

Automata for the Assessment of Knowledge

Cornelia E. Dowling and Cord Hockemeyer, *Member, IEEE*

Abstract—The results of this paper can be applied to construct efficient algorithms for the adaptive assessment of the students' knowledge. The problem of knowledge assessment is a special case of the problem of assessing the state of a system. These results are obtained suggesting a new assessment algorithm whose formal equivalence to previously suggested assessment algorithms is proven. A simulation study illustrates the vast improvements of efficiency with the new algorithm.

Index Terms—Knowledge assessment, deterministic finite automaton, knowledge space, knowledge state, assessment language.

1 INTRODUCTION

A teacher assesses a student's knowledge in a specified field by posing a suitable first question from that field to the student, by checking whether or not the student's answer is correct, and by selecting a new question dependent on the student's answer. The teacher selects all further questions corresponding to the student's answers to previous questions. In that manner, questions which are too difficult or too easy for the individual student are avoided. The querying stops whenever the teacher has sufficient information on the student's state of knowledge. For computer-aided instruction, the teacher's adaptive assessment of a student's knowledge can be mimicked using algorithms for the assessment of knowledge based on prerequisite relationships between different learning objectives and dichotomous information on students' mastery of these learning objectives.

A theoretical framework for describing such algorithms for the adaptive assessment of knowledge is the theory of knowledge spaces developed by Doignon and Falmagne [1], [2]. This framework uses prerequisite relationships between learning objectives for the assessment. Other researchers in artificial intelligence rather try to take advantage of misconceptions revealed by wrong answers; notice that listing a priori such misconceptions and their interpretations can be quite a demanding task.

The problem of assessing a student's state of knowledge is a special case of the problem of assessing the state of a system that can be represented by specified subsets of features from a fixed set. Other applications of this general problem are failure analysis, medical diagnosis, and pattern recognition. The algorithms for the assessment of knowledge can be applied to all such problems [3].

The following example illustrates learning objectives and their prerequisite relationships. A learning objective will be

called *item* in the sequel. We assume that the mastery of each item can be tested with the help of a corresponding question. In this paper, we also assume that a student can answer such a question either correctly or incorrectly; an approach for including multiple levels of correctness has been suggested in [4].

Example 1. For simplicity, we show four items from a larger project with 77 items from the field of fractions [5]. We will refer to this example throughout the paper:

- a. Being able to find the least common multiple,
- b. Being able to add mixed numbers,
- c. Being able to add fractions with different denominators, and
- d. Being able to subtract fractions with different denominators.

Prerequisite relationships between the items are determined by querying teachers. For the four items above, these prerequisite relationships are represented by the and-or-graph in Fig. 1a. The prerequisite relationships and the graph correspond to the following two statements:

1. Mastering at least one of the items *b* or *c* is a prerequisite for mastering item *d*.
2. Mastering item *a* is a prerequisite for mastering item *c*.

These prerequisite relationships can be used for the adaptive assessment of knowledge as it is illustrated in Figs. 1b and 1c. We assume that the prerequisite relationships between the items shown in Fig. 1a are valid and that a student's knowledge on the four items above is to be tested. In Fig. 1b, we assume that a question testing the mastery of item *c* is presented to the student first and that the student answers it correctly. From this fact, we can infer that the student would also master item *a* if it were presented to him. We assume now in Fig. 1c that *d* is the second item tested and that it is also mastered. Fig. 1d illustrates the third test with item *b*, and the fact that it is not mastered. From the latter two tests, no further inferences can be made. The student's knowledge state consists, therefore, of the items *a*, *c*, and *d*.

The main advantage of such an adaptive assessment of knowledge is that the number of questions posed to the student is often far smaller than the number of items in the

• C.E. Dowling is with the Institut für Psychologie, Technische Universität Braunschweig, Spielmannstraße 12a, D-38106 Braunschweig, Germany. E-mail: C.Dowling@tu-bs.de.

• C. Hockemeyer is with the Institut für Psychologie, Karl-Franzens-Universität Graz, Universitätsplatz 2/III, A-8010 Graz, Austria. E-mail: C.Hockemeyer@computer.org.

Manuscript received 26 Nov. 1997; revised 11 Jan. 1999; accepted 11 Oct. 1999.

For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 106137.

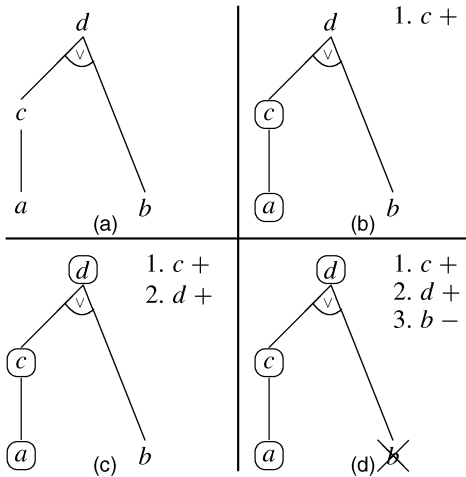


Fig. 1. Example assessment in an and-or-graph.

complete item pool. In contrast to the rather small reduction of the number of questions in this example, we obtain considerable reductions for larger item pools. Examples will be given by the results of the simulation study in Section 4.

An assessment procedure poses questions to the individual student and determines the student's state of knowledge. At each step of the procedure, a *questioning rule* is used to select the next item to be tested. Dependent on the student's performance, the state of the assessment system is changed using an *updating rule* [6], [3]. We describe the latter within the framework of the theory of deterministic finite automata. In particular, we define automata based on different representations of knowledge spaces and prove that these automata are equivalent. This is the main result of this paper. It is applied to render assessment procedures more efficient.

In Section 2, basic concepts of knowledge space theory and the theory of automata are introduced, and the updating rule published in [3] is formalized as an automaton. In Section 3, we introduce a more efficient automaton for the assessment of knowledge and prove its equivalence to the former automaton. In Section 4, we show the results of a simulation study comparing implementations of the resulting algorithms. Section 5 contains the proofs of the theorems from Section 3.

2 BASIC CONCEPTS

2.1 Knowledge Space Theory

This theory assumes that a finite set of items Q is given. Doignon and Falmagne [1], [2] define a *knowledge state* as a subset of the set of items which can be mastered by some potential student. In practice, the number of knowledge states is far smaller than the number of possible subsets of the item set since the set of knowledge states is restricted by prerequisite relationships between these items.

Example 2. Assume that $Q = \{a, b, c, d\}$ consists of the items from Example 1, and that the prerequisite relationships from Fig. 1a hold. The subset $\{c, d\}$, e.g., is not a knowledge state since it contains item c , but not its prerequisite a . Similarly, the subset $\{a, d\}$ is not a

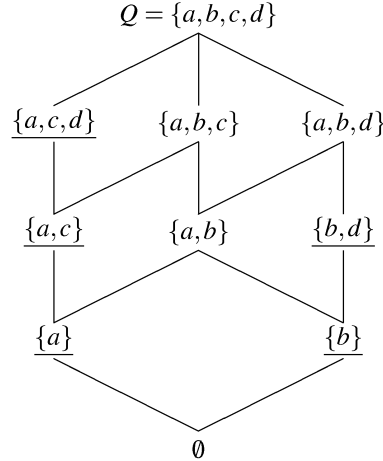


Fig. 2. The Hasse diagram of the example knowledge space.

knowledge state either, since it contains item d , but none of the prerequisites c and b . Knowledge states, i.e., subsets of items which do not contradict the prerequisite relationships, are \emptyset , $\{a\}$, $\{b\}$, $\{a, b\}$, $\{a, c\}$, $\{b, d\}$, $\{a, b, c\}$, $\{a, b, d\}$, $\{a, c, d\}$, and $Q = \{a, b, c, d\}$.

A family \mathcal{K} of subsets of a finite set of items Q is a *knowledge space* on Q whenever it has the empty set \emptyset and the set Q itself as elements, and has the property that it is closed under union,¹ i.e., for all $S \subseteq \mathcal{K}$, we have $\bigcup S \in \mathcal{K}$. A knowledge space consists of the set of all knowledge states [1]. Fig. 2 shows the Hasse diagram of the set of knowledge states from Example 2, which has the properties of a knowledge space. The knowledge states underlined in Fig. 2 will be explained soon, in Example 2. The subset of all knowledge states in \mathcal{K} that contain a question $q \in Q$ is written as \mathcal{K}_q . The subset of states not containing question $q \in Q$ is denoted by $\mathcal{K}_{\bar{q}}$.

