Serdica

Bulgariacae mathematicae publicationes

Сердика

Българско математическо списание

SERDICA — Bulgaricae
mathematicae publicationes
20 (1994) 120-132

FLEXIBLE PROBLEM SOLVING FOR INTELLIGENT TUTORING SYSTEMS WITH TRIPS

HRISTO PETKOV, DANAIL DOCHEV

ABSTRACT. This paper describes the aspects of flexibility and explicitness of a system called TRIPS (TRIgonometry Problem Solver) intended to be used for the purposes of Intelligent Tutoring Systems (ITS).

Explicitness demands presenting the knowledge from the problem domain in details, clearly and fully expressed in an obvious form. Explicitness includes also the requirement for the solution to be apparent on the basis of explanations, comprehensible for the user. The flexibility of the Domain Expert implies providing multiple reasonable solutions as well as the ability to implement different teaching strategies with the Domain Expert.

The main objectives of the procedural knowledge representation in TRIPS is to assure opportunities of making the solutions provided by TRIPS explicit and suitable for educational purposes.

The approach has been used to construct performance models of how the Teaching Expert of an overall ITS with conventional architecture actually executes the skills that are to be tutored, implementing different teaching methods.

1. Basic Architectural Issues. This paper presents a new view of problem solving, motivated by flexibility and explicitness for the purposes of education. The problem solving aspects are discussed in the context of an experimental program called TRIPS (TRIgonometry Problem Solver).

TRIPS is a new improved version of the T-TUTOR (Trigonometry Tutor), [1], [2], (1987), experimented in the domain of high school level trigonometry and it is intended for implementation in an Intelligent Tutoring System (ITS).

TRIPS has been developed in a logic programming environment of ZPROLOG v. 5.0, a product of the Institute of Informatics, [3], [4], (1989). A detailed description of the domain knowledge conceptualization and its realization in PROLOG is provided in [5].

Let us consider a conventional ITS, developed on the basic principles as presented in [6] and [7] and later on refined in [8]. The basic architectural presupposition is that the overall ITS consists of five expert modules: Domain Expert, Teaching Expert, Cognitive Expert, Environment Expert and a Psychologist. In the present paper we shall constrain this architecture to the the Domain Expert and shall lay down the foundations of some of the issues, concerning the Teaching Expert. Similar constraints for research purposes can be seen in [11], [12] and [13]. TRIPS is the Domain Expert and draws inferences, while the Teaching Expert coupled to TRIPS (TET) uses these inferences (called solutions) and applies different teching strategies over them.

TRIPS serves three roles in this overall system:

- it solves the problem step by step;
- it records the inference of the solution explicitly;
- it provides explanations on that solution;
- it functions as a cash for all the inferences made over a given domain problem.

The Teaching Expert TET performs the following functions:

- makes decisions for applying a specific teaching strategy;
- uses the data from the solution provided by TRIPS in order to implement the chosen teaching strategy;
- generates appropriate tests for the specific domain problem and for the selected teaching strategy.

2. Knowledge Representation in TRIPS. The general idea used in the knowledge representation in TRIPS is that the process of domain problem solving involves a set of procedures detached as procedural knowledge,[9]. The rewrite rules, the methods and the strategies in TRIPS are such procedures.

A *rewrite rule* is an elementary procedure designating a specific transition in problem solving motivated by a specific domain formula. The rewrite rule makes the transition by substitution of one trigonometric structure in a trigonometric expresion with another, equivalent to it.

Knowledge, specific for the domain under consideration is used to classify the rewrite rules into *transformations* (in TRIPS such transformations are – solving brackets (SB),arithmetic simplification (AS), trigonometric simplification (TS), arithmetic decomposition (AD) etc.). The transformations are transitions motivated by the trigonometric formulae but grouped according to the procedural processing.

Other procedural knowledge concepts are the method and the strategy. The *method* is a set of transformations and a finite set of knowledge based rules for applying these transformations. The method is a specific way of achieving a specific goal or a subgoal. An example of a method for simplifying a trigonometric expression is given below:

Method for Simplification by Solving brackets

- a set of transformations – AS, TS and SB.

- a set of rules:
  - Apply TS always when possible.
  - Apply AS only when TS cannot be implemented.
  - Apply SB only when no other transformation is possible.
  - If no transformation is applicable then go to an end check.

