# Formalisation and implementation of motivational tactics in tutoring systems

Teresa del Soldato
and
Benedict du Boulay

The explicit teaching knowledge implemented in the current generation of Intelligent Tutoring Systems (ITSs) concerns mostly domain-based aspects of instructional processes, overlooking motivational aspects. This paper describes an instructional planner able to make decisions (about the next task to do, whether to provide hints, etc.) in order to achieve two goals: traversing the domain — domain-based planning — and maintaining the learner's optimal motivational state — motivational planning. The traditional ITS architecture is extended to include the activities of ***motivational state modelling*** and ***motivational planning***. For example, in motivational state modelling further learners' characteristics are diagnosed, e.g. effort and confidence. Sometimes the advice offered by a motivational planner disagrees with a domain-based plan, while in other cases both plans complement each other. A method of negotiation between the motivational plan and the domain-based plan is provided in order to arrive at a decision for action by the tutor.

## Introduction

The explicit teaching knowledge implemented in the current generation of Intelligent Tutoring Systems (ITSs) concerns mostly domain-based aspects of the instructional process, overlooking its motivational aspects. However, teachers often interweave motivational tactics with the domain-based decisions, aiming to build conditions that stimulate the ***wish*** to learn[1]. Even in systems where attention is paid to motivational issues, the theory which drives the decision making is essentially implicitly embodied in the system in contrast to the explicit representation of the domain. For instance, the coach WEST (Burton & Brown, 1982) follows pedagogical principles such as "Do not tutor on two consecutive moves, no matter what", in order to prevent excess interventions that could affect the learner's interest, independence or feeling of control[2]. However, WEST does not include in its student model an explicit model of the learner's degree of independence or feeling of control. Theories of instructional motivation elaborate the influence of issues like confidence, challenge, control and curiosity in learning processes (Keller, 1983; Malone & Lepper, 1987) and suggest instructional tactics to keep the student in an optimal learning state and provide more appealing and effective interactions. The implementation of such motivational tactics in tutoring systems requires the insertion of a ***motivational state modeller*** and a ***motivational planner*** into the system's teaching expertise (del Soldato, 1992a, 1992b).

[1] According to (Lepper, Aspinwall, Mumme, & Chabay, 1990), expert teachers include among their goals "first, to sustain and enhance their students' motivation and interest in learning, ... and second, to maintain their pupils' feelings of self-esteem and self-efficacy, even in face of difficult or impossible problems."(p. 219).

[2] The goal of such a principle is explicitly described in (Burton & Brown, 1982) as to "prevent [the coach WEST] from being oppressive" (p. 91).

The motivational planner presented here is based on the motivational tactics defined by Malone and Lepper (1987) and by Keller (1983)[3], which were formalised and implemented as production rules manipulating domain-independent teaching primitives, such as ***problem***, ***help***, ***assessment***, ***answer***, etc.

## Formalisation of motivational tactics

Whereas the motivational tactics discussed in (Malone & Lepper, 1987) and (Keller,

Therefore the motivational aspects of a student model should

## Domain-based vs. motivational-based planning

Typical domain-based planners select actions according to whether the learner ***knows*** a topic or has mastered a skill. The methodology here is twofold: detecting the current state of the learner's knowledge and skill (student modelling) and reacting appropriately in order to increase this knowledge and skill (teaching expertise). To take account of motivational factors, the twin activities of "detecting the state" and "reacting appropriately" are extended by adding a ***motivational state*** and ***motivational planning*** to the traditional ITS architecture. Sometimes the advice offered by a motivational planner disagrees with a domain-based plan, while in other cases both plans complement each other. (In a similar way Lepper et al. (1993) consider these two cases, as well as a third situation: when the motivational and the domain-based strategies are independent of each other). Here we discuss motivational planning and compare its behaviour to the decisions taken by typical domain-based planners.