For a knowledge space \mathcal{K} on Q , let \mathcal{B} be the subset of \mathcal{K} defined by setting

$$X \in \mathcal{B} \quad \text{if and only if, for any } \mathcal{L} \subseteq \mathcal{K}, \\ (X = \bigcup \mathcal{L} \text{ implies } X \in \mathcal{L}).$$

This family \mathcal{B} is called the *basis* of the knowledge space \mathcal{K} . For a given knowledge space \mathcal{K} and an item $q \in Q$, we call a minimal member $X \in \mathcal{K}_q$ *minimal for q* . The basis of a knowledge space consists, therefore, of exactly those of its knowledge states that are minimal for some $q \in Q$. The set of all members of the basis that contain an item $q \in Q$ is written as \mathcal{B}_q and the set of basis elements not containing item q is denoted by $\mathcal{B}_{\bar{q}}$. A knowledge space can be reconstructed by closing its basis under union [1].

Example 3. Let $\mathcal{K} = \{\emptyset, \{a\}, \{b\}, \{a, b\}, \{a, c\}, \{b, d\}, \{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \text{ and } Q\}$ be the knowledge space from Example 2. For $q = d$, we obtain $\mathcal{K}_d = \{\{b, d\}, \{a, b, d\}, \{a, c, d\}, Q\}$ and $\mathcal{K}_{\bar{d}} = \{\emptyset, \{a\}, \{b\}, \{a, b\}, \{a, c\}, \text{ and } \{a, b, c\}\}$. The basis of this knowledge space is the family $\mathcal{B} = \{\{a\}, \{b\}, \{a, c\}, \{b, d\}, \text{ and } \{a, c, d\}\}$. The elements of the basis are underlined in Fig. 2.

1. For a family S of sets, we write their union as $\bigcup S = \bigcup_{X \in S} X$ and their intersection as $\bigcap S = \bigcap_{X \in S} X$.

2.2 Deterministic Automata

A finite nonempty set Σ of symbols is called an *alphabet*. For a given alphabet Σ and a natural number n , a sequence $\mathbf{x} = a_1 a_2 \dots a_n$ of symbols $a_1, a_2, \dots, a_n \in \Sigma$ is a *word of length n* over the alphabet Σ . For an alphabet Σ , the *empty word*, λ , is the unique word of length $n = 0$. A set of words over an alphabet Σ is called a *language*. The set of all words over an alphabet Σ is denoted as Σ^* .

A *deterministic finite automaton* D is a quintuple $(\Sigma, \mathbf{S}, \mathcal{S}_0, u, \mathbf{F})$, where

1. Σ is the *input alphabet*.
2. \mathbf{S} is a finite nonempty set of *states*.
3. $\mathcal{S}_0 \in \mathbf{S}$ is the *initial state*.
4. $u : \mathbf{S} \times \Sigma \rightarrow \mathbf{S}$ is the *state transition function*, and
5. $\mathbf{F} \subseteq \mathbf{S}$ is the set of *final states*.

Note that we have to distinguish between two different type of states, knowledge states and the states of an automaton. In the sequel, we will use the abbreviation *automaton* instead of deterministic finite automaton. For an automaton $D = (\Sigma, \mathbf{S}, \mathcal{S}_0, u, \mathbf{F})$, the *extended state transition function* \bar{u} for D is a function $\bar{u} : \mathbf{S} \times \Sigma^* \rightarrow \mathbf{S}$ defined recursively as follows for all $\mathcal{S} \in \mathbf{S}$, $v \in \Sigma$, $\mathbf{x} \in \Sigma^*$:

1. $\bar{u}(\mathcal{S}, v) = u(\mathcal{S}, v)$,
2. $\bar{u}(\mathcal{S}, \lambda) = \mathcal{S}$, and
3. $\bar{u}(\mathcal{S}, v\mathbf{x}) = \bar{u}(u(\mathcal{S}, v), \mathbf{x})$.

An automaton $D = (\Sigma, \mathbf{S}, \mathcal{S}_0, u, \mathbf{F})$ *accepts* a word $\mathbf{x} \in \Sigma^*$ if and only if $\bar{u}(\mathcal{S}_0, \mathbf{x}) \in \mathbf{F}$. The set $L(D)$ of all words accepted by an automaton D is called the *language accepted by D* . Two automata, D and D' , are *equivalent*, whenever they accept the same language. An introduction to the theory of automata and formal languages is, for example, given by Carroll and Long [7] or by Hopcroft and Ullman [8].

If two automata

$$D = (\Sigma, \mathbf{S}, \mathcal{S}_0, u, \mathbf{F})$$

and

$$D' = (\Sigma, \mathbf{S}', \mathcal{S}'_0, u', \mathbf{F}')$$

and a function $\mu : \mathbf{S} \rightarrow \mathbf{S}'$ are given, then the function μ is called a *(finite) automaton isomorphism* from D to D' if and only if the following four conditions hold:

1. $\mu(\mathcal{S}_0) = \mathcal{S}'_0$.
2. For all $\mathcal{S} \in \mathbf{S}$: $\mathcal{S} \in \mathbf{F}$ if and only if

$$\mu(\mathcal{S}) = \mathcal{S}' \in \mathbf{F}'.$$

3. For all $\mathcal{S} \in \mathbf{S}$, and for all $v \in \Sigma$:

$$\mu(u(\mathcal{S}, v)) = u'(\mu(\mathcal{S}), v).$$

4. The function μ is a bijection from \mathbf{S} to \mathbf{S}' .

Isomorphic automata are always equivalent, but not vice versa.

2.3 An Automaton for the Assessment of Knowledge

We begin with an example illustrating the idea behind the automaton which will be introduced in this section.

Example 4. For the assessment, we now use the knowledge space from Example 3, which is shown in Fig. 2, and not the and-or-graph from Fig. 1. At the beginning of the assessment, we have no information on the student and, therefore, every knowledge state of the knowledge space is assumed to be a candidate for the student's knowledge state. This is illustrated by Fig. 3a. Now, suppose that again, as in Example 1, item c is tested first and suppose that it is mastered by the student. Now, we have the information that the student's knowledge state must have item c as an element. Therefore, we eliminate the states not containing c in Fig. 3b. Further suppose that item d is tested second and also mastered. Hence, we also eliminate the knowledge states not containing item d . The third item tested is b , which is assumed not to be mastered. This time we eliminate all states containing b , and end up with a single knowledge state $\{a, c, d\}$. This knowledge state can now be assigned to the student and the assessment procedure terminates.

We specify the corresponding automaton such that it accepts the languages described in [9] and models the deterministic version of the assessment procedure in [3]. A detailed explanation of this procedure is given in [10]. An implementation of this automaton is used in the simulation study described in Section 4. This automaton realizes the questioning rule mentioned in the Section 1 as a function p from the set \mathbf{S} of states to the set Q of items extended by the symbol \perp representing the case in which an appropriate question cannot be found.

Definition 1. For a knowledge space \mathcal{K} on a set of questions Q , we define an automaton $A = (\Sigma, \mathbf{S}, \mathcal{S}_0, u, \mathbf{F})$ fulfilling the conditions 1 to 5 below:

1. $\Sigma = \{q, \bar{q} \mid q \in Q\}$.
2. $\mathbf{S} \subseteq 2^{\mathcal{K}}$ such that $\mathcal{S} \in \mathbf{S}$ if and only if,

$$\text{for all } K \in \mathcal{K}, \text{ if } \bigcap \mathcal{S} \subseteq K \subseteq \bigcup \mathcal{S}, \text{ then } K \in \mathcal{S}.$$

3. $\mathcal{S}_0 = \mathcal{K}$.
4. $u : \mathbf{S} \times \Sigma \rightarrow \mathbf{S}$ such that

$$u(\mathcal{S}, v) = \begin{cases} \mathcal{K}_v \cap \mathcal{S}, & v \in \{p(\mathcal{S}), \overline{p(\mathcal{S})}\} \\ \emptyset, & \text{otherwise,} \end{cases}$$

for $\mathcal{S} \in \mathbf{S}$, $v \in \Sigma$, and a function $p : \mathbf{S} \rightarrow Q \cup \{\perp\}$.

5. $\mathbf{F} = \{\{K\} \mid K \in \mathcal{K}\}$.