The ***strategy***, on the other side is a domain specific procedural skill for solving a given class of problems on the basis of applying a method or a set of methods.

The formulation of a strategy concept in the procedural knowledge provides the opportunity of using different methods to achieve one and the same goal or subgoal in a given strategy. Thus different reasonable alternatives of a solution are afforded.

Hence for example if M is the set of the different methods for presenting a trigonometric expression in the form of multipliers, and the different methods of this set are indexed $i = 1, 2, \ldots, M$; N-the set of different methods for simplifying a trigometric expression $(k = 1, 2, \ldots N)$; P the set of different methods for decomposition into a product of multipliers (l=1,2,..P) then an exemplary strategy for solving trigonometric equations could be as follows:

- The set of methods – $m_i \in M$, $n_k \in N$, $p_l \in P$.
- The set of procedural rules:

IF $m_i \in M$ & ...THEN $n_k \in N$; IF $n_k \in N$ & ...THEN $p_l \in P$ ...For example the strategy of simplifying domain problems (those problems are solved with applying of only one method) may be as follows:

An Example of a Strategy:

- The set of methods – $m_i \in M$, $n_k \in N$, $p_l \in P$...
- The set of heuristic procedural rules:
  - If there is more than one method left choose and apply method $m_i \in M$ on the basis of an estimate. (In TRIPS for example as such an estimate has been used the logical strength of the metaoperator. One and the same set of preconditions activates more than one metaoperator, and the greater the number of the operators included in a metaoperator, the greater its logical strength is.)
  - The metaoperators with a better estimate in terms of the final goal (greater logical strength) are tried earlier.
  - If a method succeeds, then compare it with the remaining methods by using a predefined bundle of indicators. (A method succeeds when the goal or the subgoal it is applied for is achieved. As a bundle of indicators is used the number of steps in a solution. The better steps (the steps which lead faster to the goal or which resolve a conflict) should be carried out earlier etc.).
  - If only one method can be applied, do it unconditionally.

3. Solution and its Evaluation. TRIPS uses the state space paradigm for defining the domain problems. In the context of TRIPS a solution can be defined as follows:

