Mathematical Context in Interactive Documents

A.M. Cohen, H. Cuypers and R. Verrijzer

Abstract. In this paper we introduce the concept of an interactive mathematical document. We give a formal description of such a document, which enables us to introduce the notion of a context as user and time dependent information regarding both mathematical and personal data. We also describe the realization of interactive mathematical documents within the MathDox system developed at Eindhoven University of Technology.

Mathematics Subject Classification (2000). Primary 68Mxx; Secondary 97Uxx.

Keywords. adaptivity, interactive mathematical documents, context.

1. Introduction

For some time the Discrete Algebra and Geometry group at Eindhoven University of Technology has been working on the MathDox project. MathDox stands for a format and a set of software tools for creating interactive mathematical documents, see [9]. Such documents can be viewed via a web browser. The aim of the project is to enable people to create interactive mathematical content accessible through the web. One of the most popular applications is elearning; several showcases exist, such as Wortel TU/e [37], MathAdore [26], IDA [8], and the on-line knot theory course notes [19].

In an attempt at enriching the MathDox format with adaptivity we introduce the concept of *context* in our documents. Context within MathDox documents enables MathDox to store data pertaining to visitors of the web pages, and act upon the stored state of these users. This makes MathDox more suitable for intelligent tutoring, as it enables students to be guided by an adaptive system and offers them the opportunity to create their own mathematical context.

A typical advantage of having a context for students is that a MathDox page can adjust its contents to the mathematical level of the user. Beginners in the field can be offered an elaborate explanation of the mathematical notions used on the

page. Advanced students can do with just some references instead. The context can be aware of the student's knowledge level and still allow him/her great freedom in deciding how and in which order to visit MathDox pages. Another advantage of the mathematical context in learning is that it makes it possible for students to create their own favourite running examples throughout parts of the document.

Our approach to offering MathDox users navigational freedom is based on the concept of *theory graphs*. A theory graph dictates the relations between different notions, mathematical symbols, and variables in a specific mathematical field. It gives an overview of different routes to a specific topic, which knowledge is required, and which knowledge is optional for mastering that topic. These sets of requirements and optional knowledge heavily depend on the users' current knowledge and their learning goals. MathDox' way to determine the level of a user is to determine which nodes from the theory graph are listed as being visited previously within the user's context. On the basis of these references, the MathDox page can decide at construction to include, exclude, or adapt specific parts of the content.

The theory graphs play a key role in the formal description of an interactive mathematical document given in Section 3. Before this theoretic section, we give a short introduction to MathDox in Section 2, and following up on it, in Section 4, we discuss the realization in MathDox. Section 5 provides two illustrations of the use of context in the MathDox system. In Section 6 we compare our system with other systems for interactive mathematical documents. We finish with some conclusions and discussion of further work in Section 7.

2. MathDox

Mathdox is a software system that provides means to interact with the mathematical content of an interactive document over the World Wide Web, see [9]. These interactions vary from performing computations by calling web services, see [1], to checking answers to exercises or creating and manipulating graphs of functions. MathDox combines existing interactivity of web pages with the power of mathematical computations, resulting in interactive mathematics. This is achieved by means of a scripting language and interfaces with external software, notably computer algebra systems. Triggered by user actions, pages are updated with new parameters and web services are called to generate input for the preparation of a new page. This results in adaptable pages and on the fly computations.

The source documents within the MathDox system are XML documents, written in the MathDox format. This format combines in a modular way various existing XML formats each of which contributes a useful facet for interactive documents. The XML formats used in MathDox are:

- DocBook [36], for the global structure of documents
- OpenMath [31], for semantic encoding of mathematics
- XForms [38], for user driven interactivity
- Monet [2], for communication with web services, like CAS

- Jelly [18], a programming and scripting format
- XInclude [39], for separating functionality into different files

The part of the MathDox system called *MathDox Player* makes MathDox documents accessible over the web. Its task is similar to that of a web server in the sense that both a web server and the MathDox Player offer stored documents from the server to the outside world. The main difference between a static HTML web server and the MathDox Player is that a web server offers ready-made HTML files, whereas the MathDox Player dynamically creates these HTML files. As indicated above, triggered by a user's action, the MathDox server collects the relevant data from the source document, from available user request bound information, and from service requests to (mathematical) back engines, in order to create a new view on the document. The views are presented by means of HTML pages. The MathDox actions involve a series of XSL transformations depicted in Figure 1.

