

Ordering and Combining Distributed Learning Objects through Skill Maps and Asset Structures

Dietrich Albert

Cognitive Science Section
Graz University, Austria
dietrich.albert@uni-graz.at

Luca Stefanutti

Cognitive Science Section
Graz University, Austria
luca.stefanutti@uni-graz.at

ABSTRACT

Comparing (and thus ordering) learning objects, as well as producing new objects by combining together existing ones are two basic activities in a distributed learning system which exploits object reusability as the fundamental criterion for the construction of adaptive learning systems. In this paper we present a competency-based formal approach for comparing and combining learning objects in a distributed system.

1 Introduction

A competency-based approach to learning shifts the focus from a “course model” to a “competency model” based on the dynamic personalisation and adaptation of the content to the skills, competencies and knowledge of the learner. One requirement of this approach is that learning objects are tagged with appropriate information (e.g., in the form of object metadata) on the skills that a learner needs to master the content provided by the learning object itself. This makes a learning object somehow “autonomous” and “independent” of the particular learning system in which it is embedded, allowing for reusability of content in a distributed competency-based learning environment.

Comparing (and thus ordering) learning objects, as well as producing new objects by combining together existing ones are two basic activities in a distributed learning system which exploits object reusability as the fundamental criterion for the construction of adaptive learning systems. In this paper we present a competency-based formal approach for comparing and combining learning objects in a distributed system. At the basis of our approach there is the concept of a *surmise relation* between learning objects. Essentially, two learning objects a and b are in a surmise relation if the fact that a learner masters object b can be surmised from the fact that (s)he masters object a .

The concept of a surmise relation comes from knowledge space theory (Doignon & Falmagne, 1985; Albert & Lukas, 1999; Doignon & Falmagne, 1999), a rigorous and efficient formal framework for the construction, validation, and application of e-assessment and e-learning adaptive systems. This theory is mainly a psychometric theory for the assessment of knowledge; however, as adaptivity and optimal learning paths does matter, its methods and models can be fruitfully exploited for e-learning purposes as well.

According to knowledge space theory, a *domain of knowledge* is a collection Q of items (e.g., learning objects, problems, questions, exercises, examples, etc.) in a given field of knowledge (e.g., mathematics, physics, chemistry, biology, etc.). Then, the *knowledge state* of a student is the set K of all items in Q that this student actually masters, and a *knowledge structure* for Q is a pair (Q, \mathcal{K}) in which \mathcal{K} is the collection of all knowledge states that can be observed in a certain population of students. If \mathcal{K} is closed under union (i.e., $K \cup K' \in \mathcal{K}$ whenever $K, K' \in \mathcal{K}$) then it is called a *knowledge space*. Sometimes also closure under intersection holds ($K \cap K' \in \mathcal{K}$ whenever $K, K' \in \mathcal{K}$), in which case, \mathcal{K} is called a *quasi-ordinal knowledge space*. Quasi ordinal knowledge spaces are in a 1-1 correspondence with surmise relations (i.e., they are just two alternative representations of the same concept).

Another important concept of knowledge space theory is that of a *skill map*, which is the subject of the next section. A skill map provides the basis for mapping the performance of a learner (represented by her knowledge state) into the competence level of the skills that the learner needs for mastering the content of the items contained in her knowledge state. Basically, given a set of skills, and a set of learning objects, a skill map *assigns* a subset of skills to every single learning object. Skill assignment can be done by different methods. Some of them are cognitive task analysis (Korossy, 1999), expert query (Koppen & Doignon, 1990), and systematical construction (Albert & Held, 1999).

A learning object is often a compound object, as it consists of different “components” (e.g., a text component, a picture component, a video, etc.) working together as a single unit. These components are called “assets” and are used, e.g., by authors for constructing new learning objects by means of special authoring systems. A learning object can thus be represented (described) as a set of assets along with relationships among them. If it is assumed that skills can be assigned directly to the assets (as we do here) then learning objects should, in some sense “inherit” the skills from the assets they contain. How to represent learning objects in terms of the assets they contain, as well as how to assign skills to learning objects, based on the skills assigned to the assets is the topic of Section 3. In this section we introduce special representations of learning objects, called *asset structures*. Each learning object is represented by means of an asset structure that, from a mathematical point of view, is a labeled directed graph, whose nodes are the assets of the learning object, and whose edges specify relationships among the assets. In Section 4 we will show how learning objects can be compared by using their asset structures, and Section 5 introduces an algebra on asset structures, which allows to combine the structures of two or more learning objects when they are merged into more complex objects.