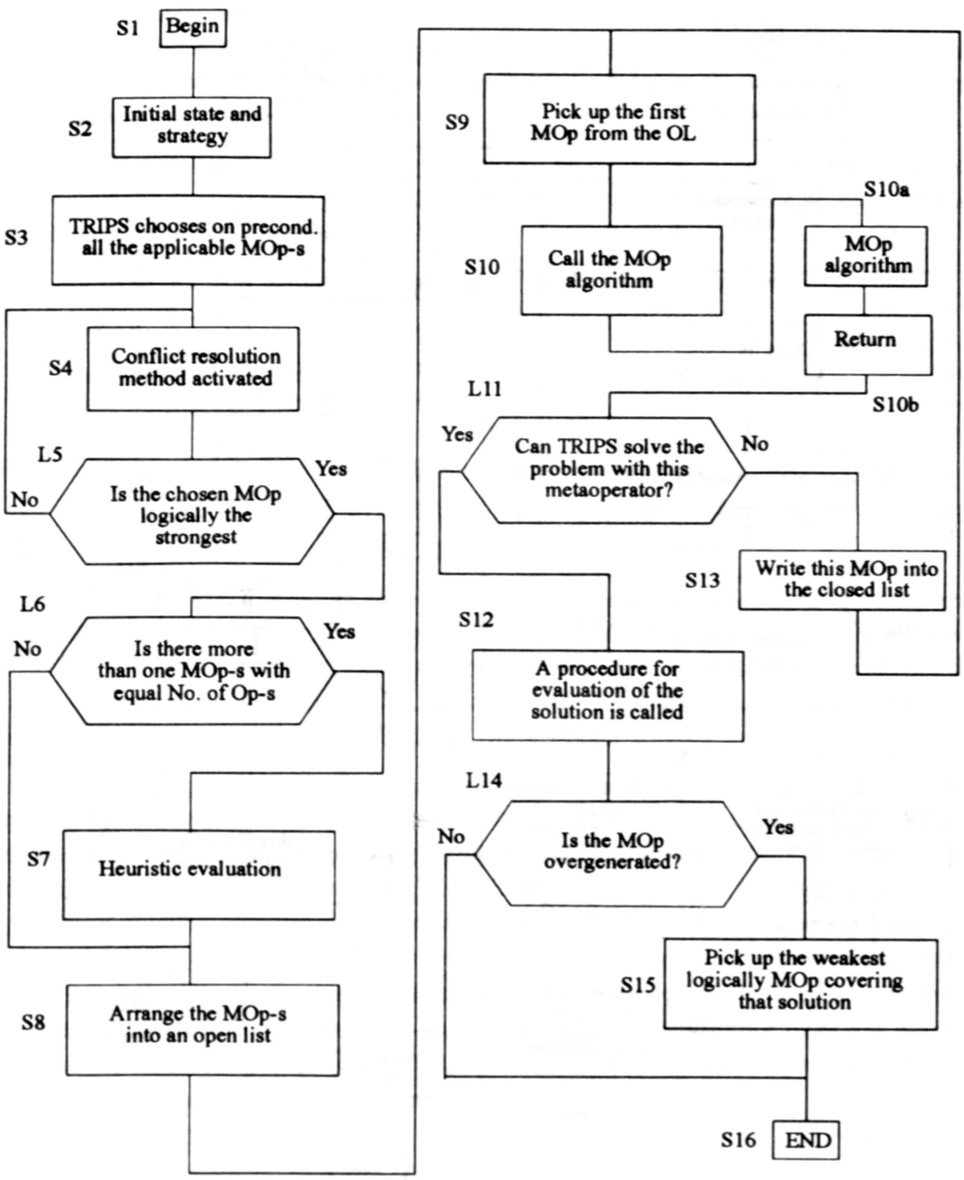


Fig. 1 An Example of a Strategy Algorithgm.

