In *Proceedings of the European Conference on Cognitive Modelling (ECCM-98)*, pp. 193-194.
Nottingham: University of Nottingham. http://www.dessalles.fr/papers/Dessalles_98010510.pdf

Separation of logical and calculation capabilities in a problem solving task

Jean-Bernard Auriol
Département Informatique
ENST
46, rue Barrault
75013, Paris - France
auriol@inf.enst.fr

Jean-Louis Dessalles
Département Informatique
ENST
46 rue Barrault
75013, Paris - France
dessalles@enst.fr

ABSTRACT
The distinction between declarative and procedural knowledge is a well-accepted one. However, few models offer a consistent implementation of this distinction. We present such a system, based on a strict separation of logical and calculation capabilities, designed to model aspects of human problem solving behaviour. We have tested our approach on the Tower of Hanoi task by comparing the results provided by our model with the performance of novice subjects. We also compared these results with the performance of a few other computational models. These comparisons are quite promising. Our model has been designed to be simple and psychologically plausible. Its current implementation is still basic. We expect further improvement from the joint introduction of two separate learning abilities, a logical one and a procedural one.

Keywords
problem solving, logical knowledge, procedural knowledge, calculation.

INTRODUCTION
The distinction between declarative and procedural knowledge is classical, both in psychology (Anderson 1983) and in Artificial Intelligence. In the latter, declarative knowledge is based on explicit, logical rules. Since procedural knowledge may also be implemented in rules (production rules), the conceptual distinction between procedural and declarative knowledge does not appear very clearly in computational models. Our aim is to find a computational model of human problem solving that would strictly integrate the separation between explicit, logical knowledge and procedural capabilities.

In (Johnson-Laird and Byrne, 1991), convincing evidence is presented that seems to undermine the existence of human logical capabilities. Mental models (Johnson-Laird, 83) would explain experimental results on logical problem solving tasks much better than logical models do. In particular, mental models account for biases, as observed in the Wason selection task (Evans, 1989), whereas logical models do not.

Evidence from the observation of natural conversations (Dessalles, 1993) suggest however that the ability we have to argue with each other in everyday verbal interactions relies on genuine logical capabilities. Our hypothesis is that the same logical capabilities are involved in problem solving. These capabilities, however, are limited and by no means sufficient, as illustrated by the poor performance of subjects on simple 'logical' tasks. We propose that the problem solving behaviour of subjects can be partly explained by the joint operation of two separate sets of capabilities: logical and calculation capabilities.

The nature of the procedural knowledge used in problem solving is also a subject of controversy. In (Johnson-Laird and Byrne, 1991), it is argued that classical problem solver are unable to explain human performance on logical problem solving tasks. Again, mental models are claimed to be a better explanation. Yet, designing a convincing implementation of a general mental modelling ability that could be applied to problem solving is not an easy task.

The model proposed here bears some similarity to Newell & Simon's General Problem Solver (Newell & Simon, 1972). It makes use of operators and tries to reduce the gap between what it wants and what it can do. Contrary to GPS, however, it does not proceed by mean-ends analysis. Our model works from actions up to goals by first doing what it can do and then trying to reduce the gap between the present situation and the desired one. Our approach contrasts also with J-F. Richard's (Richard et al. 1993) since we do not try to capture procedural knowledge and heuristics into explicit rules. Contrary to Richard's model, we limit explicit knowledge to goal generation.

In what follows, we first present the empirical study that we take as reference: human subjects solving the Tower of Hanoi problem. Then we illustrate how an appropriate coupling between logical and calculation capabilities may account for characteristic aspects of human protocols. We compare our system with other algorithms according to the ability to reproduce human actions. We discuss the significance of our results, and suggest ways of extending the approach through the introduction of learning abilities.