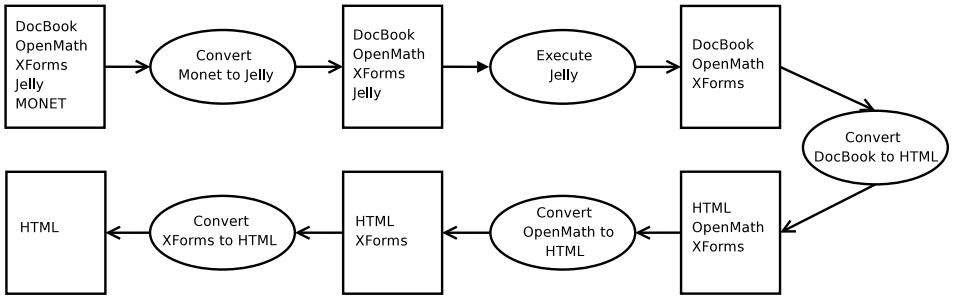

FIGURE 1. Sequence of XSLT transformations performed by the MathDox player

The MathDox Player is implemented in Orbeon Forms [32], a Java Servlet application [16] that runs in a Java Servlet container like Apache's Tomcat and JBoss.

In order to enable the users of MathDox documents to interact with the system, HTML offers various options, like buttons, text areas, and links. However, there is no standard way to communicate mathematics in a meaningful way. Modern browsers support rendering of mathematics with MathML [28], but do not provide ways to interact with the mathematics. For interaction with a large number of mathematical services offered by the system, we prefer to work with semantically rich mathematical expressions (although other formats, such as LaTeX, are supported). For authoring mathematical expressions we have developed the *MathDox formula editor* [9]. It can be employed for users to input mathematics in a natural (structured) way, and translates the input to OpenMath format without the user having to enter XML code.

For authoring MathDox sources, various tools are available, varying from the above-mentioned formula editor to LaTeX templates.

3. Interactive Mathematical Document

The MathDox system as described above is very well suited for presenting a user highly interactive mathematical pages over the web. However, when serving a document that consists of various related MathDox pages, it is desirable to add a common *context* to the different parts of the document, capturing the user's preferences, performance, browsing history, etc. Such context makes it possible to guide users through the document, present them with related examples in the various parts of the document, or offer them exercises that meet their knowledge level; see for example [12]. In this section we explain the theoretical model behind our implementation of such a context in the MathDox system. The actual implementation is discussed in the next section.

An interactive mathematical document is considered to be a collection of mathematical pages containing theorems, proofs, examples, exercises, and the like, which are delivered to the user through hypermedia. As is common in hypermedia systems, these pages need not have a strictly hierarchical structure of chapters, sections, subsections, and paragraphs. Indeed, users may take their own (guided) paths through the pages. Depending on their interest, skills or task, these paths may vary.

But not only the path that a user is following may depend on his/her personal settings, also the contents of each page are dynamically adapted to the user's context. This may, for example, imply that users can choose a favorite running example throughout a full session, or that students of a single class are studying the same pages of a course, but are offered personalized exercises, matching their individual levels.

Over the years many hypermedia systems have been developed that offer the user parts of the above-mentioned features. A standard model for such systems is the so-called Dexter (or overlay) model [17], which describes an architecture for hypermedia systems. This model also influenced the AHAM model of De Bra, Houben, and Wu [11], which focuses more on the adaptive features of the hypermedia system. Such a model can be divided into the following three parts: *domain*, *user*, and *presentation* model. We have adopted this model for the realization of interactive mathematical documents in our MathDox system.

Our domain model contains information on the knowledge domain of an interactive mathematical document. It is an abstract model for the main mathematical concepts of the interactive mathematical document, such as symbols, definitions, theorems, proofs, etc., and their relations within the interactive mathematical document. The abstract items in this model refer to collections of MathDox sources.

The user model is an abstract model for the data pertaining to a user of the system. It captures for example the logistic information of the user, such as identity, profession, knowledge level, but also information on the history of the user, such as acquired knowledge, navigation history, and scores on exercises.

Finally, the presentation model consists of a set of rules that, with input from the user model, selects and transforms parts of the domain model into a page rendered for the user.

In the following three sections we give a more precise description of the three abstract parts making up the interactive mathematical document model.

