Journal of Universal Computer Science, vol. 9, no. 12 (2003), 1455-200
submitted: 29/8/03, accepted: 18/9/03, appeared: 28/12/03

Skill Assessment in Problem Solving and Simulated Learning Environments [1]

Luca Stefanutti
Department of Psychology, University of Graz, Austria
luca.stefanutti@uni-graz.at

Dietrich Albert
Department of Psychology, University of Graz, Austria
dietrich.albert@uni-graz.at

Abstract: Simulated learning environments provide an efficient means for improving individual skills in specific problem solving and learning situations. One crucial aspect of an optimal system for simulated training environments is its capability to keep track of the improvements of the user along the whole training process. In this paper we present a set-theoretical formal framework that can be applied for the efficient assessment of the skills of an individual in a simulated learning environment. The basic concept underlying our approach is that of a functional skill mapping of the simulated learning environment through *problem spaces*.

Key Words: skill assessment, problem solving, simulated learning environment, knowledge structure

Category: J.4 Social and Behavioral Sciences

1 Introduction

Simulated training environments, like virtual reality or computerized training systems, provide an efficient means for improving individual skills in specific problem solving and learning situations. A simulated environment is usually cheaper and safer than a real one. Moreover, simulation allows reversibility of the user's action (i.e., the user can always 'redo' or 'undo' an action or a move), which is not always possible in a real environment. Furthermore, in a simulated environment, complex cognitive tasks can be decomposed into simpler sub-problems that are well-suited to the actual skills and competencies of the learner (see e.g., [Lee & Anderson, 2001]). This facilitates inductive/deductive reasoning, exercise and insight which, in turn, allow the user to learn new skills and competencies (or to strenghten existing ones) in the domain of the problem.

[1] A short version of this article has been presented at I-Know'03 (Graz, Austria, July 2-4, 2003)

An optimal training system is a training system which triggers and keeps this virtuous circle steady along the whole learning process.

The development of simulated training environments is a necessary, but not a sufficient condition for an optimal and efficient training of a learner in complex cognitive tasks. One crucial aspect of an optimal system for simulated training environments is its capability to keep track of the improvements of the user along the whole training process. This implies a dynamic adaptation of the system to the user's skills and performance (personalisation) so that her/his motivation and mental activity remains at an optimal level during the whole training session. One mechanism at the basis of this adaptation is performance and skill assessment and monitoring.

The basic idea underlying our approach is a functional (skill) mapping of the simulated learning/training environment. When the learner enters the simulated environment, s/he finds her/himself in some initial state, and her/his objective is to move to some final (solution) state by performing appropriate actions, operations and moves. From a cognitive perspective, when the user tackles a new problem, s/he uses a number of strategies that involve, among others, inductive and deductive reasoning, learning by trials and errors and insight. Whatever the strategies are, to solve a problem the user performs a sequence of (either mental or concrete) operations that allow her/him to move from one state to another until the final (solution) state of the problem is reached [Simon & Reed, 1976]. In the formal framework that we present in the next section a *problem space* is a set Q of *problem states* connected by operations, and a *problem* is an ordered pair $\langle a, b\rangle$ of problem states such that state b can be reached from state a through a suitable sequence of operations.

One obvious consequence is that if an individual is not capable of performing an operation (or a sub-sequence of operations) contained in this sequence then s/he will fail to solve the problem. A failure, however might occur also for other reasons. Problems can be constructed by transitivity: if $\langle a, b\rangle$ and $\langle b, c\rangle$ are both problems, then $\langle a, c\rangle$ is a problem too. We make a distinction between *optimal* and *human* problem solvers. If an optimal problem solver is able to solve both $\langle a, b\rangle$ and $\langle b, c\rangle$, then s/he always solves $\langle a, c\rangle$ by transitivity. This, however, does not hold in general for human problem solvers. In particular, we assume that — as far as a human problem solver is concerned — solving both $\langle a, b\rangle$ and $\langle b, c\rangle$ is not a sufficient condition for solving $\langle a, c\rangle$. The operation that combines $\langle a, b\rangle$ and $\langle b, c\rangle$ (or, in general, many different sub-problems) together is, itself, a mental operation (a kind of meta-operation) that occurs in the mind of the problem solver either by deductive reasoning or by insight.

