

Interactive Concept Mapping in ACTIVEMATH

Erica Melis
DFKI Saarbrücken
melis@dfki.de

Philipp Kärger
Universität des Saarlandes
kaphi@ags.uni-sb.de

Martin Homik
DFKI Saarbrücken
mhomik@dfki.de

Abstract: Mind maps have been used for many (including educational) purposes. Several tools are available for visualizing and creating those maps. However, evaluation and feedback is rarely available in such tools and thus, an important aspect of interactive exercises is missing. This paper presents the interactive concept map tool, ICMAP, that is integrated with the web-based learning environment ACTIVEMATH. It discusses the main diagnosis mechanisms, the generation of feedback, and the adoption of the collaborative learning tool CoolModes to develop ICMAP.

1 Introduction

Mind maps are known as a means to help humans to structure their knowledge, and to *construct* a structure in their minds. Concept maps can be considered a subclass of mind maps which visualize the relations between concepts. Some tools are available for visualizing and creating those maps. As far as we know, those tools give no or very limited feedback only. Rather, these cognitive tools targeted visualization and collaboration. In few cases, feedback can be produced by simulation software, as in SMISLE [2].

In mathematics education, understanding and discovering structure and dependencies is an immensely important aspect. Therefore, we wanted to integrate a concept map tool into the adaptive, web-based learning environment for mathematics, ACTIVEMATH [7]. The integration of a concept map tool seemed easy-to-make at first. However, several difficulties occur for meeting all requirements that reality poses and for providing reasonable feedback.

The remainder of the paper is organized as follows. We introduce requirements for ICMAP in §2 and briefly describe ACTIVEMATH and its knowledge representation in §3. Then, in §4 we detail ICMAP's interaction, feedback and verification facilities. The implementation is presented in §5. After the description of authoring exercises in §6, we conclude with a summary in §7.

2 Requirements and Informal Usages

In the EU-project LeActiveMath ¹, a collection of use cases led to a requirement analysis for an interactive concept map tool. These requirements include among others:

- input can be realized through drag'n drop
- provision of different types of feedback (indicate overall correctness, indicate only correct input locally, indicate only incorrect input locally)
- provision of placeholders for concepts and relations
- exercises can be created (authoring mode) and stored
- tool is an independent component or service for ACTIVEMATH and communicates with other components.

For the feedback and also for communicating the learner's results to the learner model component, the learner's input has to be evaluated. The diagnostic functionalities are investigated in §4.

We analyzed the various informal usages of mind maps for mathematics learning as they are described in textbooks, on the Web, anecdotal evidences, etc. Not only that mind maps have been suggested for a wide variety of situations including discussions, brainstorming, making plans, moderation, collaborative work, and project management; they – rather unsystematically – model mathematical structures, often even in ontologically inconsistent or incompatible ways (see Figure 1). Another example can be found in [5] introducing a rather ad hoc concept map for fractions, where fractional arithmetic is the central node. There is no concept for fractions itself. Instead, algorithms, interpretations and various relations indicating an application at different levels are mixed and connected to fractional arithmetic. The requests for concept maps by educationalists vary a lot and typically contain notions which are not concepts nor collections in the strict mathematical sense. Sometimes those 'mind maps' are even faulty from a purely mathematical point of view.

In order to be able to generate feedback on rather informal maps, new and aggregated 'concepts' have to be defined (e.g., by composition) and additional relations that are not available in the original domain ontology should be introducable by authors and learners (see §4.1).

3 ACTIVEMATH

ACTIVEMATH is a user-adaptive, web-based learning environment for mathematics which employs intelligent technologies. It dynamically generates interactive courses adapted to student's goals, preferences, capabilities, and prior knowledge. It integrates several components and services for explorative learning and interactive exercises. It is based on the semantic XML knowledge representation OMD \circ C.