### *Student succeeds performing the task*

Let us consider, first, a situation in which the student succeeds in solving a problem. A typical domain-based planner would acknowledge the right answer and suggest (or directly provide) a harder problem, thus making sure the student is traversing the domain in a progressive manner (see Table 1). Such behaviour is well exemplified by Peachey and McCalla's (1986) instructional planner: when the learner masters an instructional goal, the planner focuses next on goals that require the topic just mastered as pre-requisite, traversing the domain in the direction of a specific ultimate goal. Some domain-based planners elaborate the performance feedback according to the instructional context. The Meno-tutor (Woolf, 1984), for example, acknowledges the student's answer in three distinct modes: explicit, implicit and emphatic (adding details about the domain topic in question).

Table 1- Domain-based planner: tutor's actions when learner succeeds in solving problem

| | |
|---|---|
| comment: | performance feedback (and/or praise) |
| next prob: | next in the pre-requisite sequence (or harder) |

In this case, ***knowing*** or ***not knowing*** the topic, or exhibiting or not exhibiting the relevant skill, is the only issue in the student model that drives the selection of suitable actions, so the diagnosis methods basically aim at defining whether the student knows the topic. Such a methodology characterises more detailed domain-based instructional planners. For example, Wasson (1990) implemented a planner based on a domain network representation which links topics through a variety of relations as well as "pre-requisite", and actions like ***review, focus***, and ***re-achieve*** are selected to be executed. Such decisions, however, are based only on the assumption of student knowing (or not) topics. In some systems (e.g. see Anderson & Reiser, 1985), the student model has been improved by expanding the knowing-or-not binary state to a more graduated mastery scale, but still it is the learner's ***knowledge*** which drives instructional decisions.

Motivational planning takes into account other variables in the student model and widens the tutor's space of possible reactions. Just by considering binary states of ***effort*** (little/large) and ***confidence*** (low/ok) results in four different situations, each one

requiring a suitable set of actions from the tutor[5]. In one of the situations the motivational planner generates the same action as the domain-based planner (which corresponds to ***effort*** = ***large*** and ***confidence*** = ***ok***). Table 2 presents the four cases and the corresponding actions specified by the motivational planner.

When the student's confidence is diagnosed as being low, the major goal for the planner is to help the learner regain a

Table 3 - Domain-based planner: tutor's actions when learner fails in solving problem

| | |
|---|---|
| comment: | acknowledge (or correct) wrong answer |
| next prob: | same difficulty (or easier) |

The domain-based planner overlooks two issues:

1. Even if the student was not able to formulate a right answer, she may have spent a good deal of effort trying to perform the task.
2. If the learner is not spending much effort on the task (therefore not succeeding) the tutor should help to make the task more interesting and appealing.

The decisions described in Table 4 show possible ways to help an unsuccessful learner to restore her confidence (if she is a less confident student) or to increase her interest in the task.

Table 4 - Motivational planner: tutor's actions when learner fails in solving problem

confidence $\rightarrow$

paradoxical event is one of the tactics to stimulate cognitive curiosity (Keller, 1983; Malone & Lepper, 1987). Depending on the nature of the answer and the learner's mistake, the tutor may be able to use the wrong answer to generate a "clash" between what the student believes and what the domain model states [6].

### *Student gives up performing the task*

Producing right or wrong answers are not the only ways of

Tutors usually provide hints and clues when the student requests help. Lepper and Chabay raise the question of whether help should be ***always*** available to the student:

> "Should the tutor ***always*** intervene when the student requests help, or should some evidence of effort and independent work be demanded first?" (Lepper & Chabay, 1988, p. 248)

The approach adopted in this work is that independence should be encouraged, specially if the tutor has already intervened too much, and therefore decreased the student's feeling of control and independence over the interaction. Avoiding further interventions, at least for a while, is the most basic action to take in order to restore the learner's sense of independence. Help can be skipped in two situations:

1. if the student is requesting help in excess, or
2. if the student is lost and help should be delivered, but at the same time the tutor assumes that it has already intervened in excess[7].