For the automaton in Definition 1, the alphabet Σ consists of symbols q and \bar{q} indicating whether or not a question $q \in Q$ is mastered by the student. The states $\mathcal{S} \in \mathbf{S}$ of this automaton are subsets of the knowledge space \mathcal{K} . At each step of the assessment process, the state \mathcal{S} of the automaton contains all those members K of the knowledge space \mathcal{K} that are consistent with the sequence of questions mastered, and not mastered up to then. These elements $K \in \mathcal{S}$ are called the

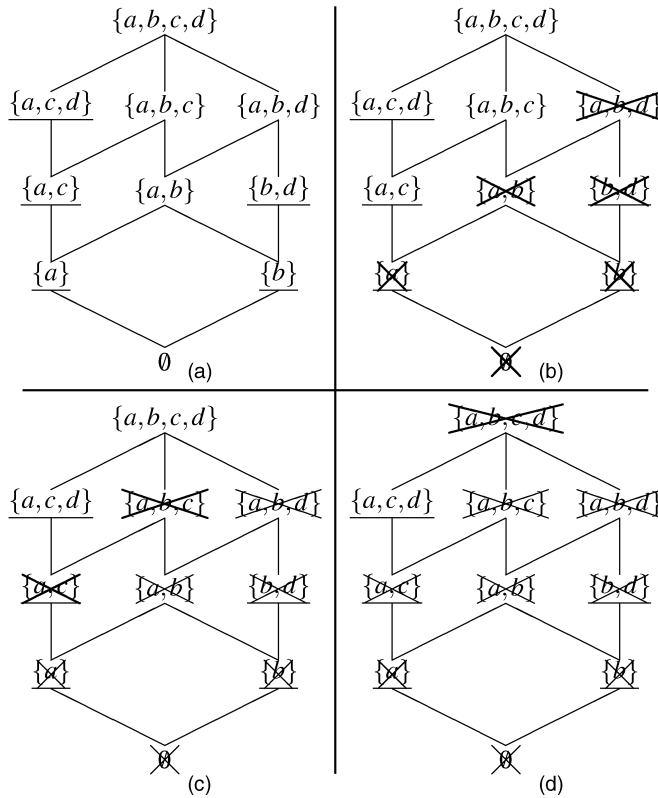


Fig. 3. An example assessment using the knowledge space.

marked knowledge states in [3]. The initial state S_0 of the automaton is the knowledge space \mathcal{K} itself since, at the beginning of the assessment, all of its knowledge states are candidates for the student’s knowledge state. The final states $\mathcal{F} \in \mathbf{F}$ are the sets containing exactly one knowledge state. Once the assessment process has finished, the state of the automaton contains the student’s knowledge state. The function p represents the questioning rule whose existence is assumed in Definition 1. We use the symbol \perp for the case when there is no appropriate new question to be posed. The state transition function u models the updating rule. For a given state \mathcal{S} and for the mastery v of the question $p(\mathcal{S})$ selected by the questioning rule, the new state $u(\mathcal{S}, v)$ is defined as the intersection of the old state \mathcal{S} , and the set \mathcal{K}_v of knowledge states. The new state $u(\mathcal{S}, v)$ fulfills the condition that it is an element of \mathbf{S} .

If the automaton A from Definition 1 fulfills the additional condition that for all $\mathcal{S} \in \mathbf{S}$,

$$p(\mathcal{S}) \in \left(\bigcup \mathcal{S} \setminus \bigcap \mathcal{S} \right) \text{ or } p(\mathcal{S}) = \perp, \quad (C1)$$

then the language $L(A)$ accepted by the automaton A is an *assessment language* for the knowledge space \mathcal{K} in the sense of the definition by Degreeef et al. [9]. In the following, we call an automaton accepting an assessment language for \mathcal{K} *assessment automaton*. Assessment languages represent those assessment procedures that are nonredundant and deterministic. The former property means that only those questions are posed during the assessment that have not been asked before and which

cannot be inferred from previous answers. The latter property ensures that each question selected by the questioning rule is completely determined by the answers to the previous questions. Degreeef et al. [9] use binary decision trees to illustrate the fact that assessment languages represent assessment procedures.

The assessment automata for this knowledge space \mathcal{K} differ by their questioning rule, i.e., the function p . For all of these automata, we have $\Sigma = \{a, \bar{a}, b, \bar{b}, c, \bar{c}, d, \bar{d}\}$. The set of states \mathbf{S} has too many members to list here. For a given function p , however, this set can be simplified by eliminating all states that cannot be reached with this function. The original and the simplified automata are equivalent, i.e., they accept the same language. Procedures for simplifying automata are, for example, described in [7]. Fig. 4 shows the automaton state transition diagram for such a simplified assessment automaton for \mathcal{K} . The initial state S_0 of this automaton is the knowledge space \mathcal{K} . It is marked by a double frame. The final states S_9 to S_{18} are marked by circles. They contain the single knowledge states that are listed to their right. Each of the “intermediary” states S_1 to S_8 consists of the union of all final states that can be reached from this state. For example, the state $S_5 = \{\{b\}, \{b, d\}, \{a, b\}, \{a, b, d\}\}$. The arrows in Fig. 4 indicate the possible transitions between the states of the automaton. An arrow from a state \mathcal{S} to a state \mathcal{S}' is marked by the literal v if $u(\mathcal{S}, v) = \mathcal{S}'$. For every state, the figure shows the transitions for the symbols $p(\mathcal{S})$ and $\bar{p}(\mathcal{S})$ if $p(\mathcal{S}) \neq \perp$. In all other cases, the automaton moves to an additional state, the *garbage state*

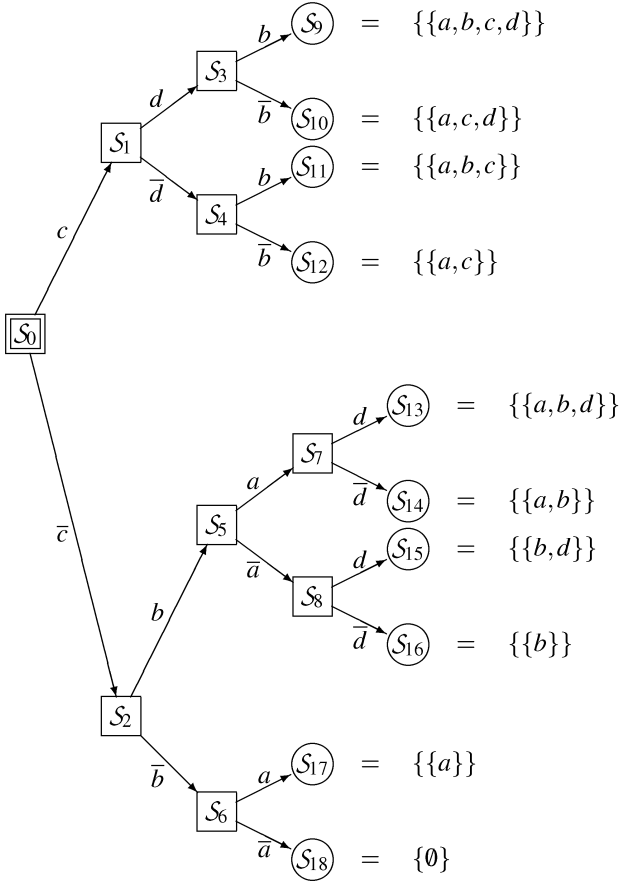


Fig. 4. An example of an assessment automaton.

$S_{19} = \emptyset$ that is not shown in Fig. 4. The automaton remains in this garbage state if it has reached it. The automaton reaches this state whenever it has read a word that is not accepted and cannot be completed by a word accepted by the automaton. In particular, after the automaton has reached a final state, it goes into the garbage state if it reads another symbol.

Fig. 5 shows the binary decision tree illustrating the assessment language accepted by the automaton from Fig. 4. The words in this language correspond to the leaves of the tree. Each of these words describes exactly one knowledge state, as shown in Fig. 5. Note that these knowledge states correspond to those that are elements of the final states of the automaton in Fig. 4, and that the structure of the binary decision tree corresponds to the state transition diagram.

3 A PARSIMONIOUS, EQUIVALENT AUTOMATON FOR THE ASSESSMENT OF KNOWLEDGE

Simulation studies in Section 4 show that programs realizing the automaton described in Section 2, require a lot of computer memory and computing time whenever the number of knowledge states is large. Therefore, we suggest another automaton for the assessment of knowledge. The programs implementing this new automaton will be shown to be far less demanding. We will prove that the two automata are equivalent under conditions that will be specified below.

