Induction of High-level Behaviors from Problem-solving Traces using Machine Learning Tools

Vivien Robinet, Gilles Bisson, Mirta B. Gordon, Benoît Lemaire

Laboratoire TIMC-IMAG
Faculté de Médecine
38706 La Tronche Cedex, France
first name.last name@imag.fr

Abstract: This paper applies machine learning techniques to student modeling. It presents a method for discovering high-level student behaviors from a very large set of low-level traces corresponding to problem-solving actions in a learning environment. Basic actions are encoded into sets of domain-dependent attribute-value patterns. Then a domain-independent hierarchical clustering identifies what we call high-level abilities (HLA), yielding diagnosis expressed in natural language, addressed in principle to teachers. The method can be applied to individual students or to entire groups, like a class. We exhibit examples of this system applied to thousands of students' actions in the domain of algebraic transformations.

Keywords: Computer-assisted instruction, Machine learning, Education, Mining methods and algorithms

1 Introduction

Interactive learning environments are viewed as interesting solutions to overcome the limits of classical one-to-many teaching methods. However, these environments should incorporate accurate representations of student knowledge in order to provide relevant guidance. In a problem-solving environment, one way to build and update this student model is to precisely follow what the student is doing, by means of a detailed representation of cognitive skills. This approach is called model-tracing [1]. Some model-tracing tutors like PAT [2] contain rules that can be used to solve the problem itself and assess the student's solution, ANDES [3] is also a model-tracing tutor which follow the student stages of resolution and updates a Bayesian network after each student action to predict what the student is actually doing. Our system belongs to this model-tracing approach but there are some differences with classical systems. In particular, our domain knowledge is not at the level of the student resolution. Instead of describing the rules the student is likely to use (which provides accurate student models but is time-consuming to design), we only rely on low-level descriptions of student actions. For instance, a system based on our method would not know how to troubleshoot an electronic card, but would be able to identify the basic features of each student action. The challenge is therefore to reconstruct high-level behaviors from this low-level data.

Actually, many learning environments are able to store very detailed traces of students' activities thus producing huge sets of low-level data. However, identifying high-level behaviors from these data is not straightforward, especially if the concepts of the domain knowledge are not explicitly encoded together with the corresponding traces. In this paper we present a general approach that aims at discovering patterns of student behaviors. Its principles are applicable whenever the information carried by the traces may be split as finite sequences of \( \text{previous state, following state} \) pairs, where each \text{following state} is the result of basic student
transformations performed on the corresponding previous state.

Our approach is based on a two-steps procedure:

- a domain-dependent representation of the information carried by the traces, which encodes each {previous state, following state} pair produced by the student, as a context-action-outcome (CAO) triplet, each item (C or A or O) of the triplet being a set of attribute-value pairs.
- a domain-independent machine-learning procedure, based on a clustering technique generating the uncovered high-level patterns, that we call hereafter high-level abilities (HLA).

The output of our system are students' HLAs, which are generalizations of the CAO triplets. They are represented within the same formalism as the CAO triplets, i.e. [context, action, outcome]. Furthermore, HLAs are translated by our system into natural language expressions understandable by teachers as well as by the students themselves. HLAs might be used as inputs to a tutoring system, for instance for generating or selecting a new set of exercises, which may be eventually coupled with the learning environment. Figure 1 displays the general architecture, composed of the learning environment (1), the encoder (2) and the machine learning construction of high-level abilities (3).

![Fig. 1. General architecture of our approach](image-url)
The paper is organized as follows: in section 2 we present the general overview of our approach. The specific learning environment for algebra learning on which we demonstrate our method is presented in section 3. Section 4 presents the domain-dependent encoding procedure and the data representation. The machine-learning procedure and the results are reported on section 5. In section 6 we compare our system to some related work. Finally, we conclude and present possible extensions of our work on section 7.

2 General principles of our approach

Our system is intended to be hooked up to a large variety of learning environments that lack an intelligent tracing system. In this section we provide an example to present the general strategy used to identify the HLAs, starting with a collection of low-level traces adequately encoded as CAO triplets.