However, if the ***confidence*** model is low, help should be provided in order to facilitate the learner succeeding on the task. The priority of ***confidence*** over ***independence*** assumed here is due to the fact a less confident student is eager to be helped, and less likely to feel annoyed by excessive interventions from the tutor. Examples of a motivational tutor's behaviour when the learner requests help are presented in Table 6.

Table 6 - Motivational planner: tutor's actions when learner requests help

| | confidence → | |
|---|---|---|
| independence ↓ | low | ok |
| low | (facilitate success)<br>provide: specific help | (increase independence)<br>comment: encourage independence<br>skip: providing help |
| ok | (facilitate success)<br>provide: specific help | (normal situation)<br>provide: generic help |

One can note the distinction between providing ***specific*** help (to less confident students) and providing ***generic*** help. Specific hints present more details about the problem and help the student in a more direct way, whereas generic help is "less intrusive". Delivering help of different degrees of generality is a tactic also considered by Lepper at al. (1993)[8]: "Increase or decrease the specificity of hints provided to the student as a function of the student's difficulty at a particular point" (p. 83).

The discrepancies between domain-based planning and motivational planning revealed here suggest that the inclusion of motivational tactics in a tutor's instructional planning mechanisms alter in a significant way the behaviour of the tutor.

[7] The second situation was included in the pedagogical principles of the coach WEST (2 7o79 m 38e78 tBurm 93Tc 30 6T

## Implementation of motivational tactics

The motivational tactics described above were implemented through the application of production rules to a database consisting of information about the state of the interaction, the student's progress in mastering the domain and the motivational state of the student. The set of production rules detects the

of difficulty (***same-diff)***. ***Similar problems*** should also present the same degree of difficulty, as well as require similar reasoning to be successfully performed.

Problems usually require a certain number of ***steps*** or attempts to be successfully solved (see next section). While the learner is dealing with steps towards a final answer or solution, the ***problem*** state is set as ***solving.*** When the student produces a final answer, whether the task is considered successfully performed or not generates the states ***succeeded*** or ***failed***. This is a rather simplistic classification, since complex domains include problems with many different degrees in which a solution may be considered "correct". However, the emphasis of this work does not rely on the

lost, performing the same step instead of progressing towards the solution)

state ***checked*** is set when the answer has been analysed and the system is planning its next action. ***Answers*** consist of states, contents and types presented in Table 9.

### *Assessment*

***Assessment*** is feedback provided by the tutor on whether the student's answer is right or wrong. In many systems, e.g. SCHOLAR (Carbonell, 1970) and BUGGY (Brown & Burton, 1978), positive assessment delivery may include or be replaced by a praising element such as "Very good". In this

been achieved. In this sense, comments of content ***level-promotion*** inform the student that the next tasks will get more difficult because the current topic or

incorrect responses given, and the extent to which the student's frontier of knowledge has been explored" (p.80). The Meno-tutor implementation also includes a wrong-answer-threshold (similar to the step-repetition-limit), defined as "the number of permitted wrong answers" (p. 67).

### *Confidence modelling*

Confidence is represented as a value (***conf-value)*** in a linear scale, and the limits for the lowest and the highest possible confidence values are set before the interaction with the student takes place. The confidence value is incremented and decremented in large or small (normal) steps. The values for these steps (named ***conf-inc, conf-dec, large-conf-inc, large-conf-dec***), are previously set like the confidence limits. As a trial value, the confidence limits were set as 10 and 0, the ***conf-inc*** as 1 and the ***large-conf-inc*** as 2 (and the values for ***conf-dec*** and ***large-conf-dec*** were set as -1 and -2 respectively), so that the student's confidence model at any moment during the interaction corresponds to any integer value within the range 0-10. A threshold value (***conf-threshold)*** is defined to distinguish between low and high confidence. For instance, if the ***conf-threshold*** value is set to the value 4 then ***conf-value*** 5 corresponds to a normal degree of confidence, and ***conf-value*** 3 is considered low confidence. The limits for the confidence scale and the low confidence threshold value may be altered if more precision is required.