In our framework a *skill* is regarded as the capability of performing a given sequence of operations, where the elementary sequences are the single operations, and the *skill state* of an individual is the collection of all the skills possessed by this individual. In the next sections we face the question of uncovering the skill state of an individual in an efficient way during the training process. Our approach can be viewed as an extension of knowledge space theory ([Albert & Lukas, 1998]; [Doignon & Falmagne, 1999]) to simulated training environments.

2 Problem spaces

A *problem space* is a labeled directed graph $\mathbf{P} := \langle Q, \Omega, v\rangle$ in which Q is a set of nodes called (problem) states, Ω a set of operations, and $v : Q \times \Omega \to Q$ a partial function specifying the edges of the graph. Nodes in the graph are labeled by elements in Q, and edges are labeled by elements in Ω.

A *string* on Ω is a sequence $\langle o_1, o_2, \ldots, o_n\rangle$ of operations in Ω. In the sequel we use lowercase Greek letters like 'σ' to denote strings on Ω, and the empty string $\langle\rangle$ is denoted by the letter ϵ, fixed throughout. The *concatenation* of two strings $\alpha = \langle a_1, a_2, \ldots, a_m\rangle$ and $\beta = \langle b_1, b_2, \ldots, b_n\rangle$ is the string $\alpha\beta = \langle a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_n\rangle$. The collection of all strings on Ω is the closure Ω^* under concatenation of Ω defined by

$$\Omega^* = \bigcup_{n\geq 0} \Omega^n.$$

Note that $\epsilon\alpha = \alpha\epsilon = \alpha$ holds for all $\alpha \in \Omega^*$.

An extension V of the function v to strings is defined (recursively) as follows: if $q \in Q$, $o \in \Omega$ and $\sigma \in \Omega^*$ then

$$V(q, \epsilon) := q; \tag{1}$$

$$V(q, o\sigma) := V[v(q, o), \sigma]. \tag{2}$$

A pair $\langle a, b\rangle$ in $Q \times Q$ is called a (solvable) *problem* in the problem space $\mathbf{P}$ if and only if $V(a, \sigma) = b$ for some $\sigma \in \Omega^*$. The set of all problems is the binary relation $P \subseteq Q \times Q$ such that, for $a, b \in Q$,

$$aPb \quad \Longleftrightarrow \quad V(a, \sigma) = b \quad \text{for some } \sigma \in \Omega^*.$$

In other words, we say that $\langle a, b\rangle$ is a problem if state b is reachable from state a through some sequence σ of operations in the graph of $\mathbf{P}$. Moreover, if $V(a, \pi) = b$ for some problem $\langle a, b\rangle$ and some string π, then we say that π *solves* problem $\langle a, b\rangle$. A string solving some problem in P is called a *path* in $\mathbf{P}$, and the collection of all problems solvable by path π is

$$V(\cdot, \pi) = \{\langle a, b\rangle \in Q^2 : V(a, \pi) = b\},$$

while the collection of all paths (i.e., all strings solving some problem) in $\mathit{\Omega}^*$ is

$$\mathit{\Pi} := \{\pi \in \mathit{\Omega}^* : V(\cdot, \pi) \neq \emptyset\}.$$

In the sequel we use the terms *path* and *skill* to denote, essentially, the same kind of objects, namely sequences of operations in $\mathit{\Omega}$ solving some problem. However, while *path* is used, more in general, to denote sequences of operations in $\mathbf{P}$, the term *skill* denotes a path that 'belongs' to some individual. This means that a path π solving some problem $\langle a, b\rangle$ represents the skill of some individual if this last is capable of solving $\langle a, b\rangle$ by means of π. In this sense, every skill is represented by some path, but the opposite does not hold in general.