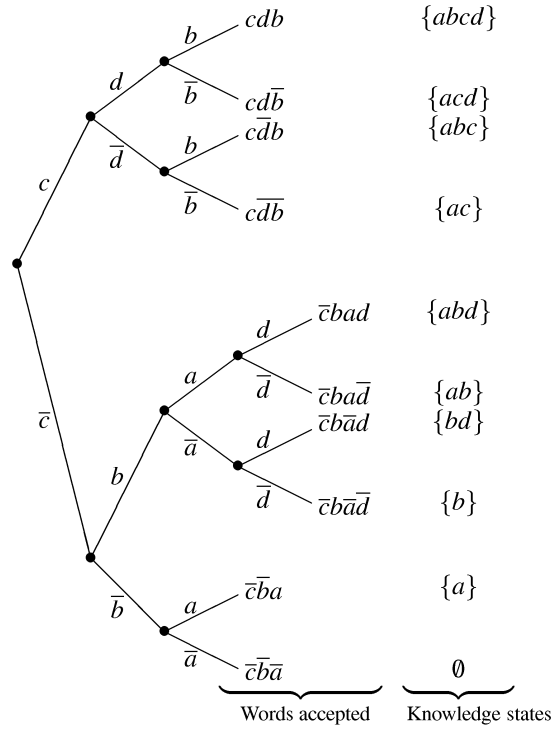


Fig. 5. A binary decision tree for the assessment language.

In this section, we describe our main results. The proofs of the theorems are given in detail in Section 5. In this manner, we first focus on the algorithm and then elaborate on its theoretical background.

We develop a new automaton that is based on the idea of representing the knowledge space only by its basis and not by other states. We introduce a marking for the items $q \in Q$ and another marking for the elements $B \in \mathcal{B}$ of the basis. These markings indicate whether or not an item or a member of the basis is known to be mastered, or known not to be mastered. A member B of the basis \mathcal{B} is called *mastered* if it is fully mastered, i.e., if all of its elements $q \in B$ are mastered. A member $B' \in \mathcal{B}$ is considered not to be mastered if at least one question $q \in B'$ is known not to be mastered. The markings are formalized as the Cartesian product introduced in Definition 2. They are updated each time after the student tested has answered a question.

Definition 2. Let Q be a finite set of questions, let \mathcal{K} be a knowledge space on Q and let \mathcal{B} be its basis. Let \mathbf{X} be the Cartesian product $\mathbf{X} = 2^Q \times 2^Q \times 2^{\mathcal{B}} \times 2^{\mathcal{B}}$. The elements \mathcal{X} of the Cartesian Product \mathbf{X} are written as $\mathcal{X} = (M, N, \mathcal{M}, \mathcal{N})$. Define a relation \subseteq on \mathbf{X} by setting $\mathcal{X} \subseteq \mathcal{X}'$ if and only if the following four conditions hold for $\mathcal{X} = (M, N, \mathcal{M}, \mathcal{N})$ and $\mathcal{X}' = (M', N', \mathcal{M}', \mathcal{N}')$:

1. $M \subseteq M'$,
2. $N \subseteq N'$,
3. $\mathcal{M} \subseteq \mathcal{M}'$, and
4. $\mathcal{N} \subseteq \mathcal{N}'$.

In the definition above, the sets M and N represent the questions that are known at that time to be mastered, and not mastered, respectively. The families \mathcal{M} and \mathcal{N} of

subsets of Q represent the members of the basis \mathcal{B} that are known at that time to be mastered, or not mastered, respectively. The relation \sqsubseteq from Definition 2 is a partial order on \mathbf{X} . For two elements $\mathcal{X}, \mathcal{X}' \in \mathbf{X}$, we have $\mathcal{X} \sqsubseteq \mathcal{X}'$ if \mathcal{X} represents less information on the mastery of the items and members of the basis.

Definition 3. Let $Q, \mathcal{K}, \mathcal{B}$, and \mathbf{X} be as in Definition 2, and let \mathbf{S} be as in Definition 1. For any $K \in \mathcal{K}$, let \mathcal{B}^K be the set of all members of the basis that are a subset of K , $\mathcal{B}^K = \{B \in \mathcal{B} \mid B \subseteq K\}$. Let $\bar{K} = Q \setminus K$ and let $\bar{\mathcal{B}}^K = \mathcal{B} \setminus \mathcal{B}^K$. Define a mapping $h : 2^{\mathcal{K}} \rightarrow \mathbf{X}$ by setting, for $S \subseteq \mathcal{K}$,

$$h(S) = \left(\bigcap S, Q \setminus \bigcup S, \right. \\ \left. \bigcap \{ \mathcal{B}^K \mid K \in S \}, \bigcap \{ \bar{\mathcal{B}}^K \mid K \in S \} \right),$$

and a mapping $d : \mathbf{X} \rightarrow 2^{\mathcal{K}}$ by requiring for

$$\mathcal{X} = (M, N, \mathcal{M}, \mathcal{N}) \in \mathbf{X}$$

that

$$d(\mathcal{X}) = \left\{ K \in \mathcal{K} \mid M \subseteq K, N \cap K = \emptyset \right. \\ \left. \bigcup \mathcal{M} \subseteq K \text{ and } \mathcal{N} \cap \mathcal{B}^K = \emptyset \right\}.$$

In Section 5, it will be shown that the pair (h, d) of mappings constitute a so-called *Galois connection*. The maps h and d from Definition 3 are used to define the new automaton. They assign two types of markings to each other, the function h maps the marking of Falmagne and Doignon [3] to the new marking suggested in this paper and function d is the assignment in the converse direction.

Definition 4. Let $Q, \mathcal{K}, \mathcal{B}$, and \mathbf{X} be defined as in Definition 2, and let $h : 2^{\mathcal{K}} \rightarrow \mathbf{X}$ and $d : \mathbf{X} \rightarrow 2^{\mathcal{K}}$ be the mappings from Definition 3. Define an automaton $\mathbf{B} = (\Sigma, \mathbf{T}, \mathcal{T}_0, w, \mathbf{G})$ fulfilling the conditions 1 to 5 below:

1. $\Sigma = \{q, \bar{q} \mid q \in Q\}$,
2. $\mathbf{T} = \{h(S) \mid S \in 2^{\mathcal{K}}\}$,
3. $\mathcal{T}_0 = \{\emptyset, \emptyset, \emptyset, \emptyset\} \in \mathbf{T}$,
4. $w : \mathbf{T} \times \Sigma \rightarrow \mathbf{T}$ such that

$$w(\mathcal{T}, v) = \begin{cases} (h \circ d)((M \cup \{q\}, N, \mathcal{M}, \mathcal{N})) & v = o(\mathcal{T}) \\ (h \circ d)((M, N \cup \{q\}, \mathcal{M}, \mathcal{N})) & v = \bar{o}(\mathcal{T}) \\ (Q, Q, \mathcal{B}, \mathcal{B}) & \text{otherwise.} \end{cases}$$

for $\mathcal{T} \in \mathbf{T}$, $v \in \Sigma$, and a function $o : \mathbf{T} \rightarrow Q \cup \{\perp\}$.

5. $\mathbf{G} = \{(K, \bar{K}, \mathcal{B}^K, \bar{\mathcal{B}}^K) \mid K \in \mathcal{K}\} \subseteq \mathbf{T}$.

Theorem 1. Let \mathbf{A} and \mathbf{B} be the automata from Definition 1 and from Definition 4, respectively. Let h be the map from Definition 3. The function h is an isomorphism from \mathbf{A} to \mathbf{B} if, for all $S \in \mathbf{S}$, we have $p(S) = o(h(S))$.

Since the automata \mathbf{A} and \mathbf{B} are isomorphic, they are also equivalent, i.e., they accept the same language. With this result, the automaton \mathbf{B} can replace the automaton \mathbf{A} for the assessment of knowledge if we know a way to compute, for some $\mathcal{X} \in \mathbf{X}$, the combined correspondence

$(h \circ d)(\mathcal{X})$ without computing $d(\mathcal{X})$. Theorem 2 below is the foundation for an algorithm computing $(h \circ d)(\mathcal{X})$. This algorithm is described in [11].

Theorem 2. Let \mathbf{X} , d , and h be defined as in Definition 3. Let $\mathcal{X} = (M, N, \mathcal{M}, \mathcal{N}) \in \mathbf{X}$ such that $d(\mathcal{X}) \neq \emptyset$. We obtain $\mathcal{X} = (h \circ d)(\mathcal{X})$ if and only if the three conditions 1 to 3 below are fulfilled for \mathcal{X} :

1. For all $q \in Q$, we have $\mathcal{B}_q \subseteq \mathcal{N}$ if and only if $q \in N$.
2. For all $B \in \mathcal{B}$, we have $B \in \mathcal{M}$ if and only if $B \subseteq M$.
3. For all $q \in M$, we have $(\bigcap \mathcal{B}_q \setminus \mathcal{N}) \subseteq M$.

Condition 1 states that an item is not mastered if none of the basis elements containing this item is mastered and, vice versa. The mastery of an element of a basis was defined in Section 3. Please note, even the fact that none of the basis elements minimal for an item q is mastered, is sufficient to conclude that q itself is not mastered. By condition 2, an element of the basis is mastered if and only if all of its items are mastered. Condition 3 is based on the fact that, for each item mastered, there exists at least one element of the basis which is mastered and contains this item. Therefore, all items are mastered that are in the intersection of all basis elements which might be mastered and contain the first item.