The student's confidence model (the numerical value associated to ***conf-value***) is dynamically adjusted during the interaction according to the rules described in Table 13.

Table 13 - Confidence modelling

| rule | answer type | answer content | confidence model |
|---|---|---|---|
| C1 | ***low-conf*** | ***pos/neg*** | decrement by ***conf-dec*** |
| C2 | ***high-conf*** | ***pos/neg*** | increment by ***conf-inc*** |

| rule | steps | answer content | confidence model |
|---|---|---|---|
| C3 | none | ***help request*** | decrement by ***conf-dec*** |

| rule | problem state | with / without help | confidence model |
|---|---|---|---|
| C4 | ***succeeded*** | without ***help*** | increment by ***large-conf-inc*** |
| C5 | ***succeeded*** | with ***help*** | increment by ***conf-inc*** |
| C6 | ***failed*** | without ***help*** | decrement by ***conf-dec*** |
| C7 | ***failed*** | with ***help*** | decrement by ***large-conf-dec*** |

Rules 1 and 2 refer to the answer expression, as explained in the description of ***answer types*** (see previous section). Rule 3 reflects the case of a student asking for help from the tutor before even trying to perform the task. The four last rules concern the result of the task. If the task is accomplished, the student's confidence in future successes rises, whereas if the student failed in performing the task, the expectancy of a following success decreases. Refining this model, successes obtained completely independent of help from the tutor are likely to increase the learner's confidence in a more dramatic way than successes obtained after being helped. On the other hand, a failure despite the hints provided by the tutor saps the learner's confidence more than if the student fails but success was not facilitated in any sense.

### *Effort modelling*

Table 14 presents a model for classifying students' effort as a function of their persistence to solve the problem and requests for help to perform the task. It is assumed that persistence to solve the problem can be measured through the number of attempts to get a solution, or ***steps*** performed, so that ***many*** steps reflects a greater degree of effort from the learner. The quantification of few/many attempts is defined by the domain expert, according to each problem's level of difficulty. A value is set as a threshold between few and many steps (***few-steps-lim)***, so any quantity of attempts higher than that limit is considered ***many steps***, otherwise the student has only performed ***few steps***. Besides the number of steps performed, a student who requests hints from the tutor or accepts help offered by the tutor spends less effort than learners

amount (rule I3). On the other hand, when the

described in the next section may ***suggest*** help. Therefore rule D6 deals with the case of a ***rejected*** offer of help, in which case the tutor does nothing.

Table 16 - Domain-based planner

| rule | STUDENT MODEL / HISTORY | ACTION |
|---|---|---|
| D1 | ***problem-state*** = ***succeeded*** | provide ***assessment type right***<br>suggest ***problem type harder*** |
| D2 | ***problem-state*** = ***failed*** | provide ***assessment type wrong***<br>suggest ***problem type same-diff*** |
| D3 | ***problem-state*** = ***given-up*** | suggest ***problem type same-diff*** |
| D4 | ***problem-state*** = ***rejected*** | suggest ***problem type same-diff*** |
| D5 | ***help-state*** = ***requested*** | provide ***help content present-step*** |
| D6 | ***help-state*** = ***rejected*** | ***(help not-needed )*** |
| D7 | ***path-state*** = ***lost*** | provide ***help content next-step*** |