In the algebra learning context of our present application, a HLA may be, for instance, not modifying the inequation sign when moving a negative multiplicative term from one side of an equation to the other. Since this may arise systematically or just by inattention, we use a statistical approach to assess the significance of local behaviors over a large set of students' CAO triplets. Our aim is to make relevant generalizations from low-level CAO triplets to HLAs.

Let us show an example of a HLA automatically produced by our system from fifty transformations performed by a student. These transformations are mainly movements of terms in equations. A movement is a shortcut which is taught to students (at least in France) to shrink the number of resolution steps. Beginners are taught that to solve an equation such as $7x-4=3$, they have to apply the same operation to both sides (adding 4), but later in the studies they are taught that they have to "move" the $-4$ from one side to the other, while changing its sign. A diagnosis of a HLA produced by our system looks as follows: Incoherent high-level ability consisting in moving a positive term of an additive operation in an equation. This movement is performed with or without changing its sign. The final operation is additive. Before detailing our method, we first present the algebra learning environment used to collect the student traces.

3 Algebra Learning Environment

The APLUSIX learning environment [4] allows students to solve algebraic problems using an equation editor. Given algebraic equations or inequations to be solved, students using APLUSIX proceed step by step as they would do on a notebook. The only imposed constraint is that the expressions entered at any resolution step must be well formed from a syntactic point of view. Figure 2 presents a snapshot of the system, showing a proposed exercise and a student's resolution in three steps. APLUSIX stores all of the student's intermediate results, indicated on the figure as step 1 and step 2 of the resolution. Of course, the granularity of the data is variable, since the transformation from one student's step to the next one may involve implicit mental operations and/or several simultaneous algebraic transformations. For example, the second step combines two actions: the multiplicative term $-4$ was moved from the LHS to the denominator of the RHS of the equation without changing the sense of the inequality, and the fraction was simplified.
Fig. 2. Snapshot of an APLUSIX screen, showing the resolution of an exercise, through a decomposition into 2 steps. Each step may correspond to many elementary algebraic transformations.

In order to implement a systematic treatment and provide an automatic student model, we need to homogenize the granularity of the collected data. This is done by introducing whenever necessary virtual elementary steps based on the domain knowledge. In the particular case of our algebra learning environment, these steps are produced by ANAIS, a particular software developed by the APLUSIX team, which decomposes the complex student's steps into intermediate elementary steps. To this end, it contains a full set of elementary algebraic rules identified by experimented teachers and didactic experts as being usually implemented by students. These rules are algebraic transformations, that may be either correct (for instance, \((a+b)^2 \rightarrow a^2+2ab+b^2\)), or incorrect (for instance \(x^n \rightarrow n\times\)).

ANAIS strives to describe the student transformations from one step to the next one as resulting from the successive application of rules obtained through a best-first search in the space of all possible algebraic transformations in its knowledge base. Searching this path between the two student steps is guided by a distance between each intermediate equation considered and the final equation corresponding to the second student step. Computing a distance between equations as strings is quite difficult. Therefore equations are represented as trees. The search space is potentially huge and different paths are possible especially when incorrect rules are involved. This is akin to the famous assignment of credit and blame problem [5].

As a consequence, the student's production is segmented into \{previous state, following state\} pairs, where each following state stems from the corresponding previous state after the application of a single elementary transformation rule. Each pair of states is labeled as correct or incorrect according to the semantics of the rule that has been used to generate it. As a result, we obtain a consistent and homogeneous data set "enriched" with the ANAIS virtual steps. This set is the input to our system. For example, referring back to the second step of figure 2, transformation: \(-4x < 2 \rightarrow x < -1/2\), ANAIS identifies two steps:

- \(-4x < 2 \rightarrow x < 2/(-4)\) (incorrect rule applied)
- \(x < 2/(-4) \rightarrow x <-1/2\) (correct rule applied)

Thus, the two corresponding \{previous state, following state\} pairs are: \{-4x < 2, x < 2/(-4)\}, labeled incorrect, and \{x < 2/(-4), x < -1/2\} labeled correct. These are the inputs to our modeling system.
Using the terminology proposed by VanLehn [5], student steps recorded by APLUSIX correspond to physical events whereas ANAIS virtual steps correspond to knowledge events.