2 Skill maps and surmise relations among assets

In this section we face the problem of how to assign skills to the different assets. A skill map (Korossy, 1993; Doignon, 1994; Düntsch & Gediga, 1995; Korossy, 1999) is a triplet $\langle X, A, \tau \rangle$ in which A is a nonempty set of items (*assets*, in the sequel), X is a nonempty set of skills, and $\tau : A \rightarrow 2^X$ is a mapping such that $\tau(a) \neq \emptyset$ for all $a \in A$. The interpretation of τ is the following: for any a in A , $\tau(a)$ is the set of all skills that are sufficient to master asset a .

A knowledge structure on the assets can be easily derived from a skill map. By the conjunctive model of Doignon (Doignon, 1994; Doignon & Falmagne, 1999), a subset K of A is said to be *delineated* by a subset Y of X if and only if

$$K = \varphi(Y) = \{a \in A : \tau(a) \subseteq Y\}. \quad (1)$$

In words, every asset a in K is mastered by (some subset of) skills in Y , and K contains all such items. Doignon and Falmagne show that the collection of all subsets $K \subseteq A$ fulfilling (1) is a knowledge structure close both under union and under intersection - i.e., a quasi-ordinal knowledge space.

A surmise relation ‘ \wedge ’ among assets is then obtained in the following way: for any two assets $a, b \in A$,

$$a \wedge b \Leftrightarrow \tau(a) \subseteq \tau(b),$$

that is, we say that mastery of asset b can be *surmised* from the mastery of asset a whenever the set of skills assigned to a is a subset of that assigned to b . The surmise relation ‘ \wedge ’ is reflexive and transitive and, thus, a quasi order with the equivalence classes $[a]$ collecting all assets having the same set of skills assigned to them. When $a \wedge b$ holds for some assets a and b in A , we say that asset a is a *prerequisite* for asset b or, equivalently, that b is *as much demanding* (in terms of skills required) as a .

3 Asset structures

A single learning object often consists of a number of different assets (text, video, pictures) grouped together in a meaningful way. In this section we present an approach for ordering and combining existing (potentially distributed) learning objects which is based on a particular kind of mathematical objects called *asset structures*. First we present the essential concepts of the mathematical foundation of asset structures,

and then we show, with a few examples, how they can be used for obtaining surmise relations among learning objects, as well as for obtaining new learning objects from the existing ones. The mathematics of asset structures is strongly based on a formalism, mostly applied in computational linguistics, called the *feature structure formalism* (Carpenter, 1992; King, 1994). An asset structure can be viewed as a special kind of feature structure in which the assets are partially ordered according to a given skill map (see Section 2). Informally, every single asset in an asset structure is characterized by a certain number of features (like, for instance, the type of asset, the topic, the language, and so on). Thus, we shall start by considering a set of features and a set of assets.

Let $F = \{f_1, f_2, \dots, f_m\}$ be a set of features, and $\langle A, \leq \rangle$ a partially ordered set of assets, with $A = \{a_1, a_2, \dots, a_n\}$. Let both A and F be fixed throughout and such that $A \cap F = \emptyset$. An asset structure is a 4-tuple $S = \langle Q, \bar{q}, \alpha, \gamma \rangle$, where Q is a set of nodes, $\bar{q} \in Q$ is called the root node of the structure, $\alpha : Q \rightarrow A$ is a partial function assigning assets to some of the nodes, and $\gamma : Q \times F \rightarrow Q$ is a partial function specifying the edges of the structure. An asset structure can be represented by means of a labeled directed graph (digraph), in which nodes are labeled by assets, and edges are labeled by features.