¹www.leactivemath.org

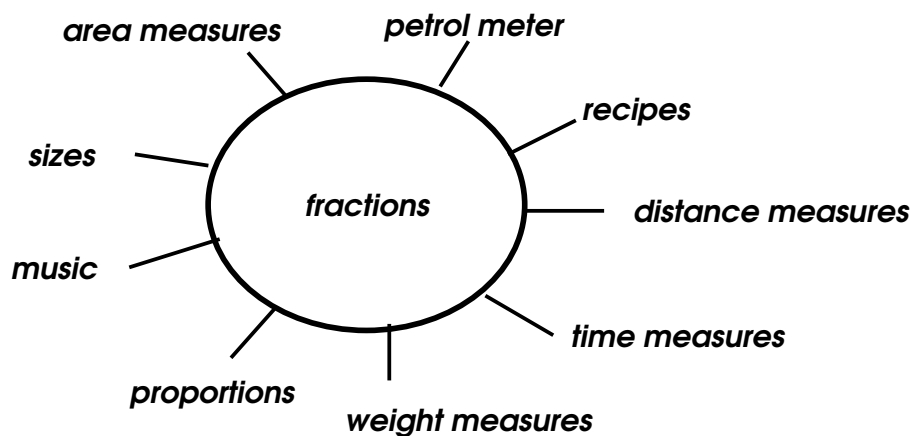


Figure 1: Mind map from Hilbert-Schule, 6th grade

3.1 Knowledge Representation

ACTIVEMATH's knowledge representation [1] is an extended OMDoc [3] - a semantic markup format for mathematical documents. OMDoc has evolved as an extension of the OpenMath² standard for mathematical expressions. In OMDoc, mathematical documents are represented by a collection of typed items annotated with metadata. These metadata annotations define mathematical and educational properties of learning objects and relate these to each other, establishing ontologies of mathematical and educational knowledge.

ICMAP deals with a reduced set of OMDoc knowledge items and relations which we describe in the following. Each item belongs to one of three layers: abstract, concept or satellite layer. The abstract layer contains OpenMath symbols, which are stored in OpenMath content dictionaries. They define elements of a formal theory. The concept layer comprises definitions and theorems. Definitions are statements that define a meaning of symbols, while theorems typically describe relations between mathematical concepts. Symbols and concept items are the main items of mathematical ontology. The satellite layer includes example and exercise items. Examples illustrate concepts and exercises are used for training and testing competencies w.r.t. a concept. Concepts and satellites are stored in ACTIVEMATH's mathematical database MBASE and are accessible via a unique identifier.

In order to establish a mathematical ontology and to connect areas of mathematics, collections of items have to be related to. In OMDoc, the element `theory` serves the purpose of assembling knowledge items in mathematical theories. Smaller theories can be assembled to bigger ones via the import mechanisms of OMDoc.

Knowledge items are linked by relations. In the following, we define a set of relations used for interaction and evaluation in ICMAP. Let S be the set of symbols, Def the set of

²see <http://www.openmath.org/>

definitions, *Thm* the set of theorems, *Ex* the set of examples and *Exc* the set of exercises. Additionally, let $C = (Def \cup Thm)$ be the set of concept items, $Satellites = (Ex \cup Exc)$ the set of satellite items, and $\mathcal{U} = S \cup C \cup Satellites$ the universe. Then, a relation that expresses a mathematical dependency³ between concepts and symbols is defined by $domPre \subseteq C \times S$. It describes which symbols are mathematically necessary in order to define the actual concept. The `for` relation links definitions to symbols, concepts to concepts, and satellites to concepts. It is defined by: $for \subseteq (Def \times S) \cup (C \times C) \cup (Satellites \times C)$. Note that a symbol can have several definitions, e.g., convergence of a function can be defined via ϵ - δ formulae or via sequences. In contrast, the `counter` relation describes the current item as a counter example for a specific concept item, i.e., $counter \subseteq (Satellites \times C)$. The `is_a` relation is a partial order on S with $is_a \subseteq S \times S$ that serves to represent mathematical hierarchies and special cases, e.g. a 'function' is a specific 'relation'.

4 Interactive Concept Mapping in ACTIVEMATH

ICMAP can be employed for exercises, for free exploration, and for demonstrations of structures and dependencies. In this section we describe the interaction and feedback facilities as well as the evaluation of user input.

A concept map in ICMAP contains nodes that represent abstract symbols, learning objects, and theories (see Figure 2). It also contains edges that correspond to relations and theory imports as described in §3.1. In addition, an author or a user can define (arbitrary) nodes and edges for a concept map.