Data used in this paper have been collected in a large scale experiment performed in middle schools in Brazil during the fall 2003-2004. A total number of 2700 students were asked to solve between 3 and 10 algebraic problems using Aplusix. After segmentation with ANAIS, their production represents 111,258 \{previous state, following state\} pairs, corresponding to an average of 41 resolution steps per student.

4 Data Representation: the Domain-dependent Encoder

After segmentation by ANAIS we transform each domain-dependent \{previous state, following state\} pair into a generic CAO triplet of items, where:

- C (context) represents the relevant part of the previous state with respect to the semantics of the transformation performed by the student;
- A (action) represents the action itself, based on an automatic analysis of the differences between previous state and following state and using the correctness label, as described above;
- O (outcome) represents the relevant part of the following state.

Each item of the CAO triplet is encoded as a set of attribute-value pairs, in order to find regularities and identify general behaviors using machine-learning approaches. All attributes take discrete values. We organize the attributes describing these items into three categories, reflecting three different description levels in the previous and following states. These description levels are:

- argument: descriptors of the element(s) directly concerned by the transformation between the previous and the following state;
- local term (simply called term hereafter): descriptors of the elements that are in the neighborhood the argument;
- expression: descriptors of some global properties of the state.

Coming from our domain of algebra learning, here is an example (Figure 3) where we show the three descriptor levels of the previous state of an incorrect transformation.

![Figure 3](image)

In the domain of movements in algebraic equations, we defined 25 attributes to express the context, 6 for the action and 6 for the outcome, totaling 37 attributes for each CAO triplet. Table 1 contains the list of the most relevant ones for the transformation described in Figure 3. Note that non-relevant attributes are not detailed here: being unchanged by the action, they remain identical in the context and the outcome.
Table 1. Representation of a CAO triplet

<table>
<thead>
<tr>
<th>Context attributes</th>
<th>Context values</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg.side</td>
<td>right left</td>
</tr>
<tr>
<td>arg.location</td>
<td>beginning middle end alone</td>
</tr>
<tr>
<td>arg.polynomial</td>
<td>true false</td>
</tr>
<tr>
<td>arg.coefficient</td>
<td>true false</td>
</tr>
<tr>
<td>arg.implicitSign</td>
<td>true false</td>
</tr>
<tr>
<td>arg.operateur</td>
<td>+ – × /</td>
</tr>
<tr>
<td>arg.category</td>
<td>additive multiplicative</td>
</tr>
<tr>
<td>term.polynomial</td>
<td>true false</td>
</tr>
<tr>
<td>expr.type</td>
<td>equation inequation</td>
</tr>
<tr>
<td>expr.polynomial</td>
<td>true false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action attributes</th>
<th>Action values</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg.operateurChanged</td>
<td>true false</td>
</tr>
<tr>
<td>arg.categoryChanged</td>
<td>true false</td>
</tr>
<tr>
<td>arg.signChanged</td>
<td>true false</td>
</tr>
<tr>
<td>expr.typeChanged</td>
<td>true false</td>
</tr>
<tr>
<td>expr.correct</td>
<td>true false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outcome attributes</th>
<th>Outcome values</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg.operateur</td>
<td>+ – × /</td>
</tr>
<tr>
<td>arg.category</td>
<td>additive multiplicative</td>
</tr>
<tr>
<td>arg.negative</td>
<td>true false</td>
</tr>
<tr>
<td>expr.type</td>
<td>equation inequation</td>
</tr>
</tbody>
</table>

The content of Table 1 has to be interpreted as follows: the context part says that the argument is located on the left hand side of the equation, at the beginning, it is not polynomial, it is an integer with an explicit negative sign and the operator is multiplicative. The term is polynomial. The expression is a polynomial equation. The outcome part says that, after the transformation the operator of the argument is an addition, it belongs to an additive category and is positive. The expression is still an equation. The action attributes are derived from the context and outcome attributes. They indicate that the operator of the argument has been changed by the student, that its category and its sign have also been modified. However, the type of the expression remains the same. In addition, the last attribute indicates that this transformation is algebraically incorrect.