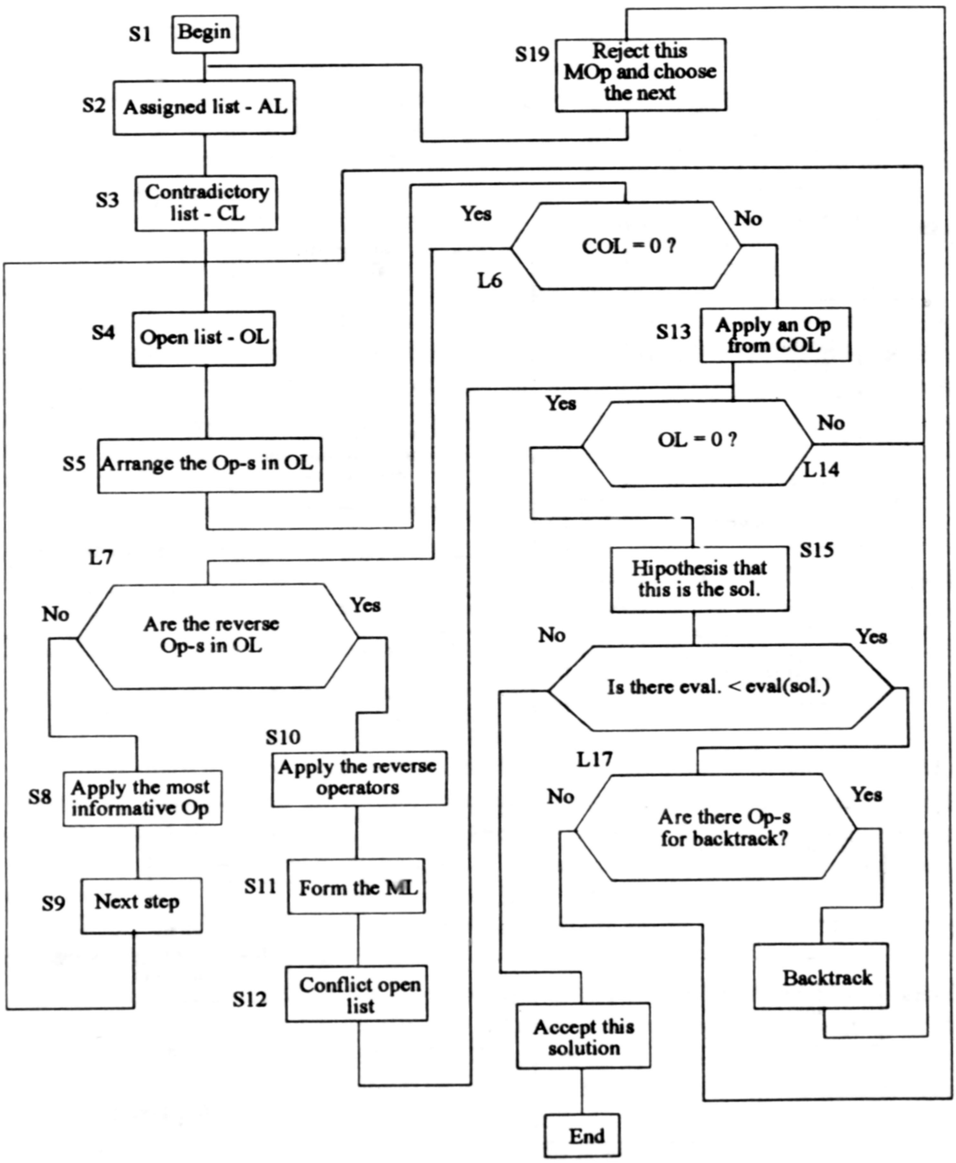


Fig. 2 An Example of a Metaoperator Algorithgm.

• The solution is presented as a set of *states* (lines); $L, l = 1, 2, \ldots, L$; set of operators (transformations), accomplishing the transition from one state to another; $T, t = 1, 2, \ldots, T$; and a set of means-ends rules for applying the transformations in a given sequence.

In this paradigm a problem can be defined by its initial state the control strategy for achieving the goal state and the goal state itself. Depending on the assignment there exist different types of problems.

An example of a problem definition in TRIPS is given below:

Given:

{initial state} $2 - \sin^2\alpha + \cos^2\alpha + \cos\alpha(\operatorname{tg}\alpha + \operatorname{ctg}\alpha)$

{control strategy} → Simplify the expression

Find out: {goal state} → The simplified expression (not known).

To solve this problem TRIPS will use the example strategy algorithm and the metaoperator algorithm, given on Fig. 1 and Fig. 2 respectively.

A general principle when developing a knowledge based system is to separate the knowledge base from the inference engine and that the knowledge should be represented in a uniform manner. This principle is violated in TRIPS by implementing metaknowledge (the strategy and the metaoperator algorithms) over the logical inference which provides a greater flexibility of the problem solving machine.

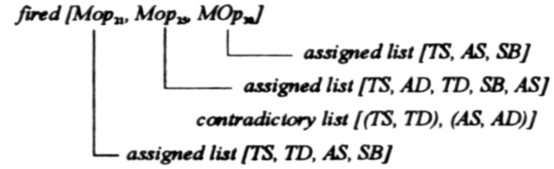
To evaluate a solution, no matter offered by the student or by the system, specific metrics should be implemented so as to estimate the behavior. In TRIPS, when solving a problem for simplifying a trigonometric expression the following metrics, based on three component vector V(A,T,U), has been introduced, where A is *arithmetic simplicity*, T – *trigonometric simplicity* and U – *uniformity*. The arithmetic simplicity is the number of monomials in the expression, taken with some factor.The trigonometric simplicity is calculated by assigning some relative values to the trigonometric terms. The constants, sin and cos are taken as a basis, and the remaining ones are assigned in respect to that basis. The uniformity is the number of the different terms in an expression. An example of a solved problem by TRIPS is given below:

Example 1:

1. $2 - \sin^2(\alpha) + \cos^2(\alpha) + \cos(\alpha) * (\operatorname{tg}(\alpha) + \cos(\alpha))$

The heuristic evaluations of this expression are: $A_0 = 4$; $T_0 = 7$; $U_0 = 5$; $V_0^2 = 90$.

And the metaoperators, fired at that step:



This is step S3 from the fired strategy algorithm on Fig. 1. The metaoperators are chosen by preconditions.

The lists are data structures, defined in the following way:

• assigned list (A) - this list includes all the transformations, currently activated for solving the problem;

• contradictory list (C) - contains pairs of a transformation and its reverse transformation;

• open list (OL) - this list includes all the transformations, applicable to the current line of the solution;

• closed list (CL)- a list of the transformations which have not been used by some reason;

• applied list (AL) - contains the applied transformation;

• milestone list (ML)- a list of transformations which are reverse to the ones applied already and which cannot be applied until the conflict open list is empty;

• conflict open list (COL) - a list of transformations which are the reason of applying a reverse transformation.