Then we use the term *skill state* to denote the subset $S \subseteq \mathit{\Pi}$ of all paths that an individual is able to compute. More concretely, if S is the skill state of some individual, and $\pi \in S$ then we say that this individual is able to compute $V(a, \pi)$ for all $a \in Q$ for which $V(a, \pi)$ is defined. Thus one first assumption in our framework is that

[S1] if π is a skill in S then an individual in state S is capable of solving all problems in $V(\cdot, \pi)$.

Our second and third assumptions provide an explicit distinction between an *optimal problem solver* and a *human problem solver*.

[S2] If α and β are two actual paths, and S the skill state of an *optimal problem solver* then:

$$\alpha\beta \in S \quad \iff \quad \alpha, \beta \in S.$$

On the other hand,

[S3] if α and β are two actual paths, and S the skill state of a *human problem solver* then:

$$\alpha\beta \in S \quad \Longrightarrow \quad \alpha, \beta \in S.$$

Clearly, **[S2]** is stronger than **[S3]**. Henceforth, with the term *skill state* we refer to the set of skills of a human problem solver. Thus, $S \subseteq \Pi$ is a skill state if and only if it is consistent with **[S3]**. We use, instead, the term *optimal skill state* to refer to subsets of Π that are consistent with **[S2]**.

One key concept in our framework is that of *string inclusion*. If ω and π are strings then ω includes π (denoted by $\pi \leqslant \omega$) if $\omega = \alpha\pi\beta$ for some $\alpha, \beta \in \Omega^*$. String inclusion is reflexive, transitive, and antisymmetric, thus $\langle \Omega^*, \leqslant \rangle$ is a partially ordered set. It follows from **[S3]** that any down-set in the partially ordered set $\langle \Pi, \leqslant \rangle$ is, in fact, a skill state.

3 Skill maps and knowledge states

A *skill map* for the problem space $\mathbf{P}$ is a triplet $\langle P, \Pi, f \rangle$ where $f : P \to 2^{\Pi}$ is a mapping such that, for any $\langle a, b \rangle \in P$,

$$f(a, b) = \{\pi \in \Pi : V(a, \pi) = b\}. \tag{3}$$

The collection $f(a, b)$ is called the set of skills assigned to $\langle a, b \rangle$ and it is, simply, the set of all paths solving $\langle a, b \rangle$. If $\langle P, \Pi, f \rangle$ is a skill map for $\mathbf{P}$ and $X \subseteq \Pi$ a set of skills, then we say that $K \subseteq P$ is the *knowledge state delineated* by X if

$$K = \varphi(X) := \{\langle a, b \rangle \in P : f(a, b) \cap X \neq \emptyset\}, \tag{4}$$

where $\varphi : 2^{\Pi} \to 2^P$ is called the *disjunctive model of skill maps* ([Doignon, 1994]; [Doignon & Falmagne, 1999]), and the collection of all knowledge states delineated by subsets of Π is the image $\varphi(2^{\Pi})$. This collection of knowledge states is what, in theory of knowledge spaces, is called a *knowledge space*.

4 Skill assessment

Knowledge space theory provides efficient procedures for uncovering the knowledge state (and the corresponding skill state) of an individual in an interactive computerized session by means of a knowledge space (see, e.g., [Falmagne & Doignon, 1988]). These procedures can be applied to the framework delineated above.

The above-mentioned assessment procedures have, at least, three nice features that make them appropriate in dynamic and adaptive assessment systems:

- *efficiency*: the knowledge state of an individual can be completely recovered after the presentation of a small fraction of the whole set P of problems;
- *adaptivity*: an individual is never presented with problems that are too difficult or too easy for her/him to solve.
- the procedure is stochastic and always terminates with the best estimate of the knowledge state of an individual also in the case in which this state is not stable (i.e., the user oscillates among two or more states) during the assessment process.