Table 17 - Motivational planner

| rule | student model / history | top-level tactics | tactic |
|---|---|---|---|
| M1 | ***conf-value*** < ***conf-threshold*** | — | ***increase confidence*** |
| M2 | ***effort-value*** < ***medium*** | — | ***increase effort*** |
| M3 | ***effort-value*** > ***medium*** | — | ***maintain effort*** |
| M4 | ***help-state*** = ***rejected*** | — | ***respect control*** |
| M5 | ***problem-state*** = ***given-up***<br>above giv-up-lim | — | ***respect control*** |
| M6 | ***indep-value*** < ***indep-threshold*** | not ***increase confidence*** | ***increase control*** |
| M7 | ***problem-state*** = ***succeeded*** | ***increase confidence*** | ***inc. experience success*** |
| M8 | ***problem-state*** = ***failed*** | ***increase confidence*** | ***facilitate success*** |
| M9 | ***problem-state*** = ***given-up*** | ***increase effort***<br>not ***increase confidence***<br>not ***respect control*** | ***encourage effort*** |
| M10 | ***problem-state*** = ***given-up*** | ***increase confidence***<br>not ***respect control*** | ***facilitate success*** |
| M11 | ***problem-state*** = ***succeeded*** | ***increase effort*** | ***stimulate challenge*** |
| M12 | — | ***stimulate challenge***<br>***increase confidence*** | ***emphasise promotion*** |
| M13 | ***problem-state*** = ***failed*** | ***increase effort***<br>not ***increase confidence*** | ***stimulate curiosity*** |
| M14 | ***perf-value*** = ***good*** | ***facilitate success***<br>***increase effort*** | ***remind successes*** |
| M15 | | | |

limit", defines a value for the number of times the tutor can insist on helping the students when they explicitly abandon the task. Analogous to all the other parameters in the system, the value for ***giv-up-lim*** is set for every interaction, and the trial value suggested here is 2: if the learner gives up performing the task for the second time the tutor respects the learner's decision. Therefore Rule M5 still bears a certain degree of

problems are always suggested rather than imposeder

The tutor's behaviour envisioned in table 4 is generated through rules N4, N5 and N7. Rules N4 and N5 generate actions which completely disagree with the domain-based plan, encouraging the student to keep solving the problem instead of "accepting" the learner's failure. Rules N6 and N7 refer directly to the research results obtained by Schunk (1989): facing the choice of praising both the student's performance and effort, the tutor favours the former (rule N6). Nevertheless, when large effort was spent although success was not achieved, the tutor acknowledges the student's persistence (rule N7).

When the student gives up accomplishing the task (see Table 5), one of rules N8 or N9 is activated. Whereas the domain-based plan moves to an alternative problem, the motivational plan determines that either the student's success should be facilitated (for less confident learners, rule N8) or more effort should be encouraged (for confident but not persistent learners, rule N9). In both cases, though, the negotiation planner determines that the student should be encouraged to persist in solving the problem. If the tutor is trying to facilitate the learner's success, hints are directly provided, and if the student has been successful in previous tasks, those results are flagged in order to encourage the learner's persistence (rule N10). For confident students, on the other hand, the tutor comments on the lack of effort (the tutor does get a bit demanding sometimes) and offers help. Obviously the learner may insist on abandoning the task anyway by rejecting the tutor's help, in which case the tutor moves to a new problem as the domain-based planner determines, because the motivational tactic ***respect control*** will be generated and prevents the tactic ***encourage effort*** being included in the motivational plan again.

The last six rules concern whether to intervene to help the student succeeding with the task or to skip interruptions at all, as stated in Table 6. Rule N11 simply disregards the help delivery present in the domain-based plan in order to avoid interventions as decided by the motivational planner. T