To give an example, let $\{\text{picture, topic, sub-topic, text, language}\}$ be the set of features, and $\{\text{PICTURE1, DESCRIPTION, ENGLISH, MATH, MATRIX_INVERSION}\}$ be the collection of assets. Suppose, then, that a simple learning object is described by the asset structure $S_1 = \langle Q_1, \bar{q}_1, \alpha_1, \gamma_1 \rangle$, where $Q_1 = \{0, 1, 2, 3, 4, 5\}$ is the set of nodes, $\bar{q}_1 = 0$ is the root node, and α_1 and γ_1 are defined as follows: $\alpha_1(0)$ is not defined, $\alpha_1(1) = \text{PICTURE1}$, $\alpha_1(2) = \text{DESCRIPTION}$, $\alpha_1(3) = \text{MATH}$, $\alpha_1(4) = \text{ENGLISH}$, and $\alpha_1(5) = \text{MATRIX_INVERSION}$; $\gamma_1(0, \text{picture}) = 1$, $\gamma_1(0, \text{text}) = 2$, $\gamma_1(0, \text{topic}) = 3$, $\gamma_1(1, \text{topic}) = 3$, $\gamma_1(2, \text{topic}) = 3$, $\gamma_1(2, \text{language}) = 4$, and $\gamma_1(3, \text{sub-topic}) = 5$. The digraph representing this asset structure is displayed in Figure 1. The structure S_1 describes a very simple learning object containing a picture along with some text explanation. Both text and picture have “linear algebra” as topic.

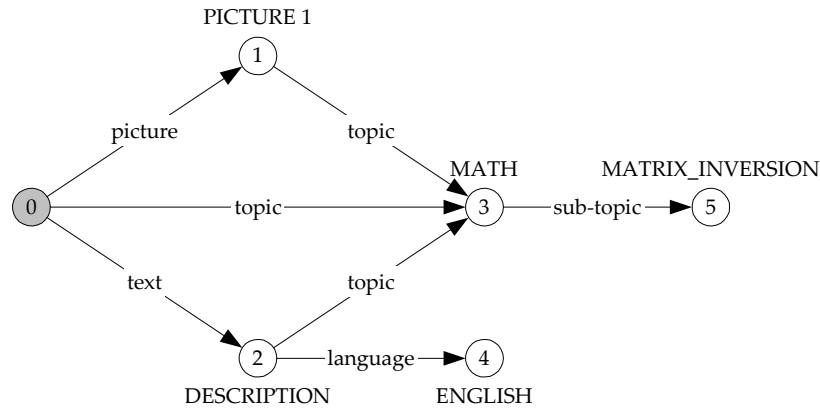


FIGURE 1. Example of an asset structure for a simple learning object displaying a picture along with some text explanation on a topic in linear algebra.

The root node of the structure is node 0 and, as it can be seen, each node can be reached from this node following some path in the graph. The root node is the entry node in the asset structure of the learning object, and the edges departing from this node specify the main features of the learning object itself. Thus, in our example, the learning object represented by S_1 is characterized by three different features: picture, text and topic. The values of these three features are given by $\alpha_1(\gamma_1(0, \text{picture})) = \text{PICTURE1}$, $\alpha_1(\gamma_1(0, \text{text})) = \text{DESCRIPTION}$, and $\alpha_1(\gamma_1(0, \text{topic})) = \text{MATH}$.

Observe, for instance that node 5 can be reached from node 0 by following the path $\langle \text{picture, topic, sub-topic} \rangle$. The fact that, in this example, each node is reachable from the root node through some path is not a coincidence, as we explicitly require that every node in an asset structure be reachable from the root node. This requirement can be stated formally. To do so we need to introduce the formal definition of a path, and an extension of the function γ to paths. Let $S = \langle Q, \bar{q}, \alpha, \gamma \rangle$ be an asset structure. First, we consider sequences

$\sigma = \langle f_1, f_2, \dots, f_k \rangle$ of features, with $\varepsilon = \langle \rangle$, the empty sequence of features, fixed throughout. The collection of all such sequences is