A detailed presentation of these procedures is beyond the scope of this paper; we refer the reader to [Falmagne & Doignon, 1988], [Doignon & Falmagne, 1999], [Dowling & Hockemeyer, 1999] for a comprehensive introduction. We just outline here the basic concepts underlying these methods (we call them the *basic assessment procedure*).

Let K^* be the (true but unknown) knowledge state of a learner, and $\mathcal{K}$ be the knowledge space on the set P of problems used for the assessment. At the outset, with no prior information about K^*, the basic assessment procedure starts with the assumption that K^* is one of the states in $\mathcal{K}$ (we denote this by setting $\mathcal{H} = \mathcal{K}$). A first problem $\langle a, b\rangle \in P$ is presented to the learner. If the problem is successfully solved by the learner, all states $K \in \mathcal{H}$ not containing $\langle a, b\rangle$ are removed from $\mathcal{H}$. If the problem is not solved, all states $K' \in \mathcal{H}$ containing $\langle a, b\rangle$ are removed from $\mathcal{H}$. Then a new step takes place and a new problem is presented until $\mathcal{H}$ contains exactly one knowledge state. This state is the estimate of the true state K^* of the user. The problem chosen in each step is the one who maximizes the uncertainty of the observer. To clarify this, let $\mathcal{H}_n$ be the collection of knowledge states remaining in step n. Moreover, given a problem $x \in P$, let $\mathcal{H}_{n,x}$ be the collection of all states in $\mathcal{H}_n$ containing x, i.e.,

$$\mathcal{H}_{n,x} = \{K \in \mathcal{H}_n : x \in K\}.$$

Then the next problem chosen is the one for which the absolute difference

$$h_x = \left| \frac{|\mathcal{H}_{n,x}|}{|\mathcal{H}_n|} - \frac{1}{2} \right|.$$

is minimal. This difference is exactly zero when x belongs to half the number of states in $\mathcal{H}_n$. In this case, whatever the response of the learner to problem x, exactly $|\mathcal{H}_n|/2$ states will be eliminated from $\mathcal{H}_n$. If there are more than one problem minimizing this difference, then the next problem is chosen at random among them.

As an example consider the knowledge space $(P, \mathcal{K})$ on a set $P = \{1, 2, 3, 4, 5\}$ of five different problems, where

$$\mathcal{K} = \{\emptyset, \{1\}, \{2\}, \{1, 2\}, \{1, 3\}, \{1, 2, 3\}, \{1, 2, 4\}, \{1, 2, 3, 4\}, P\},$$

and suppose that the (unknown) knowledge state of the learner is $K^* = \{1, 2, 3\}$. In order to choose the first problem one needs to calculate h_x for each x in P. Table 1 shows this computation for $\mathcal{H}_0 = \mathcal{K}$. It can be seen from the table that

Item (x)	$\lvert\mathcal{H}_{0,x}\rvert$	h_x
1	7	0.28
2	6	0.17
3	4	0.06
4	3	0.17
5	1	0.39

Table 1: *Computation of h_x for choosing the first problem*

there is only one problem minimizing h_x, namely problem 3. This will be the problem presented to the learner in the first step of the procedure. Since $3 \in K^*$, the learner will solve this problem. Then, one needs to update the collection $\mathcal{H}_0$ accordingly. A new collection $\mathcal{H}_1$ is constructed containing all states K in $\mathcal{H}_0$ such that $3 \in K$:

$$\mathcal{H}_1 = \{\{1, 3\}, \{1, 2, 3\}, \{1, 2, 3, 4\}, P\}.$$

Among the remaining problems, that who minimizes h_x in this step is 4, with $h_4 = 0$. Since $4 \notin K^*$, the learner will fail this problem, thus all states in $\mathcal{H}_1$ containing this problem are removed, yielding