As soon as the assessment automaton \mathbf{B} receives the information that a new item $q \in Q$ is either mastered or not mastered, it computes, for a given $\mathcal{T} \in \mathbf{T} \subseteq \mathbf{X}$ and for the mastery $v \in \{q, \bar{q}\}$ of q , the new state $w(\mathcal{T}, v)$. If $v = q$, then $w(\mathcal{T}, v) = (h \circ d)(M \cup \{q\}, N, \mathcal{M}, \mathcal{N})$ by Definition 4. If $v = \bar{q}$, then $w(\mathcal{T}, v) = (h \circ d)(M, N \cup \{q\}, \mathcal{M}, \mathcal{N})$. The sets M and N , and the families \mathcal{M} and \mathcal{N} are incremented iteratively by items, or members of the basis, respectively, until all three conditions from Theorem 2 are fulfilled. The case $d(\mathcal{X}) = \emptyset$ only occurs if the automaton \mathbf{B} receives an answer for a question that has not been asked, and is, therefore, irrelevant in practice.

After having shown the equivalence of the algorithms, the new algorithm's demand on computing time and memory needs to be compared to the traditional one. The traditional algorithm stores all knowledge states and eliminates states, as described in Example 4. For each update, it has to iterate through the remaining states. Therefore, it follows that both computing time and storage required are a linear function of the product $|\mathcal{K}| \times |Q|$. The new algorithm stores all members of the basis of a knowledge space and, for each item, a list of all basis elements containing this item. As described by Dowling et al. [11], the update procedure propagates information on the student's answers in a constraint satisfaction network. Therefore, both computing time and storage required depend linearly on the product $(\sum_{B \in \mathcal{B}} |B|) \times |Q|$. This estimate is, however, only an upper bound for the computing time.

Example 5. We illustrate the new algorithm using the same knowledge space and the same sequence of answers as in the Examples 1 and 4. Fig. 6a shows a slightly simplified version of the constraint network. The elements of the basis are characterized by triangles containing the \wedge symbol because of the fact that one must master all items

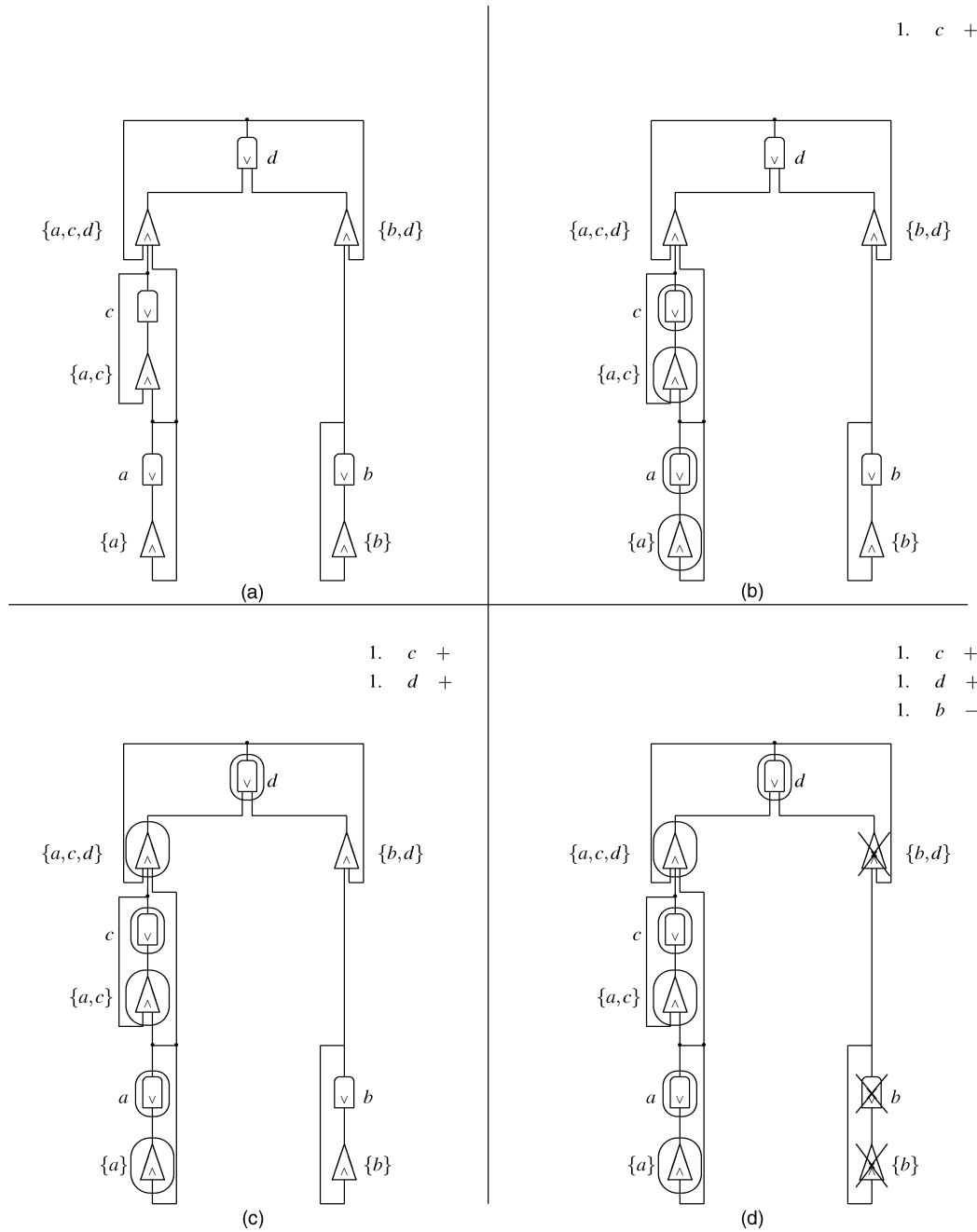


Fig. 6. An example assessment using the basis.

in order to master a basis element. The items are illustrated by the half-ovals containing the \vee symbol because it is sufficient to master one element minimal for an item in order to master the item itself. At the beginning of the assessment, we know nothing about the student. Let us assume that item c is tested first and answered correctly. According to Condition 3, we can conclude that item a is also mastered. Following Condition 2, we can also conclude that the elements $\{a\}$ and $\{a, c\}$ are mastered. This is illustrated by the ovals around the items and basis elements in Fig. 6b. Further suppose that, at the second step, item d is also

answered correctly. From applying Condition 2 again, it follows that the element $\{a, c, d\}$ of the basis is mastered, as illustrated in Fig. 6c. Finally, suppose that item b is not mastered. Using Condition 1, we can now conclude that the basis elements $\{b\}$ and $\{b, d\}$ are not mastered either. This fact is represented by the crosses in Fig. 6d. We now have assigned all items and basis elements to the sets M or N and M or N , respectively. The assessment is therefore finished and we obtain the student's knowledge state as the set $\{a, c, d\}$. This set can be constructed as both the set of the encircled items or as the union of the encircled elements of the basis.

TABLE 1
Average Number of Questions by the Algorithm A and B for
Three Different Knowledge Spaces

Knowledge Space	Size	Optimum	Alg. A	Alg. B
A	121943	16,90	17,12	17,12
B	747283	19.51	19.83	21.07
C	1310950	20.32	20.60	21,83

TABLE 2
Amount of Memory Required by the Algorithms A and B for
Knowledge Spaces and Their Bases

Knowledge Space	Algorithm A		Algorithm B	
	Size	Memory [KB]	Size	Memory [KB]
A	121943	1429.0	55	9.7
B	747283	8757.2	65	9.4
C	1310950	15363.7	60	9.0

4 A SIMULATION STUDY

The purpose of the simulation study is to compare the performance of two programs that have implemented the automata A and B, and are called algorithm A and B, respectively, in the sequel. The selection of the algorithm is our first simulation parameter. Our set Q corresponds to 77 items from the field of fractions that were collected in a project described in [5]. In this project, knowledge spaces on this set Q were derived from the teachers' expert judgments. We used three of these different knowledge spaces, each of which was derived from the judgments of one expert. The knowledge space selected constitutes the second simulation parameter. We simulated the knowledge assessment of 500 students per space by selecting each student's knowledge state from the respective space randomly and by assessing the student's knowledge state with both algorithms separately. During the assessment, questions were chosen according to a questioning rule described below. The answer patterns were derived from the students' knowledge states K , i.e., question q was assumed to be answered correctly if and only if $q \in K$. The number of questions asked, the amount of memory required, and the time needed for the first update were recorded. For algorithm A, the time required for the updating decreases exponentially with the number of questions already posed to the student. The time required for the first update, therefore, is the longest time a student has to wait for a question with this algorithm. For algorithm B, there is no such systematic change to the waiting time during the assessment process.