The list data structures of each transition is called justification. If J is the set of all the justifications for a given problem (contains justifications of all the steps from all the solutions of that problem), then $J : j_{11}, j_{12}, \dots j_{1n}(1); j_{21}, j_{22}, \dots j_{2m} \dots j_{q1}, j_{q2}, \dots, j_{qp}$ where the first index of the justification corresponds to the transition in a specific solution and the second index indicates the alternative solution called variant.

On the next step of the algorithm the metaoperators are arranged in accordance with their logical strength in the metaoperator open list (steps S4, L5, L6, S7 & S8 in the strategy example algorithm). In our case this is the list [MO_{p25}, MO_{p21}, MO_{p30}]. The corresponding rule from the strategy is to apply the better earlier. The logically strongest MOp25 is the first one of the metaoperator open list, used in order to solve this problem.

The operators, associated with MO_{p25} are:

$MO_{p25} \rightarrow$ [TS,AD,TD,SB,AS]; C $\rightarrow$ [(TS,TD),(AS,AD)]

The possible choices at this step are:

A) apply AD $\rightarrow 1 + 1 - \sin^2(\alpha) + \cos^2(\alpha) + \cos(\alpha) * (\mathrm{tg}(\alpha) + \cos(\alpha))$

B) apply TD $\rightarrow 2 - (1 - \cos^2(\alpha)) + \cos^2(\alpha) + \cos(\alpha) * (\mathrm{tg}(\alpha) + \cos(\alpha))$

C) apply SB $\rightarrow 2 - \sin^2(\alpha) + \cos^2(\alpha) + \cos(\alpha) * \mathrm{tg}(\alpha) + \cos(\alpha)^2$

D) apply TS $\rightarrow 2 - \sin^2(\alpha) + \cos^2(\alpha) + \cos(\alpha) * (\sin(\alpha)/\cos(\alpha) + \cos(\alpha))$

The metaoperator algorithm is activated. The heuristic evaluations are as follows: $A_{AD} = 5$; $T_{AD} = 7.5$; $U_{AD} = 5$; $V^2_{AD} = 106.25$ # $A_{TD} = 4$; $T_{TD} = 7.5$; $U_{TD} = 4$; $V^2_{TD} = 88.25$ # $A_{SB} = 5$; $T_{SB} = 7.5$; $U_{SB} = 5$; $V^2_{SB} = 106.25$; # $A_{TS} = 4$; $T_{TS} = 7.5$; $U_{TS} = 5$; $V^2_{TS} = 97.25$.

When executing the steps S2, S3, S4, S5, L6, L7, S8, S9, S10, S11, S12, S13 and L14 of the metaoperator algorithm the following justifications are generated: A – [TS,AD,TD,SB,AS]; C – [(TS,TD), (AS,AD)]; OL – [TD, TS,SB,AD]; CL – []; AL – [TD] generate [AS, SB] written in the *conflict open list*; ML – [TS]; COL – [AS,SB].

At this step of the algorithm the contradiction should be solved. It can be solved either by applying the TD or by the AD operator, but the TD is much more informative. Note that TD is applied here not because it is the first in the open list, but because it is the first contradictory operator that can be applied. If TD was not at the first place in the open list, the operators preceding it should have been written automatically into the closed list, without being applied. After applying TD, the TS operator, contradictory to it, is written into the milestone list, and the reasons for applying the TD, the operators AS and SB are written into the conflict open list. AS and SB are with the greatest priority and should be applied at the first opportunity.

2. $2 - (1 - \cos^2(\alpha)) + \cos^2(\alpha) + \cos(\alpha) * (\mathrm{tg}(\alpha) + \cos(\alpha))$

With heuristic evaluation $A_0 = 4$, $T_0 = 7.5$, $U_0 = 4$, and $V_0^2 = 88.25$. The operators applicable at this stage are SB and TS.