$$\mathcal{H}_2 = \{\{1, 3\}, \{1, 2, 3\}\}.$$

At this point, the next problem is 2 ($h_2 = 0$). Since $2 \in K^*$, the learner will solve this problem, and thus one will be left with the collection

$$\mathcal{H}_3 = \{\{1, 2, 3\}\}.$$

The procedure stops here since $\mathcal{H}_3$ contains exactly one state representing a deterministic estimate of the knowledge state of the learner. Two remarks will be done here. First, it should be noted that, in this example, the knowledge state of the learner has been recovered after three steps, i.e. after presenting problems 3,4 and 2. Problems 1 and 5 have never been presented to the learner, however the information stored in the knowledge structure $\mathcal{K}$ made it possible to infer what the response of the learner to these two problems would have been. This means that the assessment procedure exploits the dependencies among the items in order to shorten the number of questions presented to the learner.

In the second place, the procedure presented here is a deterministic one. As mentioned at the beginning of this section, there exist probabilistic assessment procedures taking into account, for instance, the probability of a careless error, or that of a lucky guess in attempting to solve a problem, and provide a probabilistic estimate of the knowledge state of a learner.

5 An example application: Towers of Hanoi

In this section we give a concrete example of how the formal framework delineated in the previous sections applies in practical problem solving contexts. Towers of Hanoi is a classical game that has a rather long tradition in experimental studies on human problem solving (see e.g., [Ewert & Lambert, 1932]; [Gagne & Smith, 1962]; [Simon, 1975]; [Goel & Grafman, 1995]; [Anderson & Douglas, 2002]). Thus, we take it as a first example application of our formal framework. The psychological importance of this game resides in that its solution requires that some well defined planning and sub-goaling strategies are mastered by the problem solver.

Basically, the rules of the game are rather easy to understand. There are three pegs called source (1), temporary (2) and destination (3) pegs, and a number of disks (say, 4) of decreasing diameter stacked on the first peg, forming a tower. Figure 1 depicts a schematic representation of the game. The objective of the game is to move the tower of disks from the source peg to the destination peg under the following constraints: (i) in each single move only one disk can be moved from a peg to another; (ii) only the top disks of the pegs can be moved; (iii) a disk cannot be placed over a smaller disk.

A single move of the game is represented by a pair (i, j), where $i, j \in \{1, 2, 3\}$, meaning that the top disk of peg i is moved to the top of peg j. Thus, what-

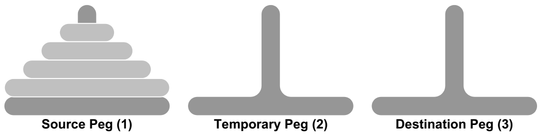

Figure 1: *Schematic representation of the "Towers of Hanoi" with four disks.*

ever the number of disks, with a number of three pegs there are in the whole 6 different moves. The set of all these moves is denoted by

$$M := \{(1,2),(1,3),(2,1),(2,3),(3,1),(3,2)\}.$$

Note however that not all moves are allowed in all circumstances because of constraint (iii) (we will come back to this point later).

The basic strategy to solve the problem with an arbitrary number n of disks (called the *goal recursion strategy*) is rather simple and its application is an example of recursive sub-goaling (Simons, 1975). Given a tower of n disks, the $(n-1)$-th sub-tower is the stack of $n-1$ disks lying over the larger disk in the original tower — mathematically, a tower can be viewed as a linear order T_n of disks, and the $(n-1)$-th subtower $T_{n-1} \subset T_n$ is simply obtained by removing the larger element from T_n. Then, the goal recursion strategy (GRS) can be stated as follows: to move a tower T_n from peg i to peg j having an auxiliary peg k,

(1) if $T_{n-1} \neq \emptyset$ move tower T_{n-1} from i to k;

(2) move the larger disk $l \in T_n \setminus T_{n-1}$ from i to j;

(3) if $T_{n-1} \neq \emptyset$ move tower T_{n-1} from k to j.

