# A Learning-Oriented Knowledge Representation for Teaching Interfaces *

**Paulo J. Garrido**
Industrial Electronics Department
University of Minho
Portugal
pgarrido@dei.uminho.pt

**Abstract** – *This paper presents a formalism for representing knowledge, intended for use in teaching interfaces. The driving concern in developing the formalism was to get an explicit method for sequencing the presentation of materials to be learned. The first criterion for doing the sequencing is that it satisfies a relation of constructability or learning support among the materials. If the relation is restricted to be a function then it is possible to represent it by a directed acyclic graph, as it happens in other constructive processes. The directed acyclic graph representation enables automatic generation of presentation sequences satisfying the learning support relation. It turns out that the formalism suggests other criteria in a natural way. These criteria may be added in a variety of ways to refine the presentation and enhance the responsiveness of the interface to learner needs.*

**Keywords:** Teaching interfaces, computer-aided instruction, e-learning, knowledge representation, constructive processes.

## 1 Introduction

By a teaching interface one understands a specialized man-machine interaction system aimed to assist, facilitate and drive an user to the learning of some piece(s) of knowledge or the acquiring of some competence(s). Computer-aided instruction is an application that requires teaching interfaces, but it is not the only one. The growing complexity of computer applications requires that, in general, a teaching interface be present, implicitly in the form of a help system, or explicitly in the form of a tutorial.

To the best knowledge of the author, design of teaching interfaces relies on intuitive, non-formalized procedures for *sequencing* the presentation of materials or learning goals. This paper describes a formal method to structure teaching interfaces based on a building map from what is to be learnt next to what must be known now. A graph representation of this function suggests flexible and principled ways to do such sequencing. The formal method makes usual assumptions underlying non-formalized procedures explicit and is flexible to accommodate changes in the assumptions. It seems amenable to straightforward software implementation.

The statement underlying this formal approach is that learning is a constructive process. Knowledge is built upon knowledge. This idea is formalized in Section 2 by introducing a constructability relation as a relation from a set to its proper subsets. Functional constructability relations have simpler properties and one uses them to define building maps. Building maps establish for each knowledge element one minimal necessary set of other elements to be in possession of the learner before the learning of the knowledge element is attempted.

Exploring the representation of a building map by a directed acyclic graph (DAG) is made in Section 3. It is observed that the possibilities of such representation go beyond furnishing solutions for sequencing matters in a first presentation. Section 4 covers some design and software implementation aspects. Section 5 concludes resuming the results and commenting on perspectives.

## 2 Constructability relations

Let a knowledge domain $K$ be described as a finite set of knowledge elements:

$$K = \{k_1, k_2, \ldots, k_N\}. \quad (1)$$

One considers the situation of a learner who intends to learn $K$, having no prior knowledge of any element of $K$. It is clear that along the process, to learn an arbitrary element $k \in K$, he or she must have undergone a learning or construction process for all the elements of a given (eventually empty) subset of $K$. A set as such will be called a *support set* for $k$. In general, given a domain $K$, more than one support set may exist for an element $k$.

Formally, one defines a *constructability* or *learning support relation* $C$ over $K$ as a binary relation among the elements of $K$ and proper subsets of $K$:

$$\forall k, kCs \rightarrow s \subset K \ . \tag{2}$$

With the meaning that in the context of the $K$ domain the learner can attempt construction or learning of knowledge element $k$, if she has undergone a learning or construction process for all the elements of the subset $s$ of $K$.

It is clear that knowledge of $k$ itself cannot be a pre-requisite for learning $k$. It makes no sense that $k$ belongs to any of its support sets. As it makes no sense that to construct a thing, that thing should appear constructed in first place. This means that $C$ must satisfy the implication:

$$\forall k, kCs \rightarrow k \notin s \ . \tag{3}$$

$C$ can be seen as a relational mapping that assigns to each $k$ one or several proper subsets of $K$, say, $s_{1k}$, $s_{2k}$, …, $s_{Nk}$. Learning of $k$ can only happen after any of the $s_{1k}$, $s_{2k}$, …, $s_{Nk}$ have been learnt.