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

Then, an extension $\Gamma: Q \times F^*$ of γ to sequences of features in F is defined (recursively) as follows: for all $q \in Q$, $f \in F$, and $\sigma \in F^*$,

$$\Gamma(q, \varepsilon) = q$$

$$\Gamma(q, f\sigma) = \Gamma(\gamma(q, f), \sigma).$$

A *path* in S is a sequence $\sigma = \langle f_1, f_2, \dots, f_k \rangle$ of features in F such that $\Gamma(q, \sigma)$ is defined for some node $q \in Q$. We denote with Π_S the collection of all paths in the asset structure S . Then, our requirement for all asset structures is that, for each node $q \neq \bar{q}$ in Q , there is some path $\pi \in \Pi_S$ such that $\Gamma(\bar{q}, \pi) = q$. This just formalizes the idea that each node in the structure is reachable from the root node.

For $\pi \in \Pi_S$, the sub-structure having $\Gamma(\bar{q}, \pi)$ as root node is called the *path value* of π in S . Thus, for instance, going back to our example structure S_1 , the path value of the sequence $\langle \text{text}, \text{topic} \rangle$ of features is displayed in Figure 2.

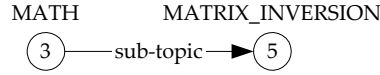


Figure 2. A digraph representing the value of the path $\langle \text{text}, \text{topic} \rangle$ in the asset structure S_1 .

Some interesting properties of asset structures are *reentrancy*, and *cyclicity*. An asset structure S is said to be *reentrant* if there are, at least, two paths $\omega, \pi \in \Pi_S$ such that $\Gamma(\bar{q}, \omega) = \Gamma(\bar{q}, \pi)$. The structure displayed in Figure 1 is reentrant, as the three paths $\langle \text{picture}, \text{topic} \rangle$, $\langle \text{text}, \text{topic} \rangle$, and $\langle \text{topic} \rangle$ have the same value, which is the structure represented in Figure 2. A structure S is said to be *cyclic* if there is some path $\omega \in \Pi_S$ and some node $q \in Q$ such that $\Gamma(q, \omega) = q$.

4 Asset structure subsumption

Different asset structures provide different kinds of information about learning objects and assets. Moreover, the assets may differ to each other in the sets of skill assigned to them. In Section 2 we have shown how assets are comparable in terms of the skills assigned to them. Given a set X of skills, and a set A of assets, a skill map $\langle X, A, \tau \rangle$ induces a partial order on A . In this section we get into the problem of comparing different asset structures taking into account both the partial order on the assets, and the degree of information stored in the different asset structures. With an informal statement, we say that an asset structure S *subsumes* another asset structure T if T contains at least as much information as S , and the mastery of the assets in T can be surmised from the mastery of the assets in S . In other words, T represents a learning object that is at least as specific and as demanding as that represented by S .

We define thus a binary relation \vee (called *subsumption*) on the asset structures by requiring that, given two structures $S_1 = \langle Q_1, \bar{q}_1, \alpha_1, \gamma_1 \rangle$ and $S_2 = \langle Q_2, \bar{q}_2, \alpha_2, \gamma_2 \rangle$, $S_1 \vee S_2$ if and only if there exists a mapping $h: Q_1 \rightarrow Q_2$ such that the following three conditions hold:

- (1) $h(\bar{q}_1) = \bar{q}_2$,
- (2) $h(\gamma_1(q, f)) \wedge \gamma_2(h(q), f)$ for all $q \in Q_1$ and all $f \in F$ such that $\gamma_1(q, f)$ is defined,
- (3) $\alpha_1(q) \wedge \alpha_2(h(q))$ for all $q \in Q_1$ such that $\alpha_1(q)$ is defined.

The function h maps nodes in Q_1 into nodes in Q_2 . Condition (1) requires that the root node of the first structure is mapped into the root node of the second. Then, if a feature f connects two nodes $p, q \in Q_1$ then, by condition (2), we must have the corresponding nodes $h(p)$ and $h(q)$ connected by the same feature f . Moreover, the asset $\alpha_1(q)$ assigned to a node $q \in Q_1$ must be a prerequisite for the asset $\alpha_2(h(q))$ of the corresponding node in S_2 .