A) apply SB $\rightarrow 2 - (1 - \cos^2(\alpha)) + \cos^2(\alpha) + \cos(\alpha) * \mathrm{tg}(\alpha) + \cos(\alpha)^2$

B) apply TS $\rightarrow 2 - \sin^2(\alpha) + \cos^2(\alpha) + \cos(\alpha) * (\mathrm{tg}(\alpha) + \cos(\alpha))$

The heuristic evaluations are: $A_{SB} = 6$; $T_{SB} = 8$; $U_{SB} = 4$; $V^2_{SB} = 116$ # $A_{TS} = 4$; $T_{PS} = 7$; $U_{TS} = 5$; $V^2_{TS} = 90$. The open list is: OL – [TS,SB]. Since the operator TS is in the milestone list then its application is forbidden until the conflict resolution list becomes empty, and this will happen after the application of SB and AS. So the TS is written into the closed list, and the SB operator is applied because it is the only applicable operator from the conflict resolution list. If there was any other operator besides AS before it in the open list it would be written into the closed list as well. The generated justifications at this step are: A – [TS,AD,TD,SB,AS]; C – [(TS,TD),(AS,AD)]; OL – [TS,SB]; CL – [TS]; AL – [SB] and SB is removed from the conflict open list; ML – [TS]; COL – [AS].

3. $2 - 1 + \cos^2(\alpha) + \cos^2(\alpha) + \cos(\alpha) * \mathrm{tg}(\alpha) + \cos(\alpha)^2$

$A_0 = 6$, $T_0 = 8$, $U_0 = 4$, $V_0^2 = 116$.

A) apply AS $\rightarrow 1 + 3 * \cos(\alpha)^2 + \cos(\alpha) * \mathrm{tg}(\alpha)$

B) apply TS $\rightarrow 2 - 1 + \cos(\alpha)^2 + \cos(\alpha)^2 + \sin(\alpha) + \cos(\alpha)^2$

$A_{AS} = 3$; $T_{AS} = 5$; $U_{AS} = 4$; $V^2_{AS} = 50$ # $A_{TS} = 6$; $T_{TS} = 6.5$; $U_{TS} = 3$; $V^2_{TS} = 87.25$

The open list is [AS,TS]. The operator from the conflict open list to be applied is AS. Since the TS operator is forbidden by milestone it is not written in the closed list because it does not precede AS. If there exists an operator foregoing AS in the open list then it should be written into the closed list. The following justifications are obtained: A – [TS,AD,TD,SB,AS]; C – [(TS, TD), (AS, AD)]; OL – [AS,TS]; CL – []; AL - [AS] and AS is removed from the conflict open list; ML – [TS]; COL – [].

When the conflict open list is empty, the operator from the milestone can be applied. The next step in the solution is:

4. $1 + 3 * \cos^2(\alpha) + \cos(\alpha) * \text{tg}(\alpha)$; $A_0 = 3$, $T_0 = 5$, $U_0 = 4$, $V_0^2 = 50$.

A) apply TS $\rightarrow 1 + 3 * \cos(\alpha)^2 + \sin(\alpha)$

The evaluations are: $A_{TS} = 6$, $T_{TS} = 6.5$, $U_{TS} = 3$, $V^2_{TS} = 87.25$.

The choice is simple and TS is applied. If the conflict cannot be solved with the choice of TD at the first step of the solution, TRIPS backtracks and tries AD. The conflict is not solved when there exist no applicable operators but in the conflict open list are left some. The justifications: A – [TS,AD,TD,SB,AS]; C – [(TS,TD), (AS,AD)]; OL – [TS]; C – []; A – [TS]; ML – []; COL – []

5. $1 + 3 * \cos^2(\alpha) + \sin(\alpha)$; $A_0 = 3$, $T_0 = 3.5$, $U_0 = 3$, $V_0^2 = 30.25$.

The conflict is resolved, and a hypothesis is made, that this is the solution. TRIPS checks out whether the last heuristic value is the min. In case of min, the solution is accepted as partial. Then TRIPS tries to apply new metaoperators, and if the system finds out any, everything is reiterated. When there are no more applicable metaoperators the system accepts the hypothesis as a solution, made with the available resources. The applied metaoperators are written in the metaoperator applied list.

At the end the solution is checked out in terms of overgeneralization. Overgeneralization here is defined as an implementation of a redundant logically stronger metaoperator (having more operators than needed for solving this problem). To clarify this see steps S12, L14 and S15 in the strategy algorithm. The overgeneralization redundancy check is as follows. A union is made out of all the applied lists from the steps of the partially accepted solution: <applied_1> # <applied_2> # ... # <applied_n>. In the example this is as follows: [TD] # [SB] # [AS] # [ST] → [TD,SB,AS,ST].