Besides (3) $C$ must satisfy another consistency condition: if knowledge element $x$ belongs to a given support set of $k$ then at least one support set of $x$ must be included in the given support set of $k$. Let $i$ and $j$ be index variables over the support sets of $k$ and $x$, respectively. Then $C$ must satisfy the implication:

$$\forall s_{ik}, x \in s_{ik} \rightarrow \exists s_{jx}, s_{jx} \subset s_{ik} \ . \tag{4}$$

The elements of $K$ mapped by $C$ to the empty set alone will be called *initials* in $K$ under $C$. These elements of $K$ are assumed learnable without prior knowledge of other elements of $K$. The elements of $K$ that do not belong to any subset in the range of $C$ will be called *terminals* in $K$ under $C$. If the learner acquires all the terminals in $C$ then it may be said that he has learnt (all of) $K$.

## 2.1 Functional constructability relations

In general construction processes $C$ must be taken as a relational map. However, in the domain of interest, restricting $C$ to be a functional map simplifies the analysis and the application of the formalism and does not seem to introduce any severe shortcomings. In this case, $C$ becomes a function that assigns to each $k$ a subset $s(k)$ of $K$, the unique support set of $k$. Under $C$, previous learning of the elements of $s(k)$ is a necessary condition to attempt the learning of $k$.

Taking $C$ as functional map simplifies in particular the analysis of the induced order in $K$. For a functional relation $C(k) = s(k)$ the conditions (2) and (3) can be written more compactly as:

$$\forall k \in K, \ s(k) \subset K \wedge k \notin s(k) \ . \tag{5}$$

Moreover, condition (4) becomes:

$$\forall k \in K \, \forall x \in s(k), \ s(x) \subset s(k) \ . \tag{6}$$

It may be seen that a function $C(k) = s(k)$ satisfying (5) and (6) induces a strict partial order in $K$, given by:

$$x \prec k \leftrightarrow x \in s(k) \ . \tag{7}$$

Two elements of $K$ will be incomparable if neither belongs to the support set of the other. The initials and terminals in $K$ under $C$ will be respectively the minimal and maximal elements of this order.

## 2.2 Building maps

A building map can indicate the order induced in $K$ by a functional $C$ concisely. This means that it is not necessary to indicate explicitly all the elements of $s(k)$ for each $k$, to have a complete specification of $C$.

Let one consider the subset $b(k)$ of $s(k)$ defined by:

$$b(k) = \{x \mid \neg \exists y, x \prec y \prec k\} \ . \tag{7}$$

The subset $b(k)$ has the property of resuming all of $s(k)$ for the learning or construction of $k$. As soon as all of its elements have been learnt or constructed, the user can proceed to $k$. It will be called the *building set* for $k$. Let one now define a learning or *building map* $B$ as one that assigns to each $k$ in $K$ a building set:

$$\forall k \in K, B : k \mapsto b(k) \ . \tag{8}$$

It may be seen that the process of taking the union of the building sets that result from iterating $B$ on the elements of building sets it generates, departing from some $k$, must give $s(k)$. With some ease of expression, $s(k)$ may be said to be the transitive closure of $B$ on $k$.

# 3 The DAG representation

A building map $B$ defined on a knowledge domain $K$ may be represented by a directed acyclic graph (DAG). Each node of the DAG is to represent one element of $K$ and an arrow exists from node $x$ to node $k$ if $x \in b(k)$. Figure 1 gives a simple example. One can observe that $k_1$ and $k_2$ are initials, $k_5$ is terminal. The building and support sets are as follows:

$$b(k_1) = s(k_1) = \varnothing$$
$$b(k_2) = s(k_2) = \varnothing$$
$$b(k_3) = s(k_3) = \{k_1\}$$
$$b(k_4) = s(k_4) = \{k_1, k_2\}$$
$$b(k_5) = \{k_2, k_3, k_4\}; s(k_5) = \{k_1, k_2, k_3, k_4\}$$

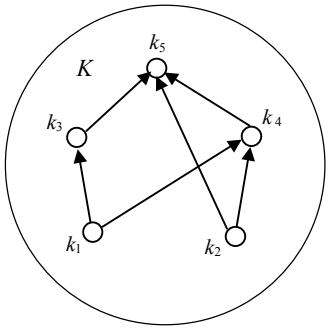


Figure 1 DAG representation of a hypothetical 5-element *K* domain and associated *B* map

To put the formal machinery working, let one suppose that a teaching interface over some knowledge domain *K* is to be designed. One must specify:

i) The elements of the knowledge domain *K*;

ii) An associated building map *B* for *K*.