We go back again to our example of Figure 1. We add to the set of assets a new asset called MATRIX_PRODUCT and we suppose that MATRIX_PRODUCT \wedge MATRIX_INVERSION (i.e., mastery of the first asset is a prerequisite for the mastery of the second asset). Then, consider the asset structure depicted in Figure 3.

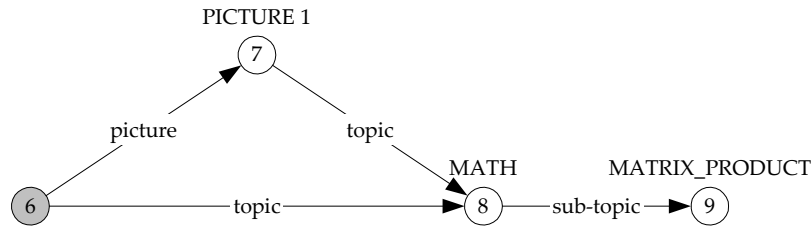


FIGURE 3. An asset structure for a learning object containing a picture explaining the concept of matrix product in linear algebra.

The structure in Figure 3 subsumes that in Figure 1, in fact there exists a mapping fulfilling conditions (1) to (3) for these two structures. Using the set theoretical notation for functional relations, the mapping is $h = \{(6,0), (7,1), (8,3), (9,5)\}$. If compared with the structure in Figure 1, that in Figure 3 contains less information (as, for instance, there is no text) and less demanding assets (in our example MATRIX_PRODUCT is supposed to be a prerequisite for MATRIX_INVERSION).

The subsumption relation is both reflexive and transitive and, thus, a quasi-order on the asset structures. If for two asset structures S_1 and S_2 both $S_1 \vee S_2$ and $S_2 \vee S_1$ hold then S_1 and S_2 are isomorphic asset structures ($S_1 \equiv S_2$); in fact it is easy to check that in this case the mapping h is a bijection. Two learning objects having isomorphic asset structures are said to be equivalent.

It should be noted that asset structure subsumption combines together two distinct orders: an order on the information stored in the structures (in terms of the complexity of the learning object), and an order on the assets themselves (in terms of skills required to master the assets). Two asset structures are comparable with respect to subsumption only if they are comparable with respect to both these two orders. An example of this fact is given in Figure 4.

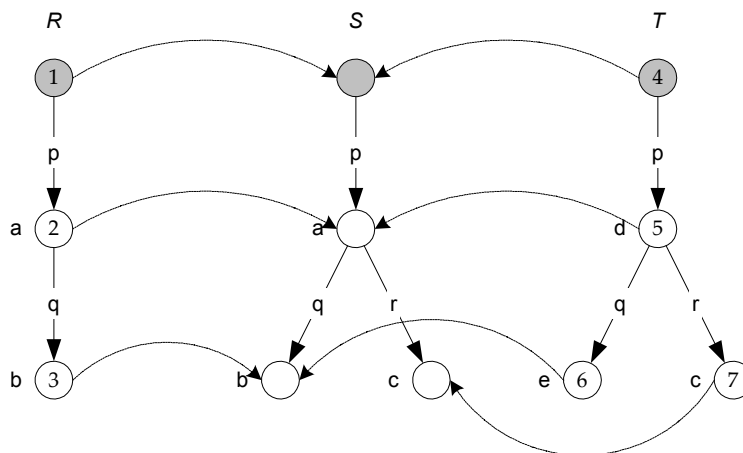


FIGURE 4. Asset structures R and S are comparable w.r.t. subsumption as well as structures T and S . However, structures R and T are not.

Let R , S , and T be three asset structures on the set $\{p, q, r\}$ of features, and on the set $\{a, b, c, d, e, f\}$ of assets. Structure R subsumes structure S because it contains less information than S , even if the assets in the comparable nodes are the same (a , b , and c). If we assume that $d \wedge a$ and $e \wedge b$, then structure T subsumes structure S because the assets in T are less demanding than those in S , even if these two structures are equivalent in terms of information stored. On the other hand, structures R and T are incomparable.