tactic -0.0945

Table 18 - Negotiation planner

| rule | DOMAIN-BASED PLAN<br>action | MOTIVATIONAL PLAN<br>tactic | NEGOTIATION PLANNER<br>delete action | <br>add action |
|---|---|---|---|---|
| N1 | suggest ***problem type harder*** | ***increase experience success***<br>not ***stimulate challenge*** | suggest ***problem type harder*** | suggest ***problem type similar*** |
| N2 | suggest ***problem type harder*** | ***stimulate challenge***<br>not ***increase confidence*** | suggest ***problem type harder*** | suggest ***problem type much-harder*** |
| N3 | suggest ***problem type harder*** | ***emphasise promotion*** | — | provide ***comment level-promotion*** |
| N4 | provide ***assessment type wrong***<br>suggest ***problem type same-diff*** | ***facilitate success*** | provide ***assessment type wrong***<br>suggest ***problem type same-diff*** | provide ***help content next-step*** |
| N5 | provide ***assessment type wrong***<br>suggest ***problem type same-diff*** | ***stimulate curiosity*** | provide ***assessment type wrong***<br>suggest ***problem type same-diff*** | provide ***help content surprise-result*** |
| N6 | provide ***assessment type right*** | ***maintain effort*** | — | provide ***comment praise-perf*** |
| N7 | provide ***assessment type wrong*** | ***maintain effort*** | — | provide ***comment praise-effort*** |
| N8 | suggest ***problem***<br>not provide ***assessment*** | ***facilitate success***<br>not ***respect control*** | suggest ***problem*** | provide ***help content next-step*** |
| N9 | suggest ***problem***<br>not provide ***assessment*** | ***encourage effort*** | suggest ***problem*** | provide ***comment trying-harder***<br>suggest ***help content next-step*** |
| N10 | — | ***remind successes*** | — | provide ***comment previous-successes*** |
| N11 | provide ***help*** | ***avoid intervention*** | provide ***help*** | skip ***help*** |
| N12 | | | | |

## Application to a concrete domain

The formalisation and implementation of motivational tactics described in this paper made use of domain independent elements (generic ***problem, help, answer***, etc.). However, evaluating the motivational planner requires its application to a concrete domain. A simple tutor for teaching Prolog debugging was designed and implemented with the purpose of being a "vehicle" for MORE[10]. In this sense, the tutor described here is simply a illustrative example of how MORE interferes in the behaviour of a tutoring system, providing the means to evaluate the motivational planner potentialities. This prototype is not meant to "compete" effectually (in domain terms) with purpose-built Prolog debugging tutors such as TADP (Brna et al., 1993).

The ***problems*** in the Prolog-debugging tutor consist of Prolog programs with bugs and the task for the student is to find and correct the bugs[11]. In this implementation the set of programs is limited to very simple programs, and each problem contains only one bug. The solution for a problem in the domain space is the correct version of the respective program.

Examples of bugs are a variable starting with a lower-case letter, a mistyped functor, or a wrong argument in a clause. Each of these bugs presents many distinct possible instances, even when one considers the application of the bug to one single program.

The level of difficulty of the ***problem*** depends on the complexity of the program combined with the degree of difficulty of the bug. In this work, the complexity of a Prolog program was defined according to Gegg-Harrison's schemata (Gegg-Harrison, 1989). The degree of difficulty of bugs, on the other hand, is not as well determined as the complexity of programs. For the purposes of the limited domain representation in this tutor, we assume that a bug of a syntactic nature, such as lower-case variable, is "easier" to detect than a semantic bug, such as a wrong argument in a clause. This assumption originates from the idea that syntactic bugs may be noticed without the need of running the program. Besides degree of difficulty, the other property for ***problems*** is similarity. ***Similar problems*** in the Prolog debugging domain consist, for example, of buggy programs generated by the application of the same bug to different programs of the same degree of difficulty.

Preliminary formative evaluation studies were performed with subjects who volunteered to interact with the Prolog-debugging tutor. The subjects were asked to report their motivational state during the interactions (e.g. level of confidence) and the interactions((e.g. 2,Tj -0.1018

***Student succeeds, with little effort but low confidence:***

This case was discussed in Table 2: the problem was "easy" to solve, so the focus of the interaction can shift towards the next level of difficulty, but because the student is not confident the system comments on the level promotion.

Had the student been feeling more confident, the system would have highlighted the increasing difficulty of the next task in a more challenging way (e.g. "The next problem will be much harder!"). One subject, who was continuously challenged by the tutor, reported being particularly stimulated by such comments ("It makes me feel more ***interested*** about the next problem ... and I don't need to check myself if the problem is too easy, the tutor tells me").