This may be done in tabular form through a standard editor or it may be done through a dedicated graphical editor. The resulting data may be represented as a graph. The specification of *B* is to be checked for consistency regarding condition (5). This is equivalent to the graph having no cycles or being effectively a DAG.

Two advantages become apparent in having a building map represented by a DAG:

– It allows for easy visualization of the *B* map and of the underlying constructability or learning support function *C* for *K*.

– It allows for generating automatically sequences adequate to lead a user through the learning of *K*.

Each of these items will be reviewed in turn.

### 3.1 Visualization of *B* and *C*

This seems a powerful tool both for learner and for designer. The learner may assess visually the magnitude and steps of the learning task and of his or her progress in learning. The designer may get insight in the structure of the knowledge domain and best learning paths.

To get the maximal benefit of this, the number of elements of *K* is under two constraints. It must allow for a readable image of the DAG under the visualization technology employed. Moreover, it must not be above medium human perceptual ability to reasonably cope with the number of different items present in an image. This gives an indication how should a given general domain of knowledge be split in *K* modules if the need arises.

### 3.2 Sequencing the presentation of *K*

Let one note that there is a one-to-one correspondence between a building map and the DAG representing it. One can refer the set of nodes of the DAG representing the support set of *k* as the support nodes of the node *k*. A *topological sort* of a DAG [3] is a sequencing of its nodes such that no node comes first than the nodes in its support set. The following are two topological sorts of the DAG in Figure 1:

$$S_1 =< k_1, k_2, k_4, k_3, k_5 >$$
$$S_2 =< k_2, k_4, k_1, k_3, k_5 > .$$

If a topological sort of a DAG is used to sequence the presentation of the elements of *K*, then it is guaranteed that for all *k*, all the elements of *s*(*k*) are presented before *k*. This basic property being granted, the adequateness of a topological sort to present *K* is to be evaluated by the designer, given the particular context of knowledge and learning. A seemingly general criterion is that the time interval between the learning of some node and its application in the learning of another should be minimized.

Topological sorts of a DAG can be automatically generated. It seems easy to augment the generation algorithms with criteria to evaluate the properties of the sorts, as the minimal time interval referred.

## 4 Design and software

The formalization of a knowledge domain proposed here has two connected aspects to be considered:

i) The choice of the number of elements of *K* and their granularity;

ii) The associated building map *B* for *K*.

The number of elements of *K* depends on the extension of the knowledge domain and the granularity chosen for the elements. As a rule, a learning element should neither overload the user capabilities for one step learning nor bore her because of its simplicity.

If, as pointed above, one wants to get maximal benefits from the learner visualizing the knowledge

domain representation then the number of elements should not be so big that the learner feels lost or feels facing a hard learning task. In addition, performance and limitations of the visualization technology are of impact.

Besides specifying *K,* the designer must specify *B*. These two specifications may proceed entangled. It is of interest to consider the possibility that the need arises to revise the specifications. This means that (through interaction with users or monitoring their performance) one has detected a better specification for *K* and *B* resulting in a more learnable presentation or a more efficient teaching interface. Clearly, the formalism proposed eases this refinement because learning considerations underlie the knowledge representation.

The software needed to begin to use the formalism described here consists mainly in a (graphical) editor for the specifications of *K* and *B*. Associated algorithms for checking that a graph is a DAG, for visualizing it and for generating topological sorts are readily available, as in [1,2,4].

The output of the editor to the teaching interface appears in first place to be the topological sort(s) chosen for presentation and a description of the DAG. The first will be used to sequence the presentation. The second may be used (together with the graph visualization software) to give the learner a vision of the task and progress made.