REFERENCE TASK
In order to test the validity of our approach, we analysed and tried to reproduce 40 protocols produced by seven adult individuals solving the Tower of Hanoi task. These protocols have been recorded in Poitiers University by Josiane Caron-Pargue (Caron-Pargue & Fievre, 1996), and their analysis was performed by a multidisciplinary

team[1] (Labat et al., 1996). The task is to move five disks of increasing size from peg **A**, the starting peg, to peg **C**, the target peg, with the help of peg **B**, the 'other' peg, by moving only one disk at a time, and without ever putting a bigger disk on a smaller one. At the beginning of the task, the five disks are on peg **A**, with both pegs **B** and **C** free. The disks are numbered in increasing size order (*i.e.* **1** is the smallest disk and **5** is the biggest one). We use the notation **Peg: (List of disks)** to mean that the disks listed are on peg **Peg**. As an example, **A: (1 2 3 4 5)** means that the five disks are on peg **A**.

According to (VanLehn, 1989), in such a problem solving task, subjects make use of **operators** in order to represent possible actions[2]. In our model also, all calculation capabilities will be contained in such operators. In the protocols we studied, it appears that subjects compute several steps in advance (*e.g.* "*I put* **1** *on* **C** *in order to put* **2** *on* **B**" (translated from French)). This suggests that subjects can apply operators recursively. However, as we will see, if this ability were not restricted, a recursive application of operators would solve the entire problem without errors. Yet many errors are to be observed. We must introduce a notion of **search depth**, which is the maximum number of times operators are applied in sequence.

As a consequence, operators alone are not sufficient to explain the solving process. What is missing is the ability to *evaluate* situations that can be reached through the application of operators. Subjects verbalisations (*e.g.* "*if I put* **3** *on* **5** *it gives nothing since I want to free my* **5** *in* **A**") suggest that this evaluation results from logical operations. Our fundamental hypothesis is to keep calculation capabilities, based on operators, and evaluation capabilities, based on basic logical skills, strictly apart. This radical **separation** between two kinds of knowledge worked for us as a guide line throughout the design.

Yet, the mere coupling between logical and calculation skills is not sufficient to account for observed behaviours. At some points in the solution, subjects express the feeling of being blocked ("*I am stuck*", or "*if I put it on* **B** *I cannot do anything further*"). They do not mean here that there is nothing they can do. Rather, they do not see anything interesting to do. We call these situations **dead ends**.

Rather than staying trapped in a dead end, subjects prefer to play moves that seem uninteresting. They sometimes make such moves by uttering comments like: "*I have to do something anyway*". Now the question is, when none of the possible moves seems to be interesting, how is one of them selected ? Random might be the answer. Yet random does not explain the reliable behaviour of subjects: when a given 'blocking' state is encountered twice, a subject tends to chose the same 'uninteresting' move. To account for this, we introduce the notion of **preference:** when several moves are possible, an operator will prefer to make one of them. Preferences may vary with time and experience, as we will suggest.

After a few unmotivated moves, subjects stop exploring the search space. They again express the feeling of being blocked. At this point, according to our model, they try to get out of the dead end by use of their logical skills. They identify the problematic point and try to see under which conditions the problem would disappear. They generate a **counterfactual:** a fact that they know to be false, but that they would like to see true (modal verbs and conditional is generally used ; *e.g.* "*All disks should be on the middle peg*", or more extreme: "*I should have this stack upside down to be able to take each piece to put them on C*"). A counterfactual can be seen as a hypothesis on which subjects try to build a solution.

Our aim is to reproduce the problem solving behaviour of subjects in the Tower of Hanoi problem. Even if this task is very much constrained, predicting human action in this case is far from easy. Subjects behave differently, and from one trial to the next they get more experienced. Moreover, a computational model of their behaviour must not be too powerful, in order to reproduce suboptimal behaviour. Also it must be generic and cognitively plausible: the model should not be *ad hoc* and should not postulate unlikely capabilities. We tried to comply with all these constraints in our modelling attempt.

THE MODEL

We first present the calculation model, based on reversible operators. Then we introduce the logical model, based on rule saturation detection. Lastly, we explain how both models interact by showing how they jointly solve a problem.