Table 19 - Level promotion

| dialogue | student model | instructional plan |
|---|---|---|
| S — ***(promptly corrects the program)*** | conf = 4 (low)<br>effort = little | |
| S — There is no bug in the program. | conf = 4<br>effort = little | provide ***assessment right***<br>comment ***level-promotion***<br>suggest ***prob harder*** |
| T — Right answer. This looks easy for you now, it's time to move to harder problems.<br>How about this program?<br>…***suggests harder problem*** | conf = 5 (ok)<br>effort = little | |

In the example provided in Table 19 (here

Table 22 - Reminding successes

| | dialogue | student model | instructional plan |
|---|---|---|---|

member(X, [X | Tail]).

member(X, [Head | Tail]) :- member(X, tail).

(Checking the head member of a list does not reveal the bug, whereas checking any element in the tail of the list results in a buggy query).

Table 24 - Stimulating the learner's curiosity

| dialogue | | student model | instructional plan |
|---|---|---|---|
| S — | ?- member(1, [1, 2, 3]). | conf = 5 (ok)<br>effort = none | |
| T — | Solution for this query:<br>yes | conf = 5<br>effort = little | |
| S — | There is no bug in the program.<br>(wrong answer) | conf = 5<br>effort = little | provide ***help surprise-result***<br>(insists on same problem) |
| T — | The solution for<br>?- member(b, [a, b, c])<br>may surprise you... | conf = 5<br>effort = little | |

### *Students request help, confidence low/ok:*

Comparing the tutor's reaction to help requests when the ***conf-value*** is above or below the ***conf-threshold***, one can note the different level of detail on the hints provided to the learner. The less confident student is given a more direct hint (Table 25) whereas the confident learner is given a more general hint (

Table 26). In the former case the problem is the buggy problem

last (Item , [Item]).

last (Item , [First | Rest]) :- last (First , Rest).

(where the bug is in the second clause) and in the confident learner case the buggy program is

last ( Item , [Rest] ).

last ( Item , [First | Rest] ) :- last ( Item , Rest ).

The independence model is not considered in the "low confidence" case, as priority is given to restoring the learner's confidence. For the confident student, the requested hint is provided since the ***indep-value*** is above the ***indep-threshold***. (In the next paragraph we discuss the situation where the ***indep-value*** is below the threshold). Generic hints were particularly appreciated by one of the subjects ("...it gives you some direction, but still lets you work").

Table 25 - Example of specific hint

| dialogue | | student model | instructional plan |
|---|---|---|---|
| S — | Help, please | conf = 4 (low) | provide ***help specific*** |
| T — | Hint: Have a look at clause 2 | conf = 4 | |

Table 26 - Example of general hint

| dialogue | | student model | instructional plan |
|---|---|---|---|
| S — | Help, please | conf = 6 (ok)<br>indep = 5 (ok) | provide ***help general*** |
| T — | Look at the original problem: there is a wrong argument somewhere | conf = 6<br>indep = 4 (low) | (help is provided: ***indep value*** is decremented) |

### *Student requests help, independence low:*

If the learner has already requested hints too many times, or the tutor has excessively intervened on its own (if the ***conf-value*** was low), resulting in a low value for the independence model, the student's help request will not be satisfied at this moment and postponed to a later help request or situation in which the 92 4.800

necessarily of a different quality to that between a student and a “machine teacher”, this raises the issue of whether tactics which work well in a

interventions in sequence (apparently WEST assumes that the learner's independence decreases after the first intervention). MORE, on the other hand, deals with the excess of interventions by having an ***explicit*** independence model, which dynamically decreases after any intervention from the tutor. In other words, WEST avoids two interventions in sequence because it could affect the learner's feeling of independence, whereas MORE represents the learner's independence through an explicit model and prevents excessive interventions when the ***independence***