# 5 Conclusions and perspectives

A method for designing systematically the presentation sequence of matters in a teaching interface has been presented. The method stands on representing a knowledge domain as a set of elements *K* with an associated building map *B*. The building map specifies for each element *k* of *K* a minimal necessary subset of *K*, the elements of which should be known in order that the learning of *k* may be immediately attempted.

A building map can be represented by a directed acyclic graph (DAG). Topological sorts of the DAG are reasonable presentations of *K* because they never present an element before those necessary to learn it. The designer may choose from the possible sorts that or those considered most suited to do the job of leading the learner through learning. The generator of the topological sorts may embed criteria to assist the designer in the choice. The topological sort chosen may be used by the teaching interface to do the sequence presentation.

Besides making explicit a rationale for sequencing matters and suggesting enlarging it, the method also suggests to use the visualization of the DAG to enhance the perception of both designer and learner of the overall learning task. For the last one, visualization may be used also to give feedback on progress.

For designers, the specification of a building map appears as the price to pay for using the method. This is a price but it is also an invitation to develop a more structured vision of the knowledge domain to be presented by the teaching interface and to measure the merits of the structuring.

The software module needed to begin using the method appears to be a building map and associated DAG representation editor, including processing functions on DAGs. It is also clear that the teaching interface should be made capable of using the editor's outputs and functions.

## 5.1 Perspectives

The use of this formalism presents two main perspectives of interest to refer here: making the interface adaptive and knowledge formalization.

If feedback from the user regarding his prior knowledge or his progress is made available, one can think of making the interface adapt to the level of knowledge and competencies demonstrated by the user and to her needs, in several ways.

The DAG representation gives also a systematic approach to gather feedback from the learner in order to establish his prior knowledge regarding the domain at stake. Having this feedback available allows for remaking the topological sort used for presentation in order that elements already known are not presented. In fact, this could be done for every user.

If one gathers feedback from the user at each learning step, then user failures may be used to initiate appropriate revising procedures or even to present matters in other ways. It should be noted that this last possibility makes appeal to non-functional building maps.

Also, if for some applications there is evidence that users become frustrated from attempts to use facilities they are not prepared to manipulate, then the described method could be in-built in the application interface, so that this would only open access to facilities upon having reasonable evidence of user proficiency.

Knowledge can be represented in many ways. The representation formalism proposed in this paper comes from concerns on making the task of learning a knowledge domain through a teaching interface as efficient as possible. It is not clear that knowledge representations geared to facilitating learning are the most concise or appear as structures people are acquainted with.

With regard to this, it is important to observe that deduction is also a constructive process. One may establish a correspondence between a set of formulae deductively built and a knowledge domain equipped with a relation of constructability or learning support. A formula in the deductive set corresponds to a knowledge element; an axiom corresponds to an initial; the set of formulae necessary to deduce a given formula corresponds to a support set. If all these sets are unique, then a correspondence between the formulae used in the last inference of a deduction and a building set holds.

This is not to mean that learning is or should be taken mainly as a deductive activity. What one would like to stress is that the similarity between axiomatic representations of knowledge and the formalism presented here may be explored. On one hand, an axiomatic representation of a knowledge domain may furnish a good start point to make a specification of a set of knowledge elements and a building map (or a learning support relation) for it. On the other hand, such a specification may constitute an important step to get an axiomatic description.

As a final comment, let one point that research in relational maps beyond the functional concept presented here, seems of interest both for the study of constructive processes in general and for the development of man-machine interfaces.

# References


[1] G. di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the visualization of graphs*. Prentice-Hall, 1999.

[2] I. Herman, G. Melançon, and M. S. Marshall, "Graph visualization and navigation in information visualization: a survey", *IEEE Transactions on Visualization and Computer Graphics*, Vol 6, No. 1, pp. 24-43, 2000.

[3] B. R. Preiss, *Data structures and algorithms with object-oriented design patterns in Java*, 1998, at http://www.brpreiss.com/books/opus5/html/book.html, 2004.

[4] R. Sedgewick, *Algorithms in C++*, Addison-Wesley, Boston, MA, 2002.