When any of the applied operators is repeated, it is written only once. Then this list is compared with the assigned list and an intersection of the two is made. We obtain: [TD,SB,AS,ST] & [TS,AD,TD,SB,AS] → [TD,SB,AS,ST]. When the intersecton is an empty set, this is exactly the metaoperator that is needed. In the other case the metaoperator may be overgeneralized. Then TRIPS tries to find any metaoperator from the MOL that can cover the solution list. Finally the logically weakest metaoperator from the open metaoperator list, covering the solution list is accepted. Thus the conflict with the overgeneralization is resolved.

This solution is explicit and suitable for teaching purposes because:

• it is obvious and self-evident exactly what concepts and procedural knowledge units are underlying the transitions at the different steps;

• the applied list contains not the specific machine rewrite rule, but the type of the transformation (the operator) which is a more general domain concept;

• the motivation for making a whole step is a definite type of transformation;

• each type of transformation can be explained further on with the applied rewrite rules;

• this solution can provide the reasons for applying a definite type of transformation at a given step.

Similar idea of conceptualization and control for generating the solution is used by Bundy & Welham [10], but without the strategy level and without the possibility for flexible and explicit inference.

The presentation of the procedural knowledge units into program modules (the operators, metaoperators and the strategies are separate modules), provides the opportunity of generating different possible solutions.

The multiple solutions of a given domain problem are ensured by:

• a backtrack over the justifications of a specific solution;

• applying another method for simplification (implying another algorithm) into the very same strategy;

• using another strategy for solving the overall problem.

4. Explanations. For the purposes of ITS all explanations in TRIPS should be comprehensible for the user. An explanation is a set of reasons for applying a specific transformation at a given step. The explanations are provided on the basis of previously applied transformations and domain heuristics over the dynamically generated data structures.

Now let us consider what explanations can be provided by TRIPS if we use the solution from Example 1.

•1• From the declaratively stated knowledge in TD may be derived that TD is the reason for applying TS. This conclusion can be drawn from COL_1 & AL_1 & COL_t=[] (COL_3).

•2• AS and SB may be assumed as a reason for applying TS. This can be derived on the basis of COL_1 & ML_1 & OL_1.

•3• The reason for not applying TS at the second and third step is the application of TD at the first step.

ML_2 & CL_2 & ML_3 & CL_3 & & AL_1

•4• As AS does not belong to OL_2 and AS$\in OL_3$ the reason for applying AS at the third step is the operator that causes the appearance of AS in OL_3, that is SB (Note that this may not be the operator from the previous step).

•5• When $CL_i = /[]$ the operator from the applied list at that step can be considered as a weak and unsubstantial direct reason for not applying the operators from CLi. If $Op_k \in AL_i$ and $Op_k \in COL_i$ the operator, that has generated at first the COL_i at some previous step is the indirect reason for not applying Op_k at step i. It is assumed that providing an explanation, based on the applied transformations at previous steps is more explicit than the explanation based on some preconditions, because the transformations are more direct and more natural for the process of problem solving than the preconditions, which in most of the cases are implicit and intuitive knowledge, [8].

In the experiments with TRIPS it was found out that the most appropriate approach for providing such an explanation is to use IF-THEN rules. Thus for example, the IF-THEN rule of the 3rd inference, offered here is of the form:

IF $Op_i \in ML_i$ & $Op_i \in ML_k$ & $Op_i \in CL_j$ & $Op_i \in CL_k$ & $Op_r \in AL_b$ & $b<j<k$

THEN <the reason for not applying Op_i is Op_j>

The problem is that the IF-THEN rules are domain knowledge heuristic and often it is difficult or time-consuming to discover and formalize them.

The explanations are necessary mainly for the purposes of ITS, but they could be used for other objectives as well (machine learning for example). TRIPS views problem solving as purely inferential, but the inferences it provides are explicit and flexible.

5. Contributions and future research. What we have described is a theoretical framework for a Domain Expert of an ITS and some specific experiences based on that framework. The contributions of TRIPS can be specified as follows:

- a modularised approach allowing incremental development and change;
- incorporation of flexible control into the domain expert;
- explicit knowledge representation, that could be used further for teaching purposes.