It is worth noting that many of these attributes are redundant in this example, but they are needed to describe other students’ behaviors. Our aim is to use attributes that allow the model to give a fine explanation of students’ behaviors, even if some of them are redundant. The generalization process described in next section selects which of them best explain the transformation.

Once the {previous state, following state} pairs are represented by CAO triplets, our approach can be fruitfully used to identify student high-level abilities (the HLAs) from the CAO low-level data. HLAs are generalizations of student's CAO triplets, performed by an independent module presented in the following section.
5 Discovering high-level abilities

5.1 Technique: Hierarchical Clustering

CAO triplets are the basic material used by our system to uncover high-level student's behavior. Our approach relies on an unsupervised learning algorithm to cluster similar CAO triplets into classes, the HLAs. This method may be applied to individuals as well as to whole groups of students. The goal is to get a set of a few HLA classes, representative of typical student or group behavior. We use a hierarchical clustering technique [6]. This algorithm groups together the two most similar (according to a distance detailed below cf. 5.3) CAO triplets into a working cluster that replaces the corresponding triplets. The procedure is applied again and again on the set of the remaining CAO triplets and the working clusters. The latter are candidate HLAs: they generalize the underlying CAO triplets. The algorithm stops when the closest distance between elements reaches a given threshold. This threshold is defined according to the level of generalization expected. This can be empirically done from a few examples or by setting an average number of HLAs over the student population.

Our attribute-based representation is combined with a statistical counting that keeps trace of the number of CAO triplets that share the same attribute value in the working cluster (or candidate HLA). We keep track of this statistical information to characterize the way attributes are generalized, and whether this generalization is significant or not, as we explained in part 2, the goal being to distinguish between systematic or occasional student's actions. CAO triplets have one and only one counter set to 1 for each attribute, the one corresponding to the actual value of the attribute. When we group together two triplets, the counters of the attribute values are updated. Table 2 contains an example in which the CAO triplet #12 is grouped with the working cluster #6, giving a new working cluster that generalizes (and replaces) both of them. The attribute values in the HLAs represent the numbers of CAO triplets sharing the corresponding attribute value in the cluster.

Table 2. Generalization of one CAO triplet and one working cluster producing a new working cluster

<table>
<thead>
<tr>
<th>Attributes</th>
<th>CAO triplet #12</th>
<th>Working Cluster #6</th>
<th>Working Cluster (#6&amp;#12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg.side</td>
<td>left right</td>
<td>left right</td>
<td>left right</td>
</tr>
<tr>
<td></td>
<td>0 1</td>
<td>1 3</td>
<td>1 4</td>
</tr>
<tr>
<td>arg.location</td>
<td>beg. mid. end.</td>
<td>beg. mid. end.</td>
<td>beg. mid. end.</td>
</tr>
<tr>
<td></td>
<td>0 1 0</td>
<td>0 1 3</td>
<td>0 2 3</td>
</tr>
<tr>
<td>arg.complex</td>
<td>true false</td>
<td>true false</td>
<td>true false</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>0 1</td>
<td>1 1</td>
</tr>
<tr>
<td>arg.polynomial</td>
<td>true false</td>
<td>true false</td>
<td>true false</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>3 0</td>
<td>4 0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

5.2 Different kinds of high-level abilities

A HLA is a generalization of underlying CAO triplets. The attribute "expr.correct" has a particular meaning. It is not used during the generalization process, but is very important to characterize the HLAs obtained. We distinguish two kinds of HLAs:

- coherent HLAs, that cluster together either correct or incorrect CAOs;
- incoherent HLAs which cluster together a statistically significant mixture, based on
a classical proportion comparison test \((p<0.1)\) of correct and incorrect CAO triplets. The fact that both proportions are significantly not equal to zero implies that it is probably not an isolated but rather a systematic behavior.

