research). The following paragraph summarizes the main domain-independent characteristics of FA-TUTOR:

Learning Objectives

1 FA-TUTOR's main objective is not to teach new factual knowledge but to make students apply previously learned problem-solving knowledge to new problems.

2 The problems have decidable (unique or multiple) solutions.

3 The problem-solving knowledge is vertically structured, i.e. it can be represented by (preferably) deep solution trees. Laterally structured knowledge is less suitable, because the explanation component essentially protocols solution paths.

Expert Model

4 The main goal of the authoring component is the semi-automatic generation of an indefinite number of case problems.

5 The expert model solves most problems on its own ("understands what it teaches").

6 The symmetry of problem solving and explanation facilitates the generation of explanations from the expert knowledge.

Tutoring and Student Model

7 FA-TUTOR is a consultancy system leaving much initiative to the student.

8 The tutoring component offers two interaction styles, one allowing the student to ask for hints during problem solving, the other allowing him to ask for an explanation of the results in case his answers are wrong.

Communication Model

9 A user-friendly interface minimizes the effort of learning and using both the tutoring and the authoring component. The following features contribute to the user-friendlyness of the system: pop-up menus, windowing, colours, semigraphics, immediate context-sensitive help.

Production Environment

10 FA-TUTOR's ideas can be realized on cheap personal computers widely used in education.

Acknowledgements

I am grateful to Ben du Boulay for encouraging me to write this paper and to Chris Sothcott for reading the manuscript.

References

Barzilay, A. (1984) An expert system for tutoring probability theory, PH.D. Dissertation, University of Pittsburgh

Begg, I. **M.,** Hogg, I. (1987) Authoring systems for ICAI, in Kearsley, G. (ed.), Artificial intelligence and instruction. Applications and methods, 323-346, Reading, Massachussets, Addison-Wesley

Borland International, Inc. (1986) Turbo Prolog. Owner's Handbook, Scotts Valley

Brown, J. S., Burton, R. R. (1978) Diagnostic models for procedural bugs in basic mathematical skills, Cognitive Science, 2, 155-192

Burton, R. R. (1982) Diagnosing bugs in a simple procedural skill, in Sleeman, D., Brown, J. S. (eds.) Intelligent tutoring systems, 1982, London, Academic Press, 157-183

Clancey, W. J. (1979) Transfer of rule-based expertise through a tutorial dialogue, PH.D. dissertation, Computer Science Department, Stanford University

Davies, N. G., Dickens, S. L., Ford, L. (1985) TUTOR - A prototype ICAI system, in Bramer, M. (ed.) Research and development in expert systems, Cambridge, Cambridge University Press

Genesereth, M. R. (1979) The role of plans in intelligent teaching systems, International Journal of Man-Machine Studies, January 1979.

Harmon, P. (1987) Intelligent job aids: How AI will change training in the next five years, in Kearsley, G. (ed.), Artificial intelligence and instruction. Applications and methods, 165-190, Reading, Massachussets, Addison-Wesley

Kimball, R. B. (1973) Self-optimizing computer-assisted tutoring: theory and practice, Technical Report No. 206 (Psychology and Education Series), Institute for Mathematical Studies in the Social Sciences, Stanford University

Lusti, M. (1986) Computergestiitztes Lernen und Künstliche Intelligenz, in Schulz, A. (ed.), Die Zukunft der Informationssysteme. Lehren der 80er Jahre. Reihe Betriebs- und Wirtschaftsinformatik, Berlin 1986, 493-503, Springer

Lusti, M. (1987) Methoden wissensbasierter Systeme bei der Entwicklung von Lernprogrammen. Ein Beispiel aus dem betrieblichen Rechnungswesen, Habilitationsschrift Hochschule St. Gallen

Park, 0., Ray, S. P., Seidel, R. J. (1987) Intelligent CAI: Old wine in new bottles, or a new vintage?, in Kearsley, G. (ed.), Artificial intelligence and instruction. Applications and methods, 11-45, Reading, Massachussets, Addison-Wesley

Richer, M. H., Clancey, W. J. (1987) GUIDON-WATCH: A graphic interface for viewing a knowledge-based system, in Lawler, R. W., Yasdani, M. (eds.)> Artificial Intelligence and Education. Learning Environments and Tutoring Systems, Vol. 1, 373-412, Norwood, Ablex Publishing

Sleeman, D. (1985) Inferring (mal)rules from pupil's protocols, in Steels, L., Campbell, J. A. Progress in artificial intelligence, Chichester, Ellis Horwood

Sterling, L., Shapiro, E. (1986) The art of Prolog. Advanced programming techniques, Cambridge, Massachussets, MIT Press

Sterling, L., Lalee, M. (1986) An explanation shell for expert systems, Comput. Intell. 2, 136-141

Wallach, B. (1987) Development strategies for ICAI on small computers, in Kearsley, G. (ed.), Artificial intelligence and instruction. Applications and methods, 305-322, Reading, Massachussets, Addison-Wesley