5 Asset structure algebra

Asset structures can be combined together to obtain more complex (both more specific and more demanding) structures, or simpler (either less specific or less demanding) structures by means of two binary operators on asset structures: a *unification* operator, and a *generalization* operator. In order to introduce these two operations, we need to define corresponding binary operations on the assets in A . The *join* of two assets A is a binary operator $_$ on A such that, given three assets $a, b, c \in A$,

$$c = a _ b \Leftrightarrow \tau(c) = \tau(a) \cup \tau(b).$$

The dual of the join operator is the *meet* of two assets, and it is defined as follows:

$$c = a \wedge b \Leftrightarrow \tau(c) = \tau(a) \cap \tau(b).$$

It is easy to see that the set A of assets is bounded complete, that is, any subset $B \subseteq A$ having an upper bound has, in fact, a least upper bound; dually, if B has a lower bound then it has also a greatest lower bound. This means that the two operations $_$ and \wedge are well-defined, and $\langle A, _, \wedge \rangle$ is an algebra on the assets.

The unification of two asset structures consists of constructing a new asset structure which combines the information stored in the two original structures. To this aim we need to define an equivalence relation \approx on the nodes of two structures. Given two structures $S_1 = \langle Q_1, \bar{q}_1, \alpha_1, \gamma_1 \rangle$ and $S_2 = \langle Q_2, \bar{q}_2, \alpha_2, \gamma_2 \rangle$, ' \approx ' is the least equivalence relation on $Q_1 \cup Q_2$ such that:

1. $\bar{q}_1 \approx \bar{q}_2$, and
2. $q_1 \approx q_2, \gamma_{12}(q_1, f) \downarrow, \gamma_{12}(q_2, f) \downarrow$ imply $\gamma_{12}(q_1, f) \approx \gamma_{12}(q_2, f)$ for all $q_1, q_2 \in Q_1 \cup Q_2$, and all $f \in F$,

where: $\gamma_{12} = \gamma_1 \cup \gamma_2$, and for any partial function ϕ , the notation $\phi(x) \downarrow$ indicates that ϕ is defined for x . For any node q , we denote with $[q]$ the equivalence class of q with respect to ' \approx '. Moreover, the equivalence relation ' \approx ' is said to be *join-preserving* if and only if $_{-}\{\alpha_{12}(r) : r \in [q]\}$ exists for every node $q \in Q_1 \cup Q_2$ such that $\alpha_{12}(q)$ is defined, where $\alpha_{12} = \alpha_1 \cup \alpha_2$. Dually, it is *meet-preserving* if and only if $\wedge\{\alpha_{12}(r) : r \in [q]\}$ exists for every node $q \in Q_1 \cup Q_2$ such that $\alpha_{12}(q)$ is defined.

The unification $S_1 \uparrow S_2$ of two asset structures is defined only if ' \approx ' is join-preserving. In which case it yields an asset structure $S_3 = \langle Q_3, \bar{q}_3, \gamma_3, \alpha_3 \rangle$ such that:

1. $Q_3 = \{[q] : q \in Q_1 \cup Q_2\}$;
2. $\bar{q}_3 = [\bar{q}_1]$;
3. $\gamma_3([q], f) = \begin{cases} [r] & \text{if there is } q' \in [q] \text{ such that } \gamma_{12}(q', f) = r, \\ \text{undefined} & \text{otherwise} \end{cases}$;
4. $\alpha_3([q]) = \begin{cases} _\{\alpha_{12}(r) : r \in [q]\} & \text{if } \alpha_{12}(q) \text{ is defined,} \\ \text{undefined} & \text{otherwise.} \end{cases}$

It can be shown that the unification of two asset structures S_1 and S_2 , when exists, is the least upper bound of $\{S_1, S_2\}$ with respect to the subsumption relation, that is the following two relations hold:

$$S_1 \vee S_1 \text{ t } S_2 \text{ and } S_2 \vee S_1 \text{ t } S_2 .$$

An example of unification is given by the three asset structures R , S , and T in Figure 4. Structure S is in fact the unification of R and T . The equivalence classes are: $[1] = \{1,4\}$, $[2] = \{2,5\}$, $[3] = \{3,6\}$, and $[7] = \{7\}$. Then, we have $\gamma_S([1], p) = [2]$, $\gamma_S([2], q) = [3]$, $\gamma_S([2], r) = [7]$, and $\alpha_S([1]) \uparrow$, $\alpha_S([2]) = a _ d = a$, $\alpha_S([3]) = b _ e = b$, and $\alpha_S([7]) = c$.