3.1. Domain

The domain model is concerned with the subject of the document. At the core of the domain is the concept of a theory graph. Related graphs are the symbol graph and the variable graph. The main ingredients of these graphs are the mathematical objects, notions, and statements used within the Interactive Mathematical Document, as well as their interdependence. Their purpose is to structure the mathematical knowledge on which a(n interactive) mathematical document is based. The author of a document needs to supply these data; for MathDox, authoring tools using the XML structure of the sources help to automate these activities. We discuss each of these three graphs in one of the next three subsections and conclude the section with an analysis of their mutual relations and the formal definition of an interactive mathematical document.

3.1.1. The theory graph. The main notion for an interactive mathematical document is the *theory graph*. In our view, a mathematical document is a collection of statements involving mathematical symbols and variables together with a logical structure on these statements. The two main types of statements are definitions and assertions. These statements are then often enlightened by examples, exercises, or remarks, or accompanied by proofs. A union of such statements hinging on one definition or assertion (possibly a compound one) comprises a *theory*, representing a node of the theory graph. In particular, at one extreme, one may view each statement as an individual theory, or at another extreme, clump an entire chapter into a theory.

Let T be a set of all theories of a document, and suppose $\sqsubseteq_T$ is the relation defined by $x \sqsubseteq_T y$ if and only if y depends on x. In a *sound mathematical context*, the relation $\sqsubseteq_T$ is a partial order on the set of symbols. Indeed, in a sound mathematical context this relation is assumed to be

- *reflexive*, i.e., $x \sqsubseteq_T x$ for all $x \in T$;
- *anti-symmetric*, i.e., $x \sqsubseteq_T y$ and $y \sqsubseteq_T x$ implies $x = y$ for all $x, y \in T$;
- *transitive*, i.e., $x \sqsubseteq_T y$ and $y \sqsubseteq_T z$ implies $x \sqsubseteq_T z$ for all $x, y, z \in T$.

We write $x \sqsubset_T y$ to denote $x \sqsubseteq_T y$ and $x \neq y$. For any partial order $\sqsubseteq$ on a set T one can define the *Hasse diagram* to be the directed graph with vertex set T, in which two vertices (also called nodes) x and y are on an edge from x to y if and only if $x \sqsubset y$ and there is no element $z \in T$ with $x \sqsubset z$ and $z \sqsubset y$. We notice that this Hasse diagram is an acyclic graph. Moreover, it is well known that the ordering $\sqsubseteq$ can be recovered from the corresponding Hasse diagram, as this ordering is the transitive closure of the adjacency relation of the Hasse diagram.

We are now in a position to give a formal definition of our theory graph. It is the Hasse diagram of the partial order $(T, \sqsubseteq_T)$.

In practice, a theory graph of an interactive mathematical document will be stored in an XML file as will be explained in Section 4.1.1.

3.1.2. The symbol graph. The next graph we want to use is the so-called *symbol graph*, i.e., a graph whose nodes are the mathematical objects and notions occurring within a mathematical document. To define these mathematical objects we will use the concept of symbols and Content Dictionaries as introduced in OpenMath and used in the emerging standard MathML 3.0 [29]. These standards provide semantic encoding of mathematical notions. In our set-up, a mathematical symbol is an OpenMath symbol defined in an OpenMath Content Dictionary. These Content Dictionaries also provide some information on the relations between the various mathematical symbols. In particular, the `FMP` of an OpenMath object provides such information; see [31]. As, in many cases, this information is too restricted, we capture this dependence in our symbol graph. (Here an author should provide the extra dependence relations.)

Let S be a set of OpenMath symbols, and suppose $\sqsubseteq_S$ is the relation defined by $x \sqsubseteq_S y$ if and only if the mathematical notion of y depends on that of x.

As before, the relation $\sqsubseteq_S$ on the set of symbols S in a sound mathematical document is assumed to be a partial ordering. So, we can define the *s*ymbol graph to be the Hasse diagram of the partial ordering $\sqsubseteq_S$ on the set S of OpenMath symbols in the interactive mathematical document.

An example of a symbol graph can be found in Section 4.1.2.

3.1.3. The variable graph. The third and last graph that we consider in our domain is the graph whose vertices are the (mathematical) variables inside our mathematical document.

A variable inside a document can be given a value by the user, by the source, or by some external program depending on other variables. For example, the values can be randomly chosen for generating exercises, or input by the user as answers to exercises or for running examples.