The more specific contributions are:

- a conceptualization of a class of mathematical knowledge;
- the development of knowledge bases for representing the conceptualized knowledge in a logic environment (PROLOG);
- hierarchical structuring of the Domain Knowledge providing opportunities for flexibility and explicitness;
- widespread use of procedural knowledge to guide the inference.

The problems of the realization of an overall ITS proved to be much more complex, than was expected 5-10 years ago. The computational resources needed exceed the capacity of modern uniprocessor computers. Thus besides the problems of conceptualization, the implementation aspects of ITS are quite serious.

A problem solver designed on the principles presented in this paper is not only suitable for ITS, but without any changes it can take advantage of a parallel computer architecture. The other agents of the ITS are nothing more than a process that could be scheduled at any time.

Even without the ITS architecture TRIPS points a way of taking advantages of parallelism. Each processor could be assigned a single consistent viewpoint (a strategy or a method) of the global knowledge base. TRIPS' architecture guarantees maximum information sharing.

The problem of course is not trivial because some of the internal functions of TRIPS are not local. This is just an implementation idea and a profound research is required to determine how to identify and control similar tasks.

REFERENCES

[1] DICHEV C., DOCHEV D., MARKOV Z., AGRE G., An Approach for Developing and Implementation of a Trigonometric Knowledge Base, Materials of the Second National Conference - PERSKOMP, Sofia, 21-24 april, 1987, (in Bulgarian).

[2] DICHEV C., DOCHEV D., MARKOV Z., AGRE G., Trigonometry Tutor, Proceedings of the 4th International Annual PEG Conference – Learning with AI, Uppsala, Sweden, July 16-18, 1989.

[3] XI PROLOG – Version 2.0: Reference Manual, ICTT Informa, Sofia, 1989, (in Bulgarian).

[4] DOCHEV D., DICHEV C., MARKOV Z., AGRE G., Programming in PROLOG – Basic Concepts and Applications, Nauka i Iskustvo, Sofia, 1989, (in Bulgarian).

[5] PETKOV, C., D. DOCHEV, Z. MARKOV, G. AGRE, Analysis and Formalization of a class of Mathematical Knowledge – Technological Report for the Ministry of Science and Education, Bulgaria, (1992).

[6] CARBONELL, J. R., AI in CAI: An artificial intelligence approach to computer aided instruction, *IEEE Transactions on Man-Machine Systems* **11** (4) 1970. pp. 190-202.

[7] HARTLEY, J. R., SLEEMAN, D. H., Towards Intelligent Teaching Systems, *International Journal of Man-Machine Studies* **5** 1973, pp.215-236.

[8] LARKIN, J., CHABAY, R. R., Computer Assisted Instruction and Intelligent Tutoring Systems: Shared Issues and Complimentary Approaches, Erlbaum, Hillsdale, NJ, 1990, pp. 10-37.

[9] ANDERSON, J. R., BOYLE C. F., CORBETT A. T., LEWIS M. W., Cognitive Modeling and Intelligent Tutoring, *Artificial Intelligence* **42** (1) 1990.

[10] BUNDY, A., WELHAM, B., Using Meta-level Inference for Selective Application of Multiple Rewrite Rules in Algebraic Manipulation, *Artificial Intelligence* **16** (2) 1981. pp. 189-212.

[11] BROWN, J. S., BURTON R., DEKLEER J., Pedagogical, Natural Language and Knowledge Engineering Techniques in SOPHIE I, II, and III", in D.Sleeman and J.Brown (Eds.), Intelligent Tutoring Systems, Academic Press, New York, 1982.

[12] MCDONALD, J., SUPPES P., Student Use of Interactive Theorem Prover, W. W. Bledsoe, D. W. Loveland (Eds.), Automated theorem proving: After 25 years, American Mathematical Society, 1984, pp. 315-360.

[13] WHITE, B. Y., FREDERIKSEN, J. R., Causal Model Progressions as a Foundation for Intelligent Learning Environments, *Artificial Intelligence* **42** Elsevier Science Publishers B.V. (North-Holland), 1991, pp. 99-132.

Institute of Informatics,
Bulgarian Academy of Science,
Acad. G. Bonchev St., Bl. 29A,
1113 Sofia, BULGARIA

Received 23.12.1992
Revised 29.04.1993