The goal recursion strategy is clearly recursive, as it calls itself in both steps (1) and (2), and the number of moves required to solve a problem with n disks amounts to $2^n - 1$.

The three steps of the GRS consist basically of two kinds of operations: moving a sub-tower from one peg to another, and moving a disk from one peg to another. Any operation of the goal recursion strategy is thus captured by the notation $t_n(i,j)$ and $d_n(i,j)$, for $i,j \in \{1,2,3\}$, where t_n means *moving subtower* $n-1$ (steps (1) and (3) of the GRS), and d_n means *moving disk n* (step (2) of the GRS).

Thus, for instance, $t_4(3, 2)$ describes an operation in which a subtower of four disks is moved from peg 3 to peg 2.

A problem space corresponding to an application of the GRS to the towers of Hanoi with n disks is depicted in Figure 2. Note that, in order to simplify the

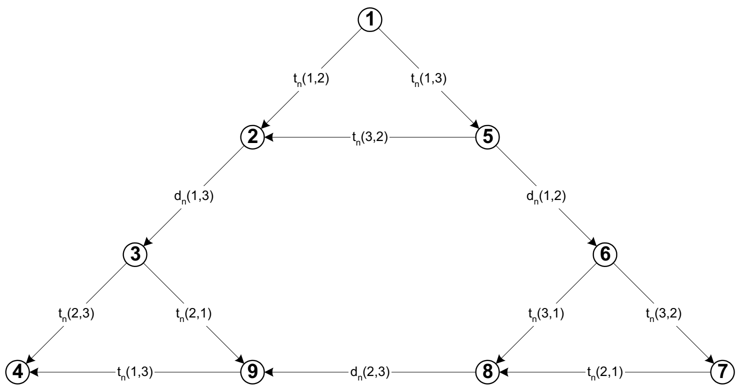

Figure 2: *Problem space corresponding to an application of the goal recursion strategy to the towers of hanoi with n disks. Numbered circles denote states of the problem, and labeled edges denote operations.*

diagram, not all possible operations are represented in the figure. In particuar, for each operation $t_n(i, j)$ (resp. $d_n(i, j)$) displayed in the diagram, there exists a reverse operation $t_n(j, i)$ (resp. $d_n(j, i)$) not displayed in the diagram. The problem space contains 9 different states, represented by the numbered circles in the diagram. Each single state is a particular configuration of the problem. For instance, in state 1 all disks are stacked on peg 1. This is the usual initial state of the problem. In states 4 and 7 all disks are staked, respectively, in pegs 3 and 2. The remaining states can be viewed as intermediate configurations of the problem. The problem space of Figure 2 is valid for any number of disks. In this sense, for each level of recursion of the GRS there is a problem space like that depicted in the figure.

As usual, subproblems are represented by pairs (x, y) of states such that y is reachable from x through a suitable sequence of operations. If one takes into account both operations displayed in the diagram and those not displayed (the *reverse operations*), in this problem space any state can be reached from any other. Moreover, some of the problems can be solved through alternative sequences of operations like for instance problem $\langle 1, 4\rangle$. In fact, both paths

$$\langle t_n(1,2), d_n(1,3), t_n(2,3)\rangle$$

and

$$\langle t_n(1,3), d_n(1,2), t_n(3,1), d_n(2,3), t_n(1,3)\rangle$$

solve this problem.