4.1 Interactivity

In interactive exercises with ICMAP, the learner can interact by manipulating (adding, deleting, labelling, replacing) nodes. He can add a node from the palette shown on the right hand side (rhs) of Figure 3. The nodes available there are dynamically generated from the exercise representation and include nodes for symbols, learning objects, theories, and templates. Apart from template nodes, each node has an associated meaning that relates to the semantics in OMDoc. For template nodes, the learner can create the meaning of a node by dragging it from the content presented by ACTIVEMATH.

The user can also manipulate edges, i.e., add, delete or name edges in a partial map. The choice in the palette is dynamically generated (see Figure 3, bottom of the rhs interaction palette). An edge can represent relations available in OMDoc as well as deductive or 'any' relations. OMDoc relations correspond to those described in §3.1. Deductive relations, such as the equivalence relation, are not explicitly used in OMDoc but their validity can be deduced. For example, if two definitions relate (`for`) to the same symbol, then they are considered equivalent. The user indicates an equivalence by drawing an

³`domPre` is an abbreviation for the OMDoc relation type `domain_prerequisite`

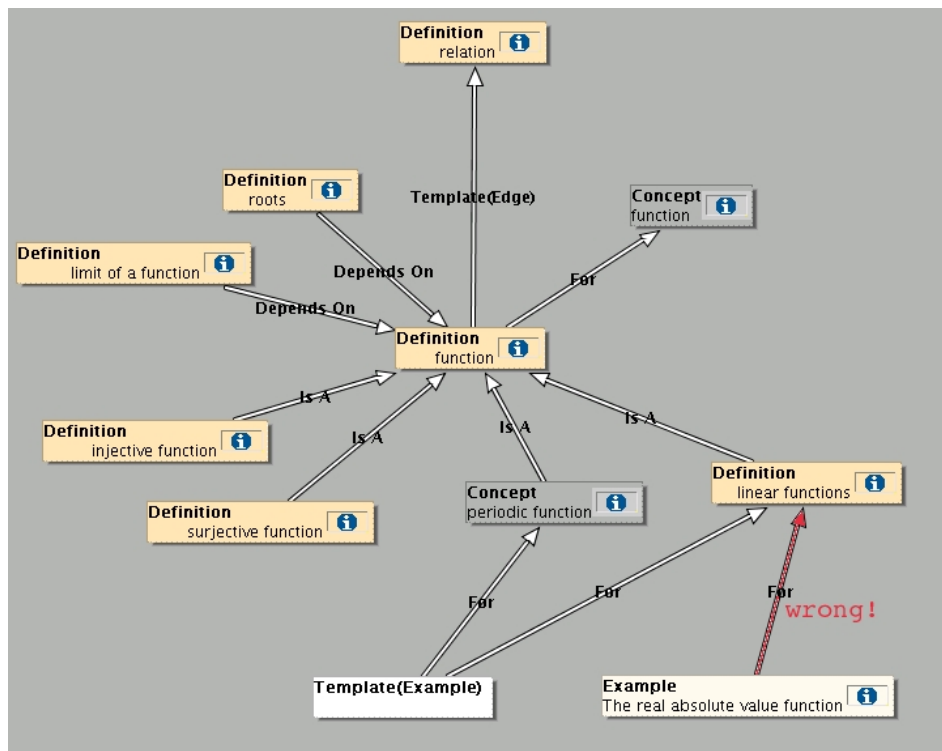


Figure 2: Screenshot from ICMAP: concept map exercise with feedback

is_equivalent edge. Another example is the belongs_to edge, which is used to relate theories. Finally, 'any' relations are presented by unspecified, no-name-edges. They enable the user to express his awareness of a relation between mathematical items, though he does not know which one.

Moreover, an author can add template edges that are supposed to be replaced by the learner. Instead of choosing from the palette, the learner can annotate edges by arbitrary relation names. Hand-writing annotations as depicted in Figure 3 are supported on tablet PCs, i.e., they can be placed anywhere, in particular they can be linked to template nodes.

The learner benefits from ICMAP not only by actively constructing a concept map and receiving feedback where possible but also from browsing (in ACTIVEMATH) the learning object information linked to nodes. Those links are presented by the info button in each node (see Figure 2). A mouse click on the info button opens ACTIVEMATH's dictionary that displays the node's knowledge object information.

At any time, the user can request a hint or verify a concept map. ICMAP offers global and local verification. The former checks all edges; the latter checks only a selected edge.