The questioning rule used by algorithm A is called the *halfsplit rule* [3]. It selects an item q which, in the optimal case, is able to split the set S into two subsets $S \cap \mathcal{K}_q$ and $S \cap \mathcal{K}_{\bar{q}}$ (for the definitions, see Section 2) of the same size. Algorithm B used a questioning rule which selects an item $q \in Q \setminus (M \cup N)$ such that the size of the following difference $\delta(q)$ is minimal. This difference is defined by setting

$$\delta(q) = \frac{\sum_{B \in \mathcal{B} \setminus (N \cup M)} |B \setminus M|}{|\mathcal{B} \setminus (N \cup M)|} - \min_{B \in \mathcal{B}_q \setminus V} |B \setminus M|,$$

where the symbols used are defined as in Sections 2 and 3. We assume that the size $|B \setminus M|$ represents the difficulty of a member $B \in \mathcal{B}$ at the time when the set M is known to be mastered. The fraction above, therefore, describes the mean difficulty of the members of \mathcal{B} from which the minimal difficulty of a member $B \in \mathcal{B}_q$ is subtracted.

The smallest number of questions for sufficient assessment is $\log_2 |\mathcal{K}|$. Table 1 shows the number of questions asked using the algorithms A and B, respectively, on the three different knowledge spaces. In this example, the questioning rule applied in algorithm A is slightly worse than the optimum; the rule applied in algorithm B is slightly worse on the former.

Table 2 shows that, for each of the knowledge spaces used, the memory required by the algorithms A and B. This table also shows the size of the spaces and the size of their bases which are relevant for the algorithm A and B, respectively. These values only depend on the knowledge structures, but not on the students. Table 3 shows the average computation time needed to compute the first update for the same knowledge spaces and their bases.

These data illustrate the theoretically known fact that, for algorithm A, the amount of memory required and the computing time linearly depend on the size of the knowledge space. For algorithm B, such dependencies on the size of the basis are neither observed nor expected. The performance of algorithm B is, however, clearly superior to that of algorithm A for large knowledge spaces.

5 PROOFS

5.1 Proof of Theorem 1

In Theorem 1, we state that there exists an isomorphism between the automata A and B. Here, we show that this isomorphism is induced by a Galois connection.

Proposition 1. *Let \mathcal{Q} , \mathcal{K} , \mathcal{B} , and \mathbf{X} be as in Definition 2, and let $\mathbf{S} \subseteq 2^{\mathcal{K}}$ be the set of states of the automaton A from Definition 1. Let the maps $h: 2^{\mathcal{K}} \rightarrow \mathbf{X}$ and $d: \mathbf{X} \rightarrow 2^{\mathcal{K}}$ be defined as in Definition 3.*

The pair (h, d) is a Galois connection between $(2^{\mathcal{K}}, \subseteq)$ and $(\mathbf{X}, \sqsubseteq)$. The Galois closure operator $(d \circ h): 2^{\mathcal{K}} \rightarrow 2^{\mathcal{K}}$ assigns to each family of sets $\mathcal{L} \subseteq \mathcal{K}$ the smallest family $\mathbf{S} \in \mathbf{S}$ containing \mathcal{L} , and satisfying $(d \circ h)(2^{\mathcal{K}}) = \mathbf{S}$.

Proof. The definition of Galois connections can, e.g., be found in [12] and [13]. The pair (h, d) is a Galois

TABLE 3
Computation Time Needed to Find the First Questions with the
Algorithms A and B for Knowledge Spaces and Their Bases

Knowledge Space	Algorithm A		Algorithm B	
	Size	Time [msec]	Size	Time [msec]
A	121943	18.14	61	0.52
B	747283	104.44	65	0.44
C	1310950	199.48	65	0.54

connection if and only if the following four conditions hold for each $\mathcal{L}, \mathcal{L}' \subseteq \mathcal{K}$, and for all $\mathcal{X}, \mathcal{X}' \in \mathbf{X}$:

1. If $\mathcal{X} \sqsubseteq \mathcal{X}'$, then $d(\mathcal{X}) \supseteq d(\mathcal{X}')$.
2. If $\mathcal{L} \subseteq \mathcal{L}'$, then $h(\mathcal{L}) \supseteq h(\mathcal{L}')$.
3. $\mathcal{X} \sqsubseteq (h \circ d)(\mathcal{X})$.
4. $\mathcal{L} \subseteq (d \circ h)(\mathcal{L})$.

To prove 1, suppose $\mathcal{X} \sqsubseteq \mathcal{X}'$. By Definition 2, we have $M \subseteq M'$, $N \subseteq N'$, $\mathcal{M} \subseteq \mathcal{M}'$, and $\mathcal{N} \subseteq \mathcal{N}'$. Since $M \subseteq M'$ and $\mathcal{M} \subseteq \mathcal{M}'$, we obtain

$$\{K \in \mathcal{K} \mid M \cup \bigcup \mathcal{M} \subseteq K\} \supseteq \{K \in \mathcal{K} \mid M' \cup \bigcup \mathcal{M}' \subseteq K\}.$$

Correspondingly, it follows from $\mathcal{N} \subseteq \mathcal{N}'$ that

$$\{K \in \mathcal{K} \mid \mathcal{N} \subseteq (\mathcal{B} \setminus \mathcal{B}^K)\} \supseteq \{K \in \mathcal{K} \mid \mathcal{N}' \subseteq (\mathcal{B} \setminus \mathcal{B}^K)\},$$

and from $N \subseteq N'$ that

$$\{K \in \mathcal{K} \mid K \cap N' = \emptyset\} \supseteq \{K \in \mathcal{K} \mid K \cap N = \emptyset\}.$$

By definition of the map d , we have $d(\mathcal{X}) \supseteq d(\mathcal{X}')$.

To prove 2, suppose $\mathcal{L} \subseteq \mathcal{L}'$. It follows that $\bigcap \{\mathcal{B}^K \mid K \in \mathcal{L}\} \supseteq \bigcap \{\mathcal{B}^K \mid K \in \mathcal{L}'\}$, and that

$$(\mathcal{B} \setminus \bigcup \{\mathcal{B}^K \mid K \in \mathcal{L}\}) \supseteq (\mathcal{B} \setminus \bigcup \{\mathcal{B}^K \mid K \in \mathcal{L}'\}).$$

We also have $\bigcap \mathcal{L} \supseteq \bigcap \mathcal{L}'$, and $(Q \setminus \bigcup \mathcal{L}) \supseteq (Q \setminus \bigcup \mathcal{L}')$. By the definition of the map h , we obtain $h(\mathcal{L}) \supseteq h(\mathcal{L}')$.

To prove 3, let $\mathcal{X} = (M, N, \mathcal{M}, \mathcal{N})$, and let

$$(M', N', \mathcal{M}', \mathcal{N}') = (h \circ d)(\mathcal{X}).$$

We prove $M \subseteq M'$. For any $q \in M$, we obtain $d(\mathcal{X}) \subseteq \mathcal{K}_q$. Since $M' = \bigcap d(\mathcal{X})$ by definition of h , we have $q \in M'$, and, hence, the result. We prove $\mathcal{M} \subseteq \mathcal{M}'$. For any $B \in \mathcal{M}$ and for any $K \in d(\mathcal{X})$, we have $B \subseteq K$ by definition of d and $B \in \mathcal{B}^K$. Using the definition of h , we get $B \in \mathcal{M}' = \bigcap \{\mathcal{B}^K\}$, and, hence, $\mathcal{M} \subseteq \mathcal{M}'$. Analogously, we obtain $N \subseteq N'$ and $\mathcal{N} \subseteq \mathcal{N}'$.

To prove 4, let \mathcal{L} be a subset of \mathcal{K} , let $\mathcal{X} = h(\mathcal{L})$, and let $\mathcal{L}' = d(\mathcal{X}) = (d \circ h)(\mathcal{L})$. For any $K \in \mathcal{L}$, we obtain $M \subseteq K$, $N \cap K = \emptyset$, $\mathcal{M} \subseteq \mathcal{B}^K$, and $\mathcal{N} \cap \mathcal{B}^K = \emptyset$. Therefore, $K \in \mathcal{L}'$ and, consequently, $\mathcal{L} \subseteq \mathcal{L}'$.