The generalization of two asset structures (denoted by $S_1 \cup S_2$) is the dual of unification. The generalization of two structures is defined only if ' \approx ' is meet-preserving. For the generalization operator we have:

$$S_1 \cup S_2 \vee S_1 \text{ and } S_1 \cup S_2 \vee S_2 ,$$

that is, the generalization of two asset structures is their greatest lower bound with respect to subsumption. Using again the structures in Figure 4 as an example, the generalization of structures R and T is the asset structure displayed in Figure 5.

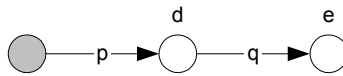


FIGURE 5. Digraph representing the generalization of the asset structures R and T displayed in Figure 4.

In the remaining part of this section we show how unification can be used to produce new learning objects by merging existing ones. We assume that when two learning objects are merged, their respective asset structures are unified. The asset structure of the merged object is the unification of the structures of the original objects. As usual, A is our partially ordered set of assets. From now on we assume the existence of an (abstract) asset $\perp \in A$ and that $\tau(\perp) = \emptyset$. Of course, $\perp \wedge a$ for all $a \in A$. This new asset has only a theoretical value, and can be meant as the *empty* asset. Furthermore, if $S = \langle Q, \bar{q}, \alpha, \gamma \rangle$ is an asset structure, then $\alpha(q)$ is defined for all nodes $q \in Q$ with, eventually, some of the nodes labeled by \perp .

Suppose that two learning objects, that we call O_1 and O_2 are characterized, by the structures S_1 and S_2 depicted in Figure 6.

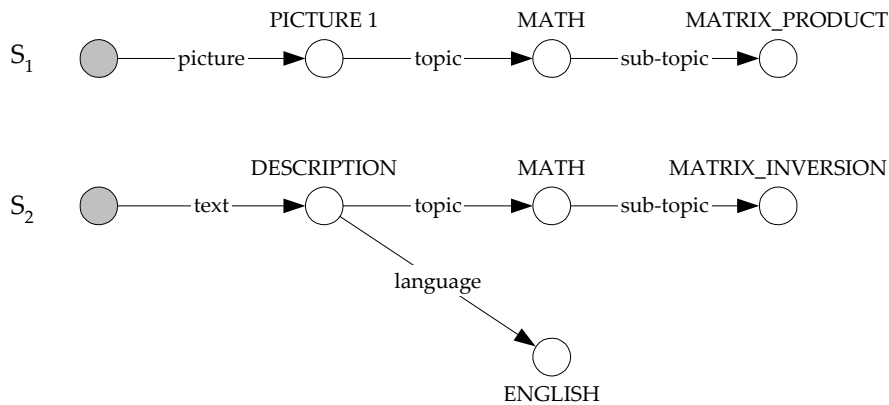


FIGURE 6. Asset structures of learning objects O_1 and O_2 (see text). Some of the nodes are not explicitly labeled. For these nodes the label \perp is intended.

If these two structures are unified, the resulting structure is that displayed in figure 7.

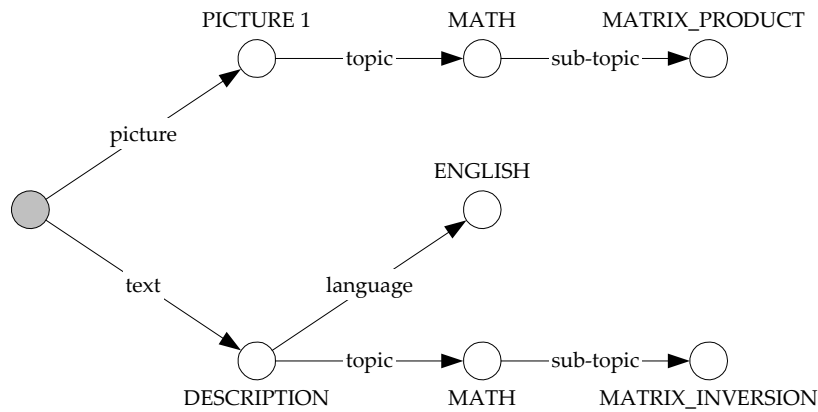


FIGURE 7. The result of the unification of two asset structures.