Calculation Capabilities

Procedural Knowledge Representation: Operators

The representation of procedural knowledge is based on operators. An operator takes the following form:

(State 1, Operation, State 2)

where **State 2** results from the application of **Operation** to **State 1**. For instance, the first move of the Tower of Hanoi task, we may have:

A: (1 2 3 4 5) B: ( ) C: ( )

1 From A to B

A: (2 3 4 5) B: (1) C: ( )

Operators are able to propose in sequence all existing legal steps from a given situation. An important feature of operators is that they can take one of the resulting states they have proposed as a new starting state. An operator can thus be applied recursively, up to its search depth. In the preceding example, with a depth of two steps, the operator would propose the following steps:

1 From A to B

1 From A to C

1 From A to B, 2 From A to C

1 From A to C, 2 From A to B

[1] In the work presented here, all aspects concerning the Tower of Hanoi task were carried out in the context of the PROVERB group managed by Josiane Caron-Pargue.

[2] Subjects, according to VanLehn, represent only what they can modify.

Preference

To account for observations, we introduce a contextual preference for operators: in a given context, the operator will propose legal steps in a certain order.

For instance, in the Tower of Hanoi task, the preference could take the following form:

Context: **From A**Preference: try **To B** first

meaning that, if the only legally moveable disk is on peg **A** and if it is legal to move it to both pegs **B** and **C**, then the operator will first propose to move it to peg **B**.

The reader will notice that preferences are a very weak form of heuristics. Preferences only give an order when exploring a search space. The preferred move is not 'better' than other possible moves, it is not even evaluated. It only happens to be proposed first.

Reversibility

Operators are reversible in two ways. Given a resulting state, an operator can propose legal steps leading to this state and the associated starting state. Given a step, an operator can propose a starting state and the corresponding resulting state. For instance, in the Tower of Hanoi task, given the step **5 From A to C**, the operator would propose:

Starting State:	**A: (5) B: (1 2 3 4) C: ( )**
Resulting State:	**A: ( ) B: (1 2 3 4) C: (5)**

Logical Capabilities

The role of the logical part of the model is to evaluate situations and design goals. Its specific form is motivated by independent studies, particularly conversation modelling (Dessalles, 1993).

Logical Knowledge Representation

Logical knowledge is represented by first-order logical rules, in an extension of the negative conjunctive normal-form[3]:

List of terms ⇒ Mod

Each term in the list is in conjunction with the rest of the list, and **Modality** (noted '**Mod**') is either **Undesirable** (noted '**Und**') or **False** (noted '**F**'). Thus:

[A, not(B), C] ⇒ Und

means that it is undesirable to have at the same time **A** true, **B** false and **C** true. In the case of the Tower of Hanoi:

[not (disk 5 on peg C)] ⇒ Und

means that it is undesirable not to have disk **5** on peg **C**. Similarly,

[disk D on peg 1, disk D on peg 2] ⇒ F

means that it is impossible to have at the same time the same disk on two different pegs. Facts are stored in memory with no specific order, in the following basic form:

(Fact, Truth Value)

where **Truth Value** can be either '**true**' or '**false**'. Facts with an unknown truth value are not stored in memory. As an example, in the case of the Towerof Hanoi, the starting position can be noted:

(disk 3 on peg A, true)

(disk 2 on peg A, true)

(disk 5 on peg A, true)

(disk 1 on peg A, true)

(disk 4 on peg A, true)

Saturation Detection

The first capability that we put forward for the logical part of our model is the systematic detection of rule saturation. A rule is said to be saturated when all the terms of the rule are known to have the truth-value with which they appear in the rule. Thus:

[A, not(B), C] ⇒ Mod

will be saturated if **A** is true, **B** false and **C** true. Depending on the modality, an undesirable or paradoxical situation will be detected in this case. Further in, we will call 'problematic situation' a situation where some rule is saturated. Rules can thus be seen as ways of detecting problematic situations.

It is important to note that rules do not have any technical effect until they are saturated. In the preceding example, if **A** is true, **B** false and **C** unknown, the rule is invisible. It is thus impossible to infer anything using it. Theoretically, we could rightfully infer **B true** from the knowledge **A** and **C true** in the preceding example. This is not allowed in our model. This does not mean that logical inferences are impossible in the model. The point is that logical inferences cannot be made by the logical part of our model alone, and that operators do have a role in what is usually called 'logical inferences' [4].