It remains to be proven that the set

$$\{(d \circ h)(\mathcal{L}) \mid \mathcal{L} \subseteq \mathcal{K}\} = \mathbf{S}.$$

For any $\mathcal{S} \in \mathbf{S}$, we prove that $\mathcal{S} = (d \circ h)(\mathcal{S})$. Let $\mathcal{X} = (M, N, \mathcal{M}, \mathcal{N}) \in \mathbf{X}$ be defined by setting $M = \bigcap \mathcal{S}$, $N = Q \setminus \bigcup \mathcal{S}$,

$$\mathcal{M} = \{B \in \mathcal{B} \mid B \subseteq M\},$$

and

$$\mathcal{N} = \{B \in \mathcal{B} \mid B \not\subseteq \bigcup \mathcal{S}\}.$$

By definition of d , we get $d(\mathcal{X}) = \mathcal{S}$ and, hence, $\mathbf{S} \subseteq \{(d \circ h)(\mathcal{L}) \mid \mathcal{L} \subseteq \mathcal{K}\}$. Conversely, we prove that, for any closed $\mathcal{L} = (d \circ h)(\mathcal{L}) \in 2^{\mathcal{K}}$, we have $\mathcal{L} \in \mathbf{S}$. Let $(M, N, \mathcal{M}, \mathcal{N}) = h(\mathcal{L})$. By definition of the mapping h , we obtain that $\bigcup \mathcal{M} \subseteq M$. It also follows, for any $K \in \mathcal{K}$, that if K and N are disjoint subsets of Q , the families \mathcal{B}^K and \mathcal{N} are also disjoint. Therefore, we obtain

$$(d \circ h)(\mathcal{L}) = \{K \in \mathcal{K} \mid M \subseteq K \text{ and } N \cap K = \emptyset\}.$$

By definition of h ,

$$(d \circ h)(\mathcal{L}) = \{K \in \mathcal{K} \mid \bigcap \mathcal{L} \subseteq K \text{ and } \bigcap \{\overline{K'} \mid K' \in \mathcal{L}\} \cap K = \emptyset\}.$$

Since $\bigcap \{\overline{K'} \mid K' \in \mathcal{L}\} = \overline{\bigcup \mathcal{L}}$, we obtain $(d \circ h)(\mathcal{L}) = \{K \in \mathcal{K} \mid \bigcap \mathcal{L} \subseteq K \text{ and } K \subseteq \bigcup \mathcal{L}\} \in \mathbf{S}$ by definition of \mathbf{S} . \square

Corollary 1. *The functions d and h induce a one-to-one correspondence between the closed families of sets $(d \circ h)(\mathcal{L}) \in 2^{\mathcal{K}}$ and the closed quadruples $(h \circ d)(\mathcal{X}) \in \mathbf{X}$.*

Remark. This result follows immediately from the fact that the pair (d, h) is a Galois connection [13].

Since the functions d and h are bijections between \mathbf{S} and \mathbf{T} , it remains to be shown that the Conditions 1, 2, and 3 specified in the definition of automaton isomorphisms in Section 2 hold for the automata $\mathbf{A} = (\Sigma, \mathbf{S}, \mathcal{S}_0, u, \mathbf{F})$ and $\mathbf{B} = (\Sigma, \mathbf{T}, \mathcal{T}_0, w, \mathbf{G})$ whenever $p(\mathcal{S}) = o(h(\mathcal{S}))$ for any $\mathcal{S} \in \mathbf{S}$.

To prove 1, we have to show that $h(\mathcal{K}) = (\emptyset, \emptyset, \emptyset, \emptyset)$. This follows from the definition of h and the fact that $\emptyset, Q \in \mathcal{K}$.

Since for all $K \in \mathcal{K}$, $h(\{K\}) = (K, \overline{K}, \mathcal{B}^K, \overline{\mathcal{B}^K})$, we have proven 2 using the definition of h .

To prove 3, we have to show that, for all $\mathcal{S} \in \mathbf{S}$, and for all $\nu \in \Sigma$, $h(u(\mathcal{S}, \nu)) = w(h(\mathcal{S}), \nu)$. Suppose that $p(\mathcal{S}) = o(h(\mathcal{S}))$, and let $\mathcal{T} = (M, N, \mathcal{M}, \mathcal{N}) = h(\mathcal{S})$. According to Condition 4 in Definition 4, there are three cases for assigning the values $w(\mathcal{T}, \nu)$. First, suppose that $\nu = p(\mathcal{S})$, i.e., $\nu = q$ for some $q \in Q$. Hence,

$$w(\mathcal{T}, \nu) = (h \circ d)((M \cup \{q\}, N, \mathcal{M}, \mathcal{N})).$$

From the definition of d , we obtain

$$\begin{aligned} w(\mathcal{T}, \nu) &= h(\{K \in \mathcal{K} \mid (M \cup \{q\}) \cup \bigcup \mathcal{M} \\ &\subseteq K \text{ and } (N \cup \bigcap \mathcal{N}) \cap K = \emptyset\}). \end{aligned}$$

Using the definition of M , N , \mathcal{M} , and \mathcal{N} , we have $w(\mathcal{T}, \nu) = h(\{K \in \mathcal{S} \mid q \in K\}) = h(u(\mathcal{S}, \nu))$.

The proof of the second case $\nu = \overline{p(\mathcal{S})}$ is analogous. In the third case in that $\nu \neq p(\mathcal{S})$ and $\nu \neq \overline{p(\mathcal{S})}$, it follows that $u(\mathcal{S}, \nu) = \emptyset$ by Definition 1, and that

$$w(\mathcal{T}, \nu) = (Q, Q, \mathcal{B}, \mathcal{B}) = h(\emptyset),$$

and, hence, that $u(\mathcal{S}, \nu) = w(\mathcal{T}, \nu)$.

5.2 Proof of Theorem 2

We first specify two properties of those $\mathcal{X} \in \mathbf{X}$ for which the condition $d(\mathcal{X}) \neq \emptyset$ holds.

Proposition 2. *Let \mathbf{X} and d be defined as in Definition 3, and let $\mathcal{X} = (M, N, \mathcal{M}, \mathcal{N}) \in \mathbf{X}$ such that $d(\mathcal{X}) \neq \emptyset$. Then, \mathcal{X} fulfills the following two conditions:*

1. $M \cup (\bigcup \mathcal{M}) \cap N = \emptyset$ and
2. $\mathcal{M} \cap \mathcal{N} = \emptyset$.

Both conditions follow immediately from the definition of d .

Proposition 3. Let \mathcal{K} be a knowledge space on a set Q of items, and let \mathcal{B} be the basis of \mathcal{K} . For an item $q \in Q$ and for a knowledge state $K \in \mathcal{K}$, it follows that

$$q \in K \text{ if and only if } \mathcal{B}^K \cap \mathcal{B}_q \neq \emptyset.$$

This result follows from the definition of the basis in Section 2.1.

To prove Theorem 2, we have to prove that $\mathcal{X} = (h \circ d)(\mathcal{X})$ if and only if \mathcal{X} fulfills the Conditions 1 to 3 specified there.

Suppose $\mathcal{X} = (h \circ d)(\mathcal{X})$ to prove that the Conditions 1 to 3 hold for \mathcal{X} .

For $q \in Q$, suppose further $\mathcal{B}_q \subseteq \mathcal{N}$ to prove 1. By definition of d , we have, for all $B \in \mathcal{B}_q$,

$$B \in \bigcap \{ \overline{\mathcal{B}^K} \mid K \in d(\mathcal{X}) \} = \mathcal{N}.$$

The definition of $\overline{\mathcal{B}^K}$ yields $B \not\subseteq K$ for all $K \in d(\mathcal{X})$ and all $B \in \mathcal{B}_q$. From Proposition 3, it follows that $q \notin K$ for all $K \in d(\mathcal{X})$, and $q \notin \bigcup d(\mathcal{X})$. Using the definition of h , we obtain $q \in \mathcal{N}$. Conversely, suppose $q \in \mathcal{N}$. We obtain $q \notin \bigcup d(\mathcal{X})$ by definition of d . By Proposition 3, $\mathcal{B}_q \cap \mathcal{B}^K = \emptyset$ for all $K \in d(\mathcal{X})$ and, therefore, also $\mathcal{B}_q \subseteq \overline{\mathcal{B}^K}$. Hence, $\mathcal{B}_q \subseteq \bigcap \{ \overline{\mathcal{B}^K} \mid K \in d(\mathcal{X}) \} = \mathcal{N}$.