These variables are ordered by a relation $\sqsubseteq_V$, where $x \sqsubseteq_V y$ means that variable y depends on variable x. Here dependence means that the value of x is used to determine the value of y. Examples are the function case, such as $y = \cos(x)$, or instances of mathematical notions introduced, such as $y = \text{VectorSpace}(3, x)$, where x is a variable whose value should be a field, for instance $x = \text{GaloisField}(9)$. As in the previous cases, this relation is considered to be a partial ordering. The Hasse diagram of this ordering is called the *variable graph*.

Finally, to each variable a type can be attached—but this is optional. The type is used to specify the kind of value that can be assigned to a variable. For instance, the specification that the above-mentioned variable x should be a field. Another example is the specification that the type of a variable should be integer, in which case `<OMI>23</OMI>` would be an allowed value, but not

```
<OMA><OMS name='plus' cd='arith1'/><OMI>20</OMI><OMI>3</OMI></OMA>.
```

The above shows that the type of a variable should not be understood in the way of a classical type theory. In our approach it refers to the type as an OpenMath object, and is concerned with questions like: Which OpenMath symbols are used? How are they nested? The type checking is performed by pattern and anti-pattern checks. An example of how this type checking is used can be found in Section 4.1.3.

Besides being useful for checking user defined input, the typing mechanism is also convenient to cast outputs of external programs to OpenMath expressions of the required type. For instance, if the software package GAP ([15]) is used to assign a value to the variable a whose type is a 7×7 matrix with values in x (as above) and returns a list of 7 lists of length 7 containing elements from GaloisField(9), then it makes sense to transform the list of lists to such a matrix before assigning the value from GAP to a.

3.1.4. Synthesis. Until now, the theory, symbol, and variables graphs are independent of each other. However, given a mathematical document, one encounters, in each assertion, example, proof, exercise, or remark of the document, a number of mathematical symbols and variables. Moreover, each symbol or variable can occur in various theories of the document. This is formalized by a relation 'occurs in' between the union of the sets of symbols and variables, and the set of theories.

The reverse relation from the set of theories to the set of symbols is called the relation 'contains'. For $s \in S \cup V$ and $t \in T$ we write $s \rightarrow t$ if s occurs in t, and $t \leftarrow s$ for the reverse relation, meaning t contains s.

Let Θ denote a map that assigns to each symbol s the set of theories in T containing s. So for $s \in S$, we have

$$\Theta(s) = \{t \in T \mid s \rightarrow t\}.$$

This set is called the *scope* of the element s. Similarly, we let Σ_S denote the map that assigns to each theory $t \in T$ the set of symbols from S, which we call the symbols occurring in t. So

$$\Sigma_S(t) = \{s \in S \mid s \rightarrow t\}.$$

Similarly, Σ_V denotes the map which assigns to each theory $t \in T$ a set of variables from V, which we call the symbols occurring in t. So

$$\Sigma_V(t) = \{v \in V \mid v \rightarrow t\}.$$

Let x be a symbol or variable and consider the set $\Theta(x)$. If this set contains a unique minimal element with respect to the poset of the theory graph, then this is called the *introducing theory* for x and it is denoted by $\theta(x)$.

We call the relation $\rightarrow$ *sound*, if

- for all symbols or variables $x \in S \cup V$ there exists an introducing theory $\theta(x)$;
- for all symbols s, s' we have that $s \sqsubseteq_S s'$ implies $\theta(s) \sqsubseteq_T \theta(s')$;
- for all variables v, v' we have that $v \sqsubseteq_V v'$ implies $\theta(v) \sqsubseteq_T \theta(v')$;
- for all variables v and each symbol s in the type of v, we have $\theta(s) \sqsubseteq_T \theta(v)$.

Soundness is useful as a sanity check on an interactive mathematical document; it prevents a reader from going astray because of obvious inconsistencies in the set-up. Our notion of soundness is primarily a kind of well-formedness of the mathematics served and has little bearing on formal mathematics.

The mathematical context emerges as the structured sets of symbols and variables, as well the values of the variables, pertaining to the theory visited by the user. As this is dependent on the user, further treatment of it belongs to the next subsection.

3.2. User

The second of three models that together form an Interactive Mathematical Document is the *user model*. It consists of logistic information, knowledge information, and mathematical context.

The *logistic information* contains information about the user, like identity, name, profession, affiliation, address, student level, etc.

The *knowledge information* is information attached to every node of the theory graph. Here, one can think of information, whether a node is mastered, visited, or ready to be read. Here is also the information on where the user is at present. The student level may change as a result of the knowledge information.