From this figure it should be noticed that the unified structure represents a learning object in which the picture and the text do not have the same topic: in fact, the topic of the picture is “matrix product”, and that of the text is “matrix inversion”. This shows that, without any further constraint, asset structure unification does not assure to obtain a result which is consistent from a semantic point of view. In order to assure a result that is consistent in this sense, one way is to define a “template” asset structure which puts constraints on the unification operation. The constraint in the example above is that the text and the picture should share the same topic. This constraint is introduced in an appropriate way through the template structure displayed in Figure 8.

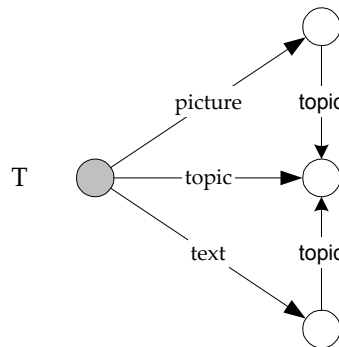


FIGURE 8. A template asset structure for a learning object containing a picture and a text on the same topic.

Template structure T specifies that the learning object has three features: a \langle picture \rangle , a \langle topic \rangle , and a \langle text \rangle , and that picture and text must share the same topic (which is just the topic of the learning object itself). If now we compute the unification among S_1 , S_2 , and T we just obtain the asset structure depicted in Figure 1. In this structure, as expected, the text and the picture correctly share the same topic. Finally, note that the sub-topic of this unified structure is MATRIX_INVERSION, that is, the more demanding asset between MATRIX_INVERSION and MATRIX_PRODUCT.

References

- Albert, D. and Held, T. (1999). Component based knowledge spaces in problem solving and inductive reasoning. In Dietrich Albert and Josef Lukas, editors, *Knowledge Spaces: Theories, Empirical Research, Applications*, pp. 15-40. Lawrence Erlbaum Associates, Mahwah, NJ.
- Albert, D. and Lukas, J. (1999). *Knowledge Spaces: Theories, Empirical Research, Applications*. Lawrence Erlbaum Associates, Mahwah, NJ.
- Carpenter, B. (1992). The logic of typed feature structures. *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge.

Doignon, J.-P. and Falmagne, J.-C. (1985). Spaces for the assessment of knowledge. *International Journal of Man-Machine Studies*, 23,175-196.

Doignon, J.-P. and Falmagne, J.-C. (1999). *Knowledge Spaces*. Springer-Verlag, Berlin.

Doignon, J.-P. (1994). Knowledge spaces and skill assignments. In Gerhard H. Fischer and Donald Laming, editors, *Contributions to Mathematical Psychology, Psychometrics, and Methodology*, pp. 111-121. Springer-Verlag, New York.

Düntsch, I. and Gediga, G. (1995). Skills and knowledge structures. *British Journal of Mathematical and Statistical Psychology*, 48, 9-27.

King, P. J. (1994). Typed feature structures as descriptions. In *Proceedings of COLING*.

Koppen, M. and Doignon, J.-P. (1990). How to build a knowledge space by querying an expert. *Journal of Mathematical Psychology*, 34, 311-331.

Korossy, K. (1993). Modellierung von Wissen als Kompetenz und Performanz [Modeling knowledge as competence and performance]. *Inauguraldissertation, Fakultät für Sozial- und Verhaltenswissenschaften, Universität Heidelberg, Heidelberg, Germany*.

Korossy, K. (1999). Modeling knowledge as competence and performance. In Dietrich Albert and Josef Lukas, editors, *Knowledge Spaces: Theories, Empirical Research, Applications*, pp. 103-132. Lawrence Erlbaum Associates, Mahwah, NJ.