To prove 2, suppose $B \in \mathcal{M}$. Since $\mathcal{X} = (h \circ d)(\mathcal{X})$ and $\mathcal{M} = \bigcap \{ \mathcal{B}^K \mid K \in d(\mathcal{X}) \}$ by definition of h , we have $B \in \mathcal{B}^K$, and $B \subseteq K$ for all $K \in d(\mathcal{X})$. Using $M = \bigcap d(\mathcal{X})$ from the definition of h , we obtain that $B \subseteq M$ if and only if $B \in \mathcal{M}$.

To prove 3, suppose $q \in M$ and $q' \in \bigcap (\mathcal{B}_q \setminus \mathcal{N})$. By definition of h , we obtain $q \in K$ for all $K \in d(\mathcal{X})$. Hence, there exists, for all $K \in d(\mathcal{X})$, an element $B_q \in (\mathcal{B}_q \setminus \mathcal{N})$ with $B_q \subseteq K$. From $q' \in \bigcap (\mathcal{B}_q \setminus \mathcal{N})$, it follows that $q' \in B_q \subseteq K$. Hence, we obtain $q' \in K$ for all $K \in d(\mathcal{X})$ and, by definition of h , $q' \in M$.

To prove the converse, namely, that it follows from the Conditions 1 to 3 that $\mathcal{X} = (h \circ d)(\mathcal{X})$, we use a contradiction. Suppose that all of the Conditions 1, 2, and 3 are fulfilled but that $\mathcal{X} \neq (h \circ d)(\mathcal{X})$. Let $\mathcal{L} = d(\mathcal{X})$ and let $\mathcal{X}' = (M', N', \mathcal{M}', \mathcal{N}') = h(\mathcal{L})$. From $\mathcal{X} \neq \mathcal{X}'$ and from $\mathcal{X} \subseteq \mathcal{X}'$, we conclude $\mathcal{X} \subsetneq \mathcal{X}'$. As a consequence, we have to change the set inclusion " \subseteq " into a strict set inclusion " \subsetneq " in at least one of the four inequalities $M \subseteq M'$, $N \subseteq N'$, $\mathcal{M} \subseteq \mathcal{M}'$, and $\mathcal{N} \subseteq \mathcal{N}'$. For each of these four cases, we show that it contradicts the Conditions 1 to 3 from Theorem 2.

Suppose that $M \subsetneq M'$. Let q be an item in the set difference $M' \setminus M$. By Condition 2, we obtain $\mathcal{B}_q \cap \mathcal{M} = \emptyset$, and, therefore, $q \notin (\bigcup \mathcal{M})$. From Condition 3 and Proposition 2, it follows, for each $q' \in M$, that there exists a basis element $B_{q'} \in \mathcal{B}_{q'} \setminus \mathcal{N}$ such that $q \notin B_{q'}$ and $B_{q'} \cap N = \emptyset$. Since $q \notin B_{q'}$, the knowledge state $K = (\bigcup_{q' \in M} B_{q'}) \cup (\bigcup \mathcal{M})$ is an element of \mathcal{L} that does not contain item q . This contradicts the assumption $q \in M'$, and, hence, the assumption $M \neq M'$.

Suppose that $N \subsetneq N'$, i.e., there exists an item $q \in N' \setminus N$. From 1, we can conclude that there exists a $B \in \mathcal{B}_q \setminus \mathcal{N}$ such that $B \cap N = \emptyset$. Since $d(\mathcal{X})$ is nonempty, we can, for an arbitrary $K \in \mathcal{L}$, construct a knowledge state $K' = K \cup B$. This knowledge state K' is also an element of \mathcal{L} , and, hence, we obtain $N = N'$.

Suppose that $\mathcal{M} \subsetneq \mathcal{M}'$. Using Condition 2, we can conclude, that also $M \subsetneq M'$ that leads to a contradiction as in the case $M \subsetneq M'$.

Suppose that $\mathcal{N} \subsetneq \mathcal{N}'$, i.e., there exists a basis element $B \in \mathcal{N}' \setminus \mathcal{N}$. From 1, we conclude that $B \cap N = \emptyset$. For any $K \in \mathcal{L}$, the knowledge state $K' = K \cup B$ also is an element of \mathcal{L} , and, therefore, $B \notin \mathcal{N}$ and $\mathcal{N} = \mathcal{N}'$.

6 CONCLUSIONS

The results of Section 3 were applied to implement the new algorithm for the assessment of knowledge. This implementation is written in C++ and is integrated in a system for the adaptive assessment and training [11]. The core of the assessment algorithm are the three Conditions 1, 2, and 3 from Theorem 2. Once a question is answered by a student, the assessment program updates the sets M , N , \mathcal{M} , and \mathcal{N} of mastered items, not mastered items, mastered basis elements, and not mastered basis elements, respectively.

The advantage of this new algorithm is that it enables an efficient adaptive assessment. The disadvantage is that it does not yet take into account the possibility that students may make careless errors or lucky guesses. Therefore, an extension of this algorithm that takes care of this problem needs to be developed. Such developments have already been accomplished for the automaton A in Section 2.3 [6], [3], [10].

ACKNOWLEDGMENTS

The authors would like to thank Jean-Paul Doignon for his helpful comments on an earlier draft of this manuscript. The work described in this paper was done while C. Hockemeyer was a student of computer science at Braunschweig.

REFERENCES

- [1] J.-P. Doignon and J.-C. Falmagne, "Spaces for the Assessment of Knowledge," *Int'l J. Man-Machine Studies*, vol. 23, pp. 175–196, 1985.
- [2] J.-P. Doignon and J.-C. Falmagne, *Knowledge Spaces*. Berlin: Springer-Verlag, 1999.
- [3] J.-C. Falmagne and J.-P. Doignon, "A Markovian Procedure for Assessing the State of a System," *J. Math. Psychology*, vol. 32, pp. 232–258, 1988.
- [4] M. Schrepp, "A Generalization of Knowledge Space Theory to Problems with More than Two Answer Alternatives," *J. Math. Psychology*, vol. 41, pp. 237–243, 1997.
- [5] K. Baumunk and C.E. Dowling, "Validity of Spaces for Assessing Knowledge about Fractions," *J. Math. Psychology*, vol. 41, pp. 99–105, 1997.
- [6] J.-C. Falmagne and J.-P. Doignon, "A Class of Stochastic Procedures for the Assessment of Knowledge," *British J. Math. and Statistical Psychology*, vol. 41, pp. 1–23, 1988.
- [7] J. Carroll and D. Long, *Theory of Finite Automata*. Englewood Cliffs, N.J.: Prentice Hall, 1989.
- [8] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*. Reading, Mass.: Addison-Wesley, 1979.
- [9] E. Degreef, J.-P. Doignon, A. Ducamp, and J.-C. Falmagne, "Languages for the Assessment of Knowledge," *J. Math. Psychology*, vol. 30, pp. 243–256, 1986.
- [10] J.-P. Doignon, "Probabilistic Assessment of Knowledge," *Knowledge Structures*, D. Albert, ed., pp. 1–56, 1994.

- [11] C.E. Dowling, C. Hockemeyer, and A.H. Ludwig, "Adaptive Assessment and Training Using the Neighbourhood of Knowledge States," *Intelligent Tutoring Systems*, C. Frasson, G. Gauthier, and A. Lesgold, eds., pp. 578–586, 1996.
- [12] O. Ore, *Theory of Graphs*. Providence, R.I.: AMS Colloquium Publications, Am. Math. Soc., 1962.
- [13] G. Birkhoff, *Lattice Theory*. Am. Math. Soc. Colloquium Publication, Am. Math. Soc., third ed., 1967.



Cornelia E. Dowling studied psychology at the University of Regensburg, Germany, where she received her doctorate degree in 1983. From 1984 to 1986, she was supported in her postdoctoral research at New York University by a Feodor-Lynen fellowship of the Alexander von Humboldt foundation. It was during this period that she became interested in knowledge assessment and adaptive computer-based learning. In the years between 1986 and 1994,

she was a lecturer at the University of Technology at Braunschweig, Germany, where she received the degree of "Habilitation." Since 1994, she has been a professor at the Fachhochschule für öffentliche Verwaltung (College for Public Administration) in Hamburg, Germany. She remains affiliated with Braunschweig University, where most of her research projects are located.



Cord Hockemeyer studied computer science and psychology (as minor) at the University of Technology at Braunschweig, Germany, where he received his Diplom in 1993. From 1994 to 1996, he was supported by a postgraduate fellowship from the German state of Lower Saxony. Since 1996, he has been working as a postgraduate researcher at the Department of Psychology, University of Graz, Austria. From 1996 to 1997 and again from 1998 to 1999, he received the Marie-Curie-Fellowships of the European Commission. He is a member of the IEEE and the IEEE Computer Society.

▷ For further information on this or any computing topic, please visit our Digital Library at <http://computer.org/publications/dlib>.