In our example application we focus on a small portion of this problem space. We will consider the sub-space consisting of the four states 1, 2, 3 and 4 and the three operations $t_n(1,2)$, $d_n(1,3)$, $t_n(2,3)$. To simplify notation we rename the three operations as follows: $a_n \equiv t_n(1,2)$, $b_n \equiv d_n(1,3)$, $c_n \equiv t_n(2,3)$. Thus, the set of states is $Q_n = \{1,2,3,4\}$ and the set of operations is $\Omega_n = \{a_n, b_n, c_n\}$. In this problem space there are in the whole 6 different problems: $\{\langle 1,2\rangle, \langle 1,3\rangle, \langle 1,4\rangle, \langle 2,3\rangle, \langle 2,4\rangle, \langle 3,4\rangle\}$. Dropping the subscript n and skipping the empty path ϵ for convenience, the set of all (nonempty) paths is

$$\Pi = \{a, b, c, ab, bc, abc\}.$$

An application of the approach presented in sections 2 and 3 yields the skill space displayed in Figure 3. Each problem in the portion of the problem space

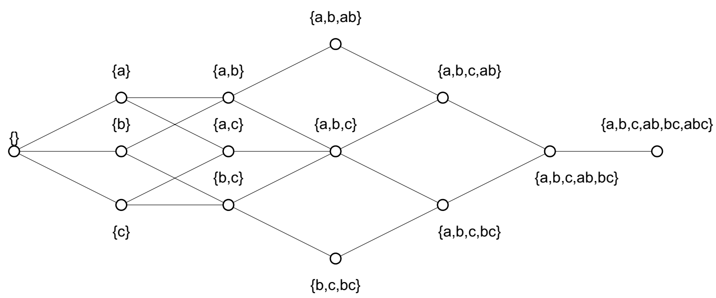

Figure 3: *Skill space for the portion of the problem space of Towers of Hanoi considered in the text.*

considered here can be solved by exactly one solution path. As a consequence, there is a one-to-one correspondence between the skill space displayed in Figure 3 and the knowledge space that can be derived from it.

Suppose now a learner is in the (unknown) skill state $S = \{a, b, c, ab\}$. The (unknown) knowledge state of this person is $K = \{\langle 1, 2\rangle, \langle 2, 3\rangle, \langle 3, 4\rangle, \langle 1, 3\rangle\}$. This is the knowledge state corresponding to S. At the outset, with no prior information on the knowledge state of the learner, the estimated skill state is $\mathcal{H}_0 = \mathcal{K}$ where $\mathcal{K}$ denotes the skill space displayed in figure 3.

The assessment procedure then works in the following way: the learner is presented with a first problem, say $\langle 1, 2\rangle$. This problem corresponds to skill a. Since $a \in S$ we assume that the learner solves this first problem and thus all skill states not containing a are removed from $\mathcal{H}_0$ obtaining thus

$$\begin{aligned}\mathcal{H}_1 := \{&\{a\}, \{a, b\}, \{a, c\}, \{a, b, c\}, \{a, b, ab\}, \{a, b, c, ab\},\\ &\{a, b, c, bc\}, \{a, b, c, ab, bc\}, \Pi\}.\end{aligned}$$

At this point the learner is presented with a second problem, say $\langle 1, 3\rangle$. This problem corresponds to skill ab. Again, this skill is in the state of the learner, thus all states in $\mathcal{H}_1$ not containing ab are removed. This way the following collection is obtained:

$$\mathcal{H}_2 := \{\{a, b, ab\}, \{a, b, c, ab\}, \{a, b, c, ab, bc\}, \Pi\}.$$

The next problem is $\langle 2, 4\rangle$ whose corresponding skill is bc. This skill is not in the state of the learner, thus all states *containing* bc are removed from $\mathcal{H}_2$ obtaining

$$\mathcal{H}_3 := \{\{a, b, ab\}, \{a, b, c, ab\}\}.$$

The fourth question is $\langle 3, 4\rangle$. The corresponding skill (c) is in S, thus state $\{a, b, ab\}$ is removed from $\mathcal{H}_3$ and we are left with a collection containing exactly one state, namely $\mathcal{H}_4 = \{S\}$.