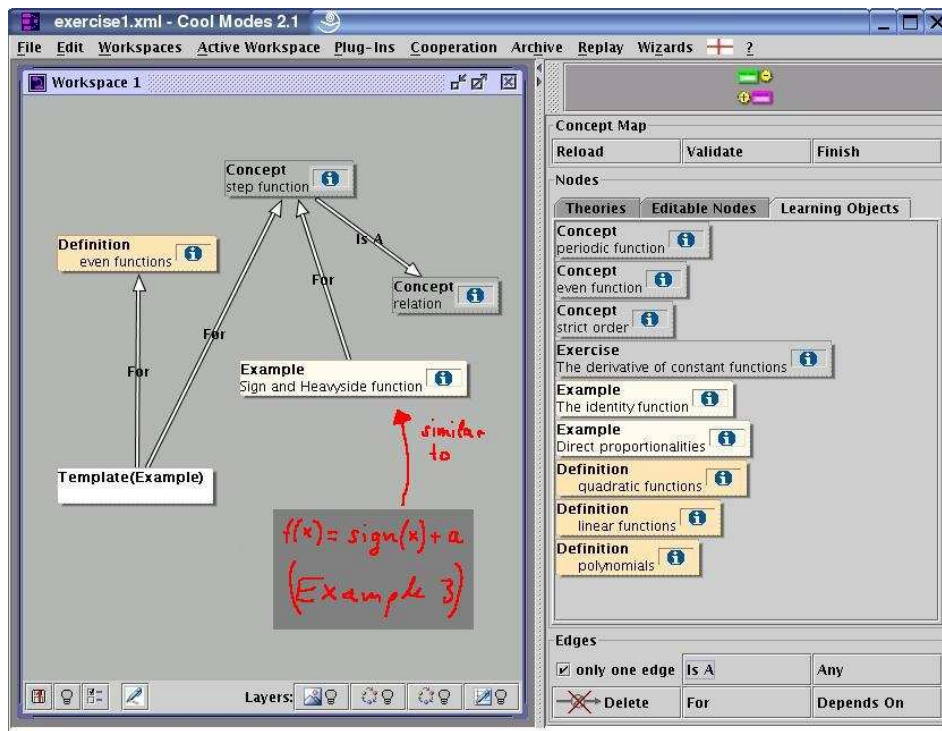


Figure 3: Screenshot from ICMAP: concept map with hand-written annotations

4.2 Feedback

There are different kinds of feedback presentation. ICMAP presents feedback graphically and textually. The former uses colors for presentation of verification results. Correctly introduced relations are rendered green; false relations are red. Additionally, edges that represent incorrect relations have local annotations, which are required, e.g., for colour blind people (see Figure 2). Textual feedback in form of an explanation is presented upon request by using the context menu of an edge. Amongst others, explanations can be “This edge has a wrong type.” or “This edge is correct, but subsumes several steps. Please elaborate.”. Also, general hints can be provided, such as “There are x important edges missing.” or “Draw an edge of type y.”.

Of course, feedback cannot be generated for all types of nodes and edges. In particular, annotations that are hand-written can not be evaluated. Still, they can be useful for the learner.

4.3 Verification

ICMAP has two verification resources: the knowledge base and authored information contained in the exercise. No verification is possible for exercises in which the learner determines his own relations or concepts (and so far for hand-written input). The verification is direct, deductive, and fault-tolerant.

By direct verification, we mean an automatic match of the learner's input with the (mathematical and educational) ontologies represented in ACTIVEMATH's knowledge representation and authored exercise. Here, metadata and imports of elements are checked to determine correctness. For example, if the learner draws an edge from node a to node b , then ICMAP requests knowledge object information linked to a , and checks if there is any relation represented by the edge that points to a knowledge object associated with node b .

Deduction can also be employed for the diagnosis in case of a failed matching, e.g., for handling transitive relations. Let $s, s_1, s_2, s_3 \in S$ be abstract symbols, $d, d_1, d_2 \in Def$ definitions, $e \in Ex$ an example, $x, y, z \in S \cup C$ any symbol or concept, and $t_1, t_2, t_3 \subseteq \mathcal{U}$ theories.