*Coherent* and *incoherent* HLAs were created to distinguish between possible misconceptions and careless errors that do not reflect student knowledge. These two categories correspond to what VanLehn calls *misunderstandings and slips* [5].

5.3 The distance

Comparison between pairs of CAO triplets and/or working clusters, hereafter called generically entities \(E\), are based on a linear combination of the distances between the items composing these entities, namely, the context \(C\), the action \(A\) and the outcome \(O\) parts. The distance between two items \(K_i\) and \(K_k\) \((K \in \{C, A, O\})\) is defined as follows:

\[
d(K_i, K_k) = \sum_{a(K)} \sum_{v(a)} p_a \left| f_{iav} - f_{kav} \right|
\]

With \(\sum_a p_a = 1\) \hspace{1cm} Eq (1)

where \(a(K)\) are the attributes of item \(K\), \(v(a)\) are the possible values of attribute \(a\) and \(f_{iav}\) is the frequency of value \(v\) of attribute \(a\) in entity \(i\). \(p_a\) allows us to weight the importance of attribute \(a\) in the sum: it allows to incorporate expert knowledge if this information is available. Without this information, all the \(p_a\) are set to the same value. The distance between two entities, say \(E_i\) and \(E_k\) is then

\[
D(E_i, E_k) = \alpha d(C_i, C_k) + (1-\alpha)\left(d(A_i, A_k) + d(O_i, O_k)\right)
\]

Eq (2)

where the coefficient \(\alpha\) allows us to emphasize either the weight of the context part or the action/outcome part (cf. Equation 1) in the distance between the entities. If the context part is given a large weight, the algorithm tends not to cluster entities that have distant contexts. The system is then more likely to discover incoherent behaviors (i.e. HLAs in which the student performs different actions in similar contexts). If the action and outcome parts are given a larger weight, the system does not tend to group entities with distant actions, even if contexts are similar. This leads to the discovery of coherent behaviors (i.e. HLAs in which the student performs similar actions in different contexts).

In next section we explain how the clustering is used to analyze not only the individual behavior of each student (cf. 5.4) but also to provide a snapshot of the behavior of a group of students (cf. 5.5).

5.4 Individual high-level ability

We applied the above described method on 111 258 transformations collected from 2 700 students in Brazil. Figure 4 displays the hierarchical clustering of all the transformations produced by student #1497 based on fifty transformations. We set \(\alpha\) to 0.8. The full tree (Figure 4) is shown for illustrative purposes. The algorithm was actually stopped at the dashed line that represents the chosen generality level, which corresponds to a threshold of 0.38. This threshold appears to be a good value according to the conducted tests. We use colors and shapes of the nodes to represent the most relevant attributes.

- The color indicates the correctness of the node (correct: light gray, incorrect: dark gray, both: middle gray).
- The shape represents the operator (+: triangle, -: square, *: pentagon, /: circle).

Let us describe some of the five HLAs obtained (with our threshold) in this example.
• HLA #1 corresponds to a correct and coherent behavior. Its attributes indicate that the student knows how to solve simple equations (where "argument.squareRoot", "argument.power" and "argument.fraction" are false) in which a negative term ("argument.negative"=true) has to be moved. It is the case of transformations like: \[ 6x-3=2x+4 \rightarrow 6x=2x+4+3 \] where the argument is represented in bold. The student correctly moves the argument from one side to the other, whatever its position ("argument.side" and "argument.position" are generalized) or coefficient ("arg.coefficient" is generalized); the argument of the outcome remains additive ("argument.categoryChanged"=false), but its sign has been changed ("argument.signChanged"=true).