It should be noted that the skill state of the learner was uncovered after four problem presentations, i.e. it was not necessary to present all six problems to the learner. This feature of the assessment procedure becomes a real advantage when the number of problems is large. In this case one expects to uncover the skill state of the learner after presenting a small fraction of the whole set of problems.

The result of the assessment can be used to train the learner on specific problems in order to improve her/his skill state. If Figure 3 is considered, the skill state $\{a, b, c, ab, bc\}$ comes immediately after S in terms of set inclusion. The difference between these two sets is $\{bc\}$, and skill bc corresponds to problem $\langle 2, 4\rangle$. Training should focus, then, on this problem.

6 Conclusion

A set-theoretical formal framework for the efficient assessment of the skills of an individual in a simulated learning environment has been presented. The basic idea underlying this approach is a skill mapping of the simulated learning/training environment. A specific model, called a 'problem space', for the learning environment is constructed, and a skill and a knowledge space are derived from this model. The skill space and the knowledge space can then be used for assessment and training purposes in the simulated environment. An example application to the problem of 'Towers of Hanoi' has been shown.

One advantage of the presented models is that a knowledge space can be derived by automatic procedures from any problem space. In this sense, one only needs to specify the problem space itself for a given learning environment. Semi-automatic procedures for the construction of a problem space are the subject of future work.

On the other hand, in order to apply these models, one needs to specify a learning environment in a rather highly structured way. In this sense, the proposed models are well-suited to some kinds of domains, in which the structure of a problem can be clearly specified in terms of discrete states and operations on these states.

References

[Albert & Lukas, 1998] Albert, D., and Lukas, J. (1998). *Knowledge Spaces: Theories, Empirical Research, Applications.* Mahwah, NJ: Lawrence Erlbaum Associates.

[Anderson & Douglas, 2002] Anderson, J.R., and Douglas, S. (2002). Tower of Hanoi: evidence for the cost of goal retrieval. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 27(6).*

[Doignon, 1994] Doignon, J.-P. (1994). Knowledge spaces and skill assignments. In G.H. Fischer and D. Laming (Eds.), *Contributions to Mathematical Psychology, Psychometrics, and Methodology*, (pp. 111–121). New York: Springer-Verlag.

[Doignon & Falmagne, 1999] Doignon, J.-P.., and Falmagne, J.-C. (1999). *Knowledge Spaces.* Berlin, Heidelberg: Springer-Verlag.

[Dowling & Hockemeyer, 1999] Dowling, C.E., and Hockemeyer, C. (1999). Automata for the assessment of knowledge. *IEEE Transactions on Knowledge and Data Engineering*, to appear.

[Ewert & Lambert] Ewert, P.H., and Lambert, J.F. (1932). Part II: The effect of verbal instructions upon the formation of a concept. *Journal of General Psychology, 6*, 400-413.

[Falmagne & Doignon, 1988] Falmagne, J.-C., and Doignon, J.-P. (1988). A Markovian procedure for assessing the state of a system. *Journal of Mathematical Psychology, 32*, 232–258.

[Gagne & Smith, 1962] Gagne, R.M., and Smith, E.C. (1962). A study of the effects of verbalization on problem solving. *Journal of Experimental Psychology, 63*, 12-18.

[Goel & Grafman, 1995] Goel, V., and Grafman, J. (1995) Are the frontal lobes implicated in ”planning” functions? Interpreting data from the Tower of Hanoi. *Neuropsychologica, 33*, 623-642.

[Lee & Anderson, 2001] Lee, F.J., and Anderson, J.R. (2001). Does learning a complex task have to be complex? A study in learning decomposition. *Cognitive Psychology, 42*, 267-316.

[Simon, 1975] Simon, H.A. (1975). The functional equivalence of problem solving skills. *Cognitive Psychology, 7*, 268-288.

[Simon & Reed, 1976] Simon, H.A., and Reed, S.K. (1976). Modelling strategy shifts in a problem-solving task. *Cognitive Psychology, 8*, 86–97.