Suppose the user draws an edge from a node labelled by 'periodic function' referring to a symbol s_1 to a node labelled by 'relation' referring to a symbol s_3 . Then the edge will be evaluated as correct, since $(s_1, s_2) \in is_a$ and $(s_2, s_3) \in is_a$, where s_2 is a symbol for 'function'. The general transitivity rule for the `is_a` relation is:

$$(s_1, s_2) \in is_a \wedge (s_2, s_3) \in is_a \longrightarrow (s_1, s_3) \in is_a$$

Similar transitivity rules apply for the 'domain prerequisite' relation (`domPre`) and for the 'belongs to' relation (`belongs_to`) for theories:

$$(x, y) \in domPre \wedge (y, z) \in domPre \longrightarrow (x, z) \in domPre$$

$$t_1 \subseteq t_2 \wedge t_2 \subseteq t_3 \longrightarrow t_1 \subseteq t_3$$

The `belongs_to` edges that connect a theory node with another node are checked against the inclusion of elements in a theory.

Apart from transitivity, equivalence relations can be deduced by a simple rule: if two definitions are for the same symbol, then they are equivalent:

$$(d_1, s) \in for \wedge (d_2, s) \in for \longrightarrow d_1 \equiv d_2$$

The diagnosis can be somewhat fault tolerant through allowing correctness modulo some simple deductions. For instance, in some context it could be tolerable to accept the relation $(d, s) \in is_a$ or $(e, s) \in for$. The fault tolerance rules are:

$$(d, s_1) \in for \wedge (s_1, s_2) \in is_a \longrightarrow (d, s_2) \in is_a$$

$$(e, d) \in for \wedge (d, s) \in for \longrightarrow (e, s) \in for$$

$$(e, d) \in for \wedge (d, s_1) \in for \wedge (s_1, s_2) \in is_a \longrightarrow (e, s_2) \in for$$

5 CoolModes usage

CoolModes [10] is a client application that provides a collaborative framework designed to support discussions and cooperative modelling processes in various domains. It allows for a variety of different modelling languages and comes with a general programming interface for defining and plugging in new visual languages.

CoolMode's graphical user interface consists of a workspace, where a model is displayed and altered, and a palette, which contains all allowed nodes and edge types w.r.t. the current visual language. Modelling happens by dragging nodes from the palette to the workspace and by choosing an edge type and linking two nodes.

ICMAP is developed as a plug-in for CoolModes. We decided to integrate CoolModes with ACTIVEMATH because of its extensible interface and collaboration facilities.

5.1 Integration of CoolModes with ACTIVEMATH

ACTIVEMATH's first plug-in for CoolModes is described in [8]. It is now extended by theory handling, typed template nodes, 'any' edges, deductive and fault-tolerant verification, event-communication, internationalization, authoring support, and by an exercise markup format.

The fact that CoolModes is client-based gave us some headache and prevents it from being used as a web-service. Therefore, CoolModes is started on demand via Java WebStart from the ACTIVEMATH environment. To do this, a jnlp file containing all information about CoolModes packages, plug-ins and the exercise resource ID is dynamically generated and passed to Java WebStart, which in turn downloads CoolModes from the server and starts it. The plug-in contains an exercise loader, which retrieves the exercise from the knowledge base MBASE and dynamically generates a (partial) concept map as well as a palette, comprising a given set of nodes and edge types for representation of mathematical items and relations. Additionally, each node is equipped internally with its item identifier.

Since mathematical formulae cannot be rendered in CoolModes itself and too large item texts are inappropriate for concept maps, ICMAP displays merely the item's type and title. In addition to general template nodes, ICMAP supports typed template nodes. Only mathematical items of a particular type are allowed to be dropped on the template node. The type specification of a node is encoded in the exercise itself, such that verification is handled easily.

Communication between ICMAP and ACTIVEMATH is twofold (see Figure 4). Requests to the MBASE are synchronous, while client-events are issued asynchronously via ACTIVEMATH's event architecture [6]. The learner model can use the information to activate its update mechanism. Exercise client-events carry a reference to the exercise and they are generated after start and finish of an exercise as well as after intermediate action steps.