Finally, *mathematical context* is the information about the mathematical context that the user has created by visiting the document, setting variables, etc. In fact, inside the mathematical context we store the *values* of each of the variables inside the set V of variables in our document. Here, the difference between the symbol graph and the variable graph becomes quite clear: the knowledge information tells us which nodes t the user has visited, and so $\Sigma_S(t)$ can be determined from it. In contrast, the knowledge of $\Sigma_V(t)$ is insufficient and needs to be complemented with the values of the variables that it lists. As these values may vary when the user progresses and can be called at each instance, these belong to the user model and so are contained and maintained in the mathematical context.

The variable graph gives us the relations between the various variables. It shows us, that the only variables that may be changed freely, without harming this relation, are the minimal elements $x \in V$ with respect to the relation $\sqsubseteq_V$. Without any problem these variables can serve as input variables. (An element x is called *minimal* with respect to a partial order $\subseteq$ if and only if for all y with $y \subseteq x$ we have $y = x$.) Changing the value of variable x that is not minimal, can only be done within the context defined by the variables y preceding x, i.e., those y with $y \sqsubseteq_V x$. This clearly can put restrictions on the values that one can assign to x. Moreover, if a variable x is changed, then the variables that have to be updated are exactly those variables z with $x \sqsubseteq_V z$. A topological sorting of these elements will provide a linear ordering in which we can update the elements.

In fact, the most important way of interaction that our interactive mathematical document provides to the users is the interaction with the user model. By visiting nodal pages and preforming well on exercises, they can alter the value of the variables in the knowledge information. Moreover, users can also change the

value of the mathematical variables and create in this way their own mathematical context.

For example, within an interactive document on permutation groups, the user visits an example in which he/she can choose two permutations g and h inside the symmetric group Sym_6 as well as an integer $i \in \{1, \ldots, 6\}$. Within the example, the orbit O of i under the action of the subgroup G of Sym_6 generated by g and h is computed. This mathematical context created by the user carries over to the next examples, where the stabilizer H in G of i is determined and where it is shown that $|O| = |G/H|$. The example is further elaborated in Section 5.2. Each time a user visits a page, the information inside the user model is used to update this page.

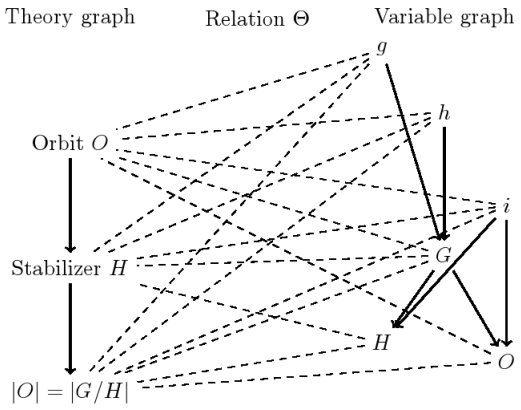

FIGURE 2. Part of the theory and variable graph and the relation Θ

3.3. Presentation model

In our model a user visits the nodes of the theory graph. According to the data in the user model, a MathDox page is created for the user that reflects the content of the node but is adapted to the user's needs and/or preferences. The creation of the page is done by a set of rules that take as input the theory that the user is visiting and the information about the user that is present in the user model.

For example, if a user wants to visit a nodal page, the system may check whether all the symbols occurring in the page are known to the user. If not, links to the introducing theories of these symbols are added. Or, if the user has not mastered all nodes on which the requested page is depending, he or she will first be guided through these nodes before granting him/her access to the page. The presentation model of our interactive mathematical document is responsible for creating the pages, and keeping the user model up to date.

Following [11], we let the presentation model of an interactive mathematical document consist of a set of rules that carry out the above tasks. So, a rule in the presentation model takes as input a triple (t, U, m), where t is the theory that is visited, U is the (relevant subset of the) data stored in the user model, also called the *context*, and m is the moment on which the rule should be applied. The moment can have two values, namely *pre* or *post*. The value *pre* refers to the moment that the user visits the theory, the value *post* refers to the moment where the user leaves a theory and moves to the next theory (which might be the same).

For each value of t and U a page is produced and offered to the user consisting of an ordered collection of renderings generated by fragments on input (t, U, pre). Similarly, there is a rule that on input $(t, U, post)$ guides the user to a new theory. Besides these mandatory rules, there are also rules in