• HLA #5 is an example of an incoherent high-level ability. In a similar case (simple equations), but with a positive argument ("argument.negative"=false), the student sometimes fails to change the sign of the argument.

![Hierarchical clustering of CAO triplets for student #1497. Five high-level abilities have been kept. Leaves contain one or more identical CAO triplets.](image_url)

Fig.4. Hierarchical clustering of CAO triplets for student #1497. Five high-level abilities have been kept. Leaves contain one or more identical CAO triplets.

An interesting question is to determine how much data is needed to obtain accurate results. This is an open question. There are no theoretical results from which we could deduce the size of the training set given the number of parameters. This information strongly depends on the distribution of the examples in the data space and thus on the shape of the clusters as well as their inter and intra inertia. These shapes could be quite complex especially if attributes are redundant. Indeed the future users of our method could tend to create redundant attributes to avoid missing regular behavior because of a lacking attribute. For example, without any attributes about the sign of a coefficient in an inequation, it would be impossible for the system to "understand" why the student sometimes changes the direction of the inequality and sometimes not.
5.5 Natural language translation

The aim of discovering HLAs is mainly to allow teachers to obtain a precise diagnosis about students. In order to produce a more legible diagnosis of each student's production, the system transforms the HLAs' attribute values into a natural language text (Figure 5), by concatenating predefined sentences. This may be a full domain-independent process provided that each attribute is described by an appropriate string. Generalization among attributes is expressed by a disjunction of the corresponding strings. In the case of a full generalization nothing is written for this attribute.

This method also generates two examples and a small comment about the coherence or incoherence for each high-level ability. Whenever the HLA is incoherent, an algorithm goes back down through the hierarchical tree until reaching the first node that clustered two coherent HLAs. It then looks for attributes that discriminate between both HLAs. These attributes are also provided because they may be correlated with the reason for the student's incoherent behavior. Here is an example of such a diagnosis generated by our system:

<table>
<thead>
<tr>
<th>HLA #40 based on 14 transformations (8 correct, 6 incorrect)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagnostic:</td>
</tr>
<tr>
<td>Incoherent high-level ability consisting in moving a positive term of an additive operation in an equation, this movement is performed with or without changing its sign. The final operation is additive.</td>
</tr>
<tr>
<td>Examples:</td>
</tr>
<tr>
<td>r-v=nx ---------&gt; r-v-nx=0</td>
</tr>
<tr>
<td>v-r=n ----------&gt; v-r+n=0</td>
</tr>
<tr>
<td>Explanation:</td>
</tr>
<tr>
<td>This student does not seem to have a coherent behavior with this context, which could be the sign of a deeper misunderstanding.</td>
</tr>
<tr>
<td>The possible causes could be:</td>
</tr>
<tr>
<td>- the term to be moved is on the right side of the equation;</td>
</tr>
<tr>
<td>- the term to be moved contains a polynomial part;</td>
</tr>
</tbody>
</table>

Fig. 5: Natural language translation of a high-level ability of student #1497. The expression "with or without" reflects the fact that the system performed a generalization of a sub-part of the student's action ("changing its sign")

Another usage of our automatic HLA discovery, currently under investigation, is to automatically generate appropriate exercises for students in case of incorrect or incoherent HLAs. For instance, the above mentioned HLA #5 would lead to the generation of an exercise in which a positive term has to be moved from one side of an equation to the other. For example: 7x+4=11x+13.

5.6 Group high-level abilities

Processing all students’ CAO triplets produced 11 026 HLAs. Their global analysis allows to identify whether some of these HLAs are shared by several students. It is not possible to simply draw a frequency chart because very few HLAs belonging to different students are strictly identical, since they result from an induction process based on the individual productions. It is thus necessary to aggregate the individual HLAs. To this end, we use the same mechanism as before because individual CAO triplets and HLAs share the same formalism. The distance threshold was set to a low value (0.1) because the goal is not to generalize HLAs but rather to smooth the differences between individual HLAs. Figure 6 displays the 38 most frequent
HLAs. The y-axis indicates the number of individual CAO triplets clustered in each HLA, with the colour representing the value of the correctness attribute. The number of students is also displayed.