ICMAP's finish exercise client-event comprises information about the learner's performance. The performance is a value between zero and one. It is derived from the quotient

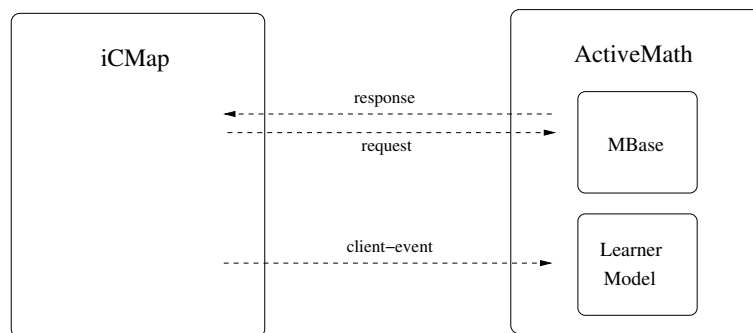


Figure 4: ICMAP communication with ACTIVE MATH

of correct action steps (create edges, substitute template nodes) and all action steps including hint invocation and evaluation. The more hints are needed and the more evaluations are processed, the lower is the performance value. Accordingly, the highest performance value is achieved, if all action steps are correct without the need of hints and intermediate evaluations.

ICMAP also issues all action step events together with the step's action information to the event handler. A registered learner model that listens for ICMAP's action events decides itself which events are filtered. Different learner models can register and analyze different events.

The verification uses a synchronous communication. For each node its corresponding mathematical item is requested from the MBASE. Further metadata are retrieved from the authored exercise. On the basis of this data ICMAP generates and presents feedback to the learner in different forms (see §4.3).

6 Authoring concept map exercises in ICMAP

ICMAP can be used not only for solving interactive concept map exercises, but for authoring exercises as well. An author can design partial concept maps and define palettes for graphical user interface.

Regarding the palette, an author can determine what kind of nodes and edges will be offered to the learner. An author is not strictly bound to ACTIVE MATH's knowledge representation. He is allowed to define own relation types, which appear as buttons in the palette. Furthermore he can define relations between mathematical items, which are not present in the MBASE.

6.1 ICMAP's concept map exercises format

Exercises in ICMAP are tightly integrated into ACTIVEMATH's knowledge representation. In fact, they are OMDoc satellites. Each exercise representation starts with an `exercise` element that carries a unique identifier and a type declaration. An extra metadata part follows, which contains amongst others a `for` relation that links the exercise to a concept. A ICMAP exercise comprises three parts: description, evaluation, and layout. The exercise main structure is shown below.

```
<exercise id="$ID" type="concept_map">
  <metadata>
    <extradata>
      <relation type="for"> <ref xref="$ID"/> </relation>
    </extradata>
  </medadata>
  ...
  <ConceptMapDescription> ... </ConceptMapDescription>
  <ConceptMapEvaluation> ... </ConceptMapEvaluation>
  <ConceptMapLayout> ... </ConceptMapLayout>
</exercise>
```

The description part describes the setting of CoolMode's palette and workspace. The palette element lists nodes and edge buttons. Nodes carry the MBASE identifier of the knowledge object. Edge buttons require an edge type description, which can represent either an OMDoc edge or an additional edge defined by the author. These type declarations are used in the workspace sub part and in the evaluation part.

The workspace element lists nodes and edges that are to be displayed, when the exercise starts. Each node has a unique identifier, which is local to the actual ICMAP exercise and carries a unique knowledge base identifier if and only if it refers to a knowledge item. Otherwise, the node is a template node that may be specified by a type. Edges describe which nodes have to be connected and the relation they represent. The relation type is restricted to the set of edge types introduces in the palette.

```
<ConceptMapDescription>
  <palette>
    <palettenode mBaseID="$ID"/>
    <theory mBaseID="$ID"/>
    <edgebutton type="$EdgeType"/>
    <edgebutton type="uses"/>
  </palette>

  <workspace>
    <node mBaseID="$ID" id="1"/>
    <node template="true" id="2"/>
    <node template="true" type="definition" id="3"/>
    <edge from="1" to="2" type="$EdgeType"/>
```

```
<edge from="2" to="3" type="uses"/>
</workspace>
</ConceptMapDescription>
```

In the map evaluation part, the author can define new relations, that may not be represented in the knowledge database and which are frequently demanded by teachers. Moreover, he can decide, which edges he considers important. In both cases, edges are described by the knowledge base items represented by their identifier and by the edge type.

```
<ConceptMapEvaluation>
  <addedRelationInstance from="$ID" to="$ID" type="$EdgeType"/>
  <importantEdge from="$ID" to="$ID" type="$EdgeType"/>
</ConceptMapEvaluation>
```

Finally, in the layout part, node positions of those nodes introduced in the workspace are assigned.