Counter-Factual Production

To get out of a problematic situation, the subject has to change the truth-value of one term of the saturated rule. This is done by producing a counterfactual. A counterfactual is a term with a truth-value that is known to be false but that cancels the problematic aspect of the current situation.

When a counterfactual is generated, the current saturated rule is of course no longer saturated. But another one might get saturated, as in:

[A, B] ⇒ Mod

[not A, C] ⇒ Mod

When **A**, **B** and **C** are true, the first rule is saturated. To get out of the problematic situation, the counter-factual **(A, false)** might be generated. But the second rule is then saturated. In those cases, the counter-factual generation can be done repeatedly until the situation is no longer

[3] A formula in negative conjunctive normal-form is false if and only if all the terms in the rule are true. So, **<A, not B, C>** is false if **A** is true, **B** is false and **C** is true.

[4] This may explain why subjects have logical capabilities and, still, make mistakes when performing logical tasks.

problematic. In this example, **(C, false)** would then be generated.

If no satisfactory state can be reached, it means that the set of rules is inconsistent. In this case, the only way to get out of the problematic situation is to change the rules.

It should be noted that, though saturated rules are stored using a last-in first-out stack in our computational model, we do not claim that subjects use such a stack. The use of a stack is merely convenient, but not necessary. Whenever the current rule is no longer saturated, the preceding saturated rule can be computed again.

Coupling logical and calculation capabilities

We introduce now the problem representation and show the kind of search performed by the model.

Problem representation

The problem representation is split into two parts. In the logical part, the situation is represented by facts. In the calculation part, the situation is represented by states. Goals are represented by undesirability rules in the logical part, and are not represented in the calculation part. Problem states are thus somehow duplicated, but goals are not.

Goal-Oriented, Preference-Oriented Exploration

The strategy used to solve the problem is to explore the search space until reaching a state where the current undesirability is no longer saturated. In order to do this, operators propose in sequence reachable states. In each proposed state, the current undesirability is checked. If the rule remains saturated, the state is rejected. If the rule is no longer saturated, the corresponding move is played. In this latter case, if there is no other saturated rule, the problem is solved. Otherwise, the search process starts again with the new current undesirability. The strategy used to solve problems can be written in the following form:

```
OPERATORS: EXPLORE PROBLEM SPACE WITHIN SEARCH
           DEPTH
IF CURRENT UNDESIRABILITY SATURATED
    CONTINUE EXPLORATION
ELSE
    PLAY PROPOSED MOVE(S)
    IF NEW UNDESIRABILITY NEW_UND
        CURRENT UNDESIRABILITY = NEW_UND
    ELSE
        STOP
```

With a restricted search depth, the set of reachable states is limited. It often happens that all of them are uninteresting. In this case, the preferred move of the operator will be played, and the search process will start again from the new state reached. After a few steps made along according to mere preference, and if no interesting state is reached, the search stops: this is a dead end.

Getting Out of Dead End: 'Counter-factual' and Operator Reversibility

A dead-end situation is characterised by the fact that the current undesirability is out of reach of the operator. To get out of a dead end, the model computes a new goal by first generating a counterfactual that 'breaks' the current saturated rule. We say that the production of a counterfactual 'breaks the rule' because it turns a saturated, and thus active, rule into an inactive, invisible one. After the introduction of a counterfactual, the situation may become paradoxical. New counterfactuals are then generated until the situation becomes neither paradoxical nor undesirable. At this point the operator is called in a reverse way: with the current situation as starting state and the counterfactual situation as ending state. The operator computes a move that will lead to the desired state. Then, it is called again with the move found as input. It returns a situation where the desired move is legal. This new situation, translated into an undesirability rule, becomes the current rule, and the search process starts again. The strategy used to get out of dead ends can be sketched this way:

```
SELECT A TERM OF THE CURRENT SATURATED RULE
INVERT THE TRUTH VALUE OF THIS TERM
IF A NEW RULE BECOMES SATURATED
    REPEAT THE PROCESS
ELSE
    CALL OPERATOR WITH
        CURRENT STATE AS STARTING STATE
        DESIRED STATE AS ENDING STATE
TURN SITUATION RETURNED BY OPERATOR
    INTO UNDESIRABILITY RULE
RE-START SEARCH PROCESS
```

A detailed example of this process is given in the next section.

HOW THE MODEL SOLVES THE TOWER OF HANOI TASK

In this section, we give more details about the implementation of goal representation, paradoxical rules and operator, in the case of the Tower of Hanoi task.

We tried to keep implementation as basic as possible. We used a stack of undesirability rules, with only one undesirability rule visible at a time. We used only one operator, which moves one disk at a time. The operator is able to propose moves with a depth of two steps. The operator is also allowed to perform his preferred move twice before counterfactual generation begins.

Logical Rules

When problem solving begins, the stack of undesirability rules is (top of the stack presented first):

$$[\textbf{not}\ \ \textbf{5 is in C}] \Rightarrow \textbf{UND}$$

$$[\textbf{not}\ \ \textbf{4 is in C}] \Rightarrow \textbf{UND}$$

$$[\textbf{not}\ \ \textbf{3 is in C}] \Rightarrow \textbf{UND}$$

$$[\textbf{not}\ \ \textbf{2 is in C}] \Rightarrow \textbf{UND}$$

$$[\textbf{not}\ \ \textbf{1 is in C}] \Rightarrow \textbf{UND}$$

All the generated goals will take the form:

$$[\textbf{not DISK in PEG}] \Rightarrow \textbf{UND}$$

We used two kinds of paradoxical rules:

$$[\textbf{DISK in PEG\_1, DISK in PEG\_2}] \Rightarrow \textbf{F}$$

$$[\textbf{not DISK in A, not DISK in B, not DISK in C}] \Rightarrow \textbf{F}$$

The first rule explains that it is impossible to have the same disk on two different pegs. The second one tells that disks have to be on one of the three pegs.

Operator

Problem states given to or returned by the operator take the following form:

Content of Peg A, Content of peg B, Content of peg C

When the system tries to escape from a dead end, the dialogue between rules and operator involves partially specified states. The set of counterfactuals constitutes a partial state definition, for instance (**4 in B**, **true**) and (**4 in A**, **false**)[5], that is sent to the operator. The operator returns a possible move **Move 4 from A to B**, and then gives a partial specification of the corresponding starting state:

(4) ( ) (1 2 3)

Disk 5 is omitted from the output of the operator because the peg on which **disk 5** is located does not interfere when **disk 4** is to be moved. The operator does not assign any place to **5**, it does not even consider this disk. The resulting undesirability that will be put on the top of the stack will be:

(not 3 in C) ⇒ UND

Disk 1 and **disk 2** are ignored here, because goal generation takes into account only the largest of the disks not already on their desired peg.

The operator's preference depends on the starting peg of the disk to be moved, and whether or not the largest disk (**disk 5**) has been put on its final destination (**peg C**). An example of preference is:

Context: **Peg A ; 5** still **in A**

Preference **:** Try **To B** first **;** Try **To C** then

It was not necessary to include the choice of the starting peg into the preference: since the operator never proposes to move the same disk twice, there is never more than one moveable disk. For instance, '**1 from B to C**' followed by '**1 from C to A'** becomes '**1 from B to A**'. It does not imply, however, that the model does not move the same disk twice. The model actually does it sometimes, but the two moves are necessarily separated by a counterfactual production.

Example of solution

At the beginning, the situation is:

(1 2 3 4 5) ( ) ( )

The top of the undesirability rule stack is:

(not 5 in C) ⇒ UND

Preferences are randomly chosen:

Context : **Peg A, 5 in A**
Preference : **To B** first**, To C** then

Context : **Peg B, 5 in A**
Preference : **To A** first**, To C** then

Context : **Peg C, 5 in A**
Preference: **To B** first**, To A** then

[5] This presupposes that **disk 4** is on **peg A**, that the current undesirability is (**not 4 in B ⇒ UND**), and that the operator cannot move **disk 4** to **peg B** within the allowed depth.

The search begins. The operator, which has a maximum depth of two steps, generates:

1 From A to B	**(2 3 4 5) (1) ( )**
1 From A to C	**(2 3 4 5) ( ) (1)**
[1 From A to B, 2 From A To C]	**(3 4 5) (1) (2)**
[1 From A to C, 2 From A To B]	**(3 4 5) (2) (1)**

None of the proposed states invalidates the current saturated rule (**[not 5 in C] ⇒ UND**), so none of the proposed steps is accepted. The operator selects its preferred move:

1 From A to B **(2 3 4 5) (1) ()**

The process starts again. The saturated rule cannot be yet invalidated, and this time the operator selects:

2 From A to C **(3 4 5) (1) (2)**

Again, the rule cannot be invalidated. But this time, the operator is no longer allowed to give its preferred move. This is a dead-end. The counterfactual generation begins:

(5 is in C, true)

which immediately leads to the saturation of:

(5 is in C, 5 is in A) ⇒ F

thus generating:

(5 is in A, false)

Now, the operator is called with the following terms as input:

A: (5) B: ( ) C: ( ) **???** **A: ( ) B: ( ) C: (5)**

The operator answers:

Move 5 From A to C

This answer is returned back to the operator, with both the starting and resulting states left to be determined. The operator now answers:

A: (5) B: (1 2 3 4) C: ( ) **A: ( ) B: (1 2 3 4) C: (5)**

This answer is then filtered to keep only the biggest disk which has different locations in desired and real situations (here, the fourth disk). The new undesirability is added on the top of the stack:

(not 4 in B) ⇒ UND

The main search process resumes with the new current undesirability.

EXPERIMENTS

Our experiment is based on the comparison between solutions given by our model and by human subjects. We performed a step by step comparison between both solutions. In order to be able to compare the solutions after the first difference in move, our solution is bound to follow the subject's solution. At each step, our model computes its next move, which we compare to the human move. The human step is always the one played. Differences are counted, and whenever the erroneous move was chosen due to operator preferences, the involved preference is inverted.

We tested the system on 40 protocols, produced by seven novice subjects, and totalling 1462 steps. We also tried different others models. Besides random strategies (pure random, random without moving the same disk twice, preferences replaced by random[6]), we experimented with a model inspired by (VanLehn, 1991).

In this latter model, three steps out of four are forced steps. Each time the model moves Disk 1, the next allowed move is to take the only other legally moveable disk and to put it on the only legal peg. Each time the model moves Disk 2, the next allowed step is to put Disk 1 back on it. The model chooses the optimal move for each unresolved move. Without correction, this algorithm gives the optimal solution.

In this paper, VanLehn did not intend to duplicate human performance at the solution level. Rather, his purpose was to study how learning takes place during problem solving. We chose to use this model as reference because it was simple to implement and gives good results.

RESULTS AND DISCUSSION

For each model, we computed the percentage of correctly predicted moves out of the total number of moves. The results obtained after these trials are:

Random:	33.68%
Random without backtrack:	68.88%
Our model without preferences:	73.76%
Inspired by VanLehn:	78.66%
Our model:	80.71%

The results given by models involving random may vary by 1%. Results given by our model also vary by 0.7% around the value we give. This last variation is due to the fact that initial preferences are fixed at random. The results of the VanLehn inspired model do not vary.

The differences between the three first models and ours are significant ($chi^2 = 15.49$, $p < 0.0005$, for the comparison between our model and the random and logic model). The difference between our model and the VanLehn inspired model is not significant ($chi^2 = 1.51$, $p < 0.25$).

The VanLehn inspired model generates by itself only the optimum solution. Thus it is not a good model of human behaviour. Our model can, and actually does, generate sub-optimum solutions. Moreover, the VanLehn inspired model makes heavy assumptions on what subject knows. The rules that determine moves three times out of four are specialised rules that an experienced subject may master, but that a novice one cannot know. This model is thus, as such, hardly plausible.

The comparison with the three random models is interesting. Our model without preferences is much better that random alone and is significantly better than random without backtrack. This confers an independent validation to the logical part of our model. Also, the results given by the complete model are better than those obtained by replacing preference by random, which indicates that preferences better account for human behaviour than random choices do.

CONCLUSION

Our results are quite promising. We have proposed a new model, based on a strict separation and coupling of two capabilities: procedural operators and limited logical capabilities. We get more than 80% of correct prediction on more than 1400 moves. Yet, further work is needed in order to establish the generality of the model and to improve it by introducing learning abilities.

A first step is to investigate how operators evolve with experience. Preferences vary over time in two ways: in a given context, they may change either as a result of statistical regularity or when leading to unsatisfactory outcomes. Also, the context itself may change, becoming more and more specific with experience. At the end of such learning processes, operators would give most of the time the best step first, which might account for expert behaviour. The architecture of our system leaves open the possibility to motivate procedural learning (here preference change) by characteristic events at the logical level. We will check whether undesirable outcomes and successive dead-end detection are appropriate times to modify operator preferences. It is also probable that subjects have expectations about the resulting state of some moves. The violation of such an expectation, leading to surprise (apparent paradox) at the logical level, could also be an adequate opportunity to learn.

Learning takes place at the logical level too. When becoming less novice, subjects change their conceptual representation. As verbalisations show in the case of the Tower of Hanoi, subjects learn to reason about groups of disks. This new ability is modelled by the same operator being called with groups of disks (*e.g.* the stack **1 2**) rather than single disks in state specifications. One of the aims of our collaborative work in the group PROVERB (Labat et al. 1996) is to understand what triggers such conceptual changes.

We presented the stack of goals as a convenient way to implement our model. The psychological plausibility of such a stack is still an open question. We will compare performance when using a stack *vs* when computing the current goal each time the preceding one has been solved.

So far, we have tested our model on the Tower of Hanoi problem only. We are currently developing another procedural context, in basic algebra, to test the generality of our approach.

We expect practical consequences from a better understanding of the relationship between procedural and logical knowledge. We believe that conceptual mistakes and calculation errors should be addressed differently in order to get a better performance remedial. We intend to show this by building a critiquing system that uses the model presented here to detect the origin of errors and to address them appropriately. Conceptual mistakes would trigger a Socratic discussion (Collins 1976, Dessalles 1993), while errors diagnosed as procedural would be remedied through an appropriate training.

[6] That is, our model where the operator preferences were replaced by random choice.

ACKNOWLEDGEMENTS
We thank Dr. Bernard Auriol for his help.

REFERENCES

Anderson, J. R. (1983). *The Architecture of Cognition*. Cambridge, USA: Harvard University Press.

Caron-Pargue, J. & Fievre, M-D. (1996). "Psycho-linguistic analysis of ten-year-old's verbal protocol". In J. Caron-Pargue & S. Gillis (ed), *Verbal production and problem solving*. Antwerp papers in linguistics 85.

Dessalles, J-L. (1993). *Modèle cognitif de la communication spontanée, appliqué à l'apprentissage des concepts - Thèse de doctorat*. Paris: ENST - 93E022.

Evans, J. S. (1989). *Bias in Human Reasoning - Causes and consequences*. Lawrence Erlbaum Associates, ed. 1990.

Johnson-Laird, P. N. (1983). *Mental Models*. London: Cambridge University Press.

Johnson-Laird, P. N. & Byrne, R. M. J. (1991). *Deduction*. Lawrence Erlbaum Associates.

Labat, J-M. et al. (1996). "Computers and human solving strategies: a comparison in the case of the tower of Hanoi puzzle". In J. Caron-Pargue & S. Gillis (ed), *Verbal production and problem solving*. Antwerp papers in linguistics 85.

Newell, A. & Simon, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

Richard, J-F., Poitrenaud, S. & Tijus, C. (1993). "Problem-solving restructuration: elimination of implicit constraints". *Cognitive Science*, *17*, 497-529.

VanLehn, K. (1989). "Problem solving and cognitive skill acquisition". In M. I. Posner (ed), *Foundations of Cognitive Science*. Cambridge: MIT Press, 527-579.

VanLehn, K. (1991). "Rule acquisition events in the discovery of problem-solving strategies". *Cognitive Science*, *15*.