Fig. 6. Histogram of the 38 most frequent HLAs. Green and red bars represent numbers of correct and incorrect CAO triplets; the blue bars are the number of students presenting the corresponding CAO triplets in their productions.

In our data, the two most frequent HLAs are correct ones. They correspond to a movement of a positive argument (HLA #11021, 1217 students) or a negative argument (HLA #11022, 1188 students). Incorrect HLAs have also been identified. For instance, HLA #10997 (565 students) is an additive movement of a negative argument from one side to the other of an inequation, without changing its sign. There are also incoherent HLAs: HLA #11023 (955 students) is an additive movement in an inequation in which the sign of a negative argument is correctly changed, but the inequation sign is sometimes also reversed, probably because of a confusion with multiplicative arguments.

These results were compared with human diagnosis made by pedagogical experts from the French project "Cognitique Algebre" who deeply analyzed some of the 2 700 student productions. These experts found a good adequacy between our HLAs and the conceptions/misconceptions they identified. However a better validation would require to run our HLAs to predict in-line what students would do and compare with what has been really done.
6 Related work

Students’ data produced by interactive learning environments are quite often huge sequences of low-level descriptions which should be automatically interpreted by changing the level of granularity [7]. Several existing systems rely on machine learning techniques to discover student knowledge behind such basic descriptions. Extracting regularities requires a rewriting of students productions in term of higher level domain-dependent attributes defined by experts.

Many systems build user models by means of supervised machine learning techniques based on predefined profiles provided by domain experts. Profile Extractor [8] induces rules from pre-classified examples, using a decision tree. Its goal is to discover preferences, needs and interests of e-learning students. Our approach is quite different since our goal is to automatically discover those profiles.

Animalwatch [9] is closer to our system. Its domain is basic arithmetic. Animalwatch analyzes a student data to predict whether she would be able to solve the problem and the time it would take her. Animalwatch uses four kinds of variables, similar to our attributes:

- **Student**: student gender, scores to initial tests;
- **Topic**: type of operator (addition, substraction, etc.) and type of operands;
- **Problem**: problem difficulty, number and difficulty of prerequisites to solve the problem (such as adding fractions, simplifying fractions), etc.
- **Context**: number of prior errors, best hint seen, etc.

After tests of several algorithms, such as a Bayesian classifier and a decision tree, the authors finally use a simple linear regression to predict the two variables.

Similarly Mrs Lindquist [10] can predict whether the current student transformation will be correct or not, using a logistic regression on factor dataset. Incorrectly predicted instances are then mined for association rules [11] to decide what operation (split, add, map) has to be applied on factors. This method leads to the creation of new factors to enrich a transfer model. The latter is then used as input for the logistic regression.

The main difference between these two systems and ours is that we are not attempting to predict performances but rather to construct a cognitive profile of the student.

Web-EasyMath [12] also relies on machine learning algorithms to construct student models in the domain of algebraic powers. The goal is to define at best a model for a new user. The student is first required to pass a test about her knowledge of the four basic operations and to assess her self-estimation on basic skills. A distance weighted k-nearest neighbor algorithm is used to asses the concept knowledge level of the new student with respect to all the students that belong to the same category.