7 Conclusion

In ACTIVEMATH, ICMAP can be used stand-alone and as a tool that comes with a concept mapping exercise or example. For concept map exercises ICMAP provides feedback as far as a mapping with the knowledge base's ontology or authored relations is possible.

For verification, ICMAP queries the knowledge base and tries to match the learner's concept map. It sends verification results to ACTIVEMATH's event framework such that the learner model can use this information for updating its beliefs about the learner, e.g., about the strength of some dependencies in his mind.

First evaluation tests with undergraduate students are scheduled for the 2005/2006 semester.

7.1 Related Work

Cognitive psychologists have suggested and implemented mind maps as cognitive tools⁴ [9, 4]. The CoolModes tool which we build upon is one of these cognitive tools and it is specifically used in an educational context.

The multi-lingual Cambridge Thesaurus tool for mathematics⁵ also offers visualizations of the relations: 'broader', 'narrower', 'referencedBy', or 'SeeAlso'. However, it does not provide feedback or relation facilities, but functions more like a dictionary with visualization of (partial) concept spaces and navigation along edges. In ACTIVEMATH such a functionality is offered by the lexicon tool, described elsewhere.

⁴see <http://www.cognitive-tools.de/>

⁵see <http://thesaurus.maths.org>

7.2 Acknowledgement

We like to thank Niels Pinkwart for his friendly help with CoolModes and Jörg Müller for the first implementations for ICMAP. Thanks also go to Marianne Moormann for suggesting/retrieving several concrete mind maps from the Web and to Giorgi Gogvadze for discussion about the exercise format.

This publication is a result of work in the context of the LeActiveMath funded under the 6th Framework Program of the European Community – (Contract IST-2003-507826). The authors are solely responsible for its content.

References

- [1] G. Gogvadze, C. Ullrich, E. Melis, J. Siekmann, Ch. Gross, R. Morales. LeActiveMath Structure and Metadata Model. Deliverable D6, LeActiveMath Consortium, 2004. accessible from <http://www.leactivemath.org/>.
- [2] W.R. Joolingen and T. Jong. Design and implementation of simulation-based discovery environments: the SMISLE solution. *Journal of Artificial Intelligence and Education*, 7:253–277, 1996.
- [3] M. Kohlhasse. OMDoc: Towards an internet standard for the administration, distribution and teaching of mathematical knowledge. In *Proceedings Artificial Intelligence and Symbolic Computation AISC'2000*, 2000.
- [4] H. Mandl and F. Fischer. Mapping Techniken und Begriffsnetze in Lern- und Kooperationsprozessen. Wissen sichtbar machen. H. Mandl and F. Fischer (eds) Hogrefe, 2000.
- [5] A. Lergenmueller and G. Schmidt, editors. *Mathematik Neue Wege, Arbeitsbuch fuer Gymnasien 6. Schuljahr*. Schroedel, Hannover, 2001.
- [6] Paul Libbrecht and Stefan Winterstein. The service architecture in the activemath learning environment. In Nicola Capuano, Pierluigi Ritrovato, and Fionn Murtagh, editors, *First International Kaleidoscope Learning Grid SIG Workshop on Distributed e-Learning Environments*. British Computer Society, 2005. See <http://ewic.bcs.org>.
- [7] E. Melis, E. Andrès, A. Frischauf, G. Gogvadze, P. Libbrecht, M. Pollet, and C. Ullrich. ACTIVE MATH: A Generic and Adaptive Web-Based Learning Environment. *International Journal of Artificial Intelligence in Education*, 12 (4): 385-407, 2001.
- [8] J. Müller, M. Mühlenbrock, and N. Pinkwart. Towards using concept mapping for math learning. *Learning Technology newsletter*, 6(3):13–15, July 2004.
- [9] M. Nueckles, J. Gurlitt, T. Pabst, and A. Renkl. *Mind Maps & Concept Maps*. Beck Juristischer Verlag, 2004.
- [10] N. Pinkwart. A plug-in architecture for graph based collaborative modeling systems. In H.U. Hoppe, M.F. Verdejo, and J. Kay, editors, *Shaping the Future through Intelligent Technologies. Proceedings of the 11th Conference on Artificial Intelligence in Education*, pages 535–536. IOS Press, 2003.