With a more generic scope, Sison & Shimura [13] propose several features that might be used to categorize systems that discover student knowledge from their behaviors. Let us define our approach with respect to some of these features:

- **student behavior complexity** (from simple values to more complex expressions). The student behavior is undoubtedly complex in our system;
- **student behavior multiplicity** (from single behavior to multiple behaviors) We are not analyzing in depth a single behavior, our system rather considers a very large set of behaviors;
- **background knowledge construction** (from completely specified to automatically extended). In our case, the domain knowledge, either correct or incorrect, cannot be extended by the system itself.
- **student model construction** (analytic or synthetic). Our approach is synthetic because it is based on behavior generalization. However, the ANAIS software which attempts to discover intermediate resolution steps is analytic.
Finally, our system can be analyzed with respect to Mayo & Mitrovic's classification [14]. They proposed a threefold classification of existing intelligent tutoring systems:

- **expert-centric systems** in which the internal representation of the domain is mainly designed by an expert;
- **efficiency-centric systems** which are partially specified and contain parameters that allow to optimize a certain criterion (evaluation time, memory used, etc.);
- **data-centric systems** which learn their structure using mainly data.

This classification was initially specific to Bayesian student modeling, but could be easily extended to other approaches. In our case, a *high-level ability* in not a pre-defined expert object, but is constructed by a generalization process using data produced by the student. Our approach could therefore be considered in this classification as a data-centric student modeling approach.

### 7 Conclusion

This paper presents a system allowing to automatically uncover *high-level abilities* of students out of problem-solving traces produced in a learning environment. Our general purpose approach makes the system applicable to many learning domains, under the assumptions that the student actions can be represented as (context, action, outcome) triplets. The system's output is a synthesis, directly understandable by teachers or didactic experts, of the knowledge of a student or a class. The system can deal with incoherent behaviors and distinguish between occasional or systematic student errors. The results may be used for automatically generating new appropriate exercises.

The domain on which we applied our system is that of algebraic transformations, mainly additive and multiplicative movements in equations and inequations. Applications to factoring and reducing algebraic expressions are currently in progress.

Modeling student actions by means of a set of attributes is an important feature of our approach. We could have used other formalisms. For instance, in our algebra domain, student actions could have been pairs of equations represented as trees and we could have invented formalisms for representing generalized actions. However, this formalism would have been too much dependent on our domain and would not have been easily extended to other domains. Attributes are a much more general way of representing student actions, especially at the low level from which our approach can perform generalizations. This formalism allows a clear distinction between the domain knowledge and the machine-learning process of building the student's model. Attributes are obviously domain-dependent, but once they have been defined, the machine-learning mechanism is ready to operate. As a consequence, the diagnosis will be expressed in terms of the attributes, thus understandable by humans.

As we mentioned earlier a missing attribute could be a source of error in our system. This will lead to inappropriately diagnose an irregular student behavior. The solution is to define a large set of attributes even if they are redundant. Attributes do not have to be cleverly designed in order to be independent from each other; the generalization process automatically selects the attributes which best explain the student behavior, provided there are enough examples. The domain independent part of our system is based on two parameters which can be modified to match the user needs. In our case this was easily done from running a small set of examples.

Our system is based on the hypothesis that student traces are temporal sequences of states, which we know is not the case for every domain. Going from one state to the other is done by only one action, the cause of a state being the only preceding state. This is probably our strongest hypothesis, but we believe that many problem-solving learning environments are based on this hypothesis. Since this approach is based on student actions, procedural domains are probably more appropriate than declarative ones for this kind of modeling. An example...
could be a learning environment in which students are asked for filling in blank words in a text (gender agreement, verb agreement …). The model could apply here: there are context attributes (other words features, distance to the subject of the sentence …), there are correct and wrong actions, and outcomes could be deduced. Action granularity is constant which avoid the use of an additional system like ANAIS to homogenize data.

Another limitation of our approach is that it does not take into account the time at which the student is exposed to each exercise. This information may be very useful to model his time learning curves, through the analysis of which high-level abilities appear or disappear through time. This may be done through the implementation of an incremental clustering algorithm such as Cobweb [15]. The information about student steps order are needed if we want to understand the student resolution strategies. It may certainly give richer diagnosis.

Acknowledgements

It is a pleasure to thank David Renaudie [16] as well as all members of the Did@tic team of the Leibniz Laboratory for their interest in this work, and fruitful discussions. This research has been in part supported by a Cognitique grant from the French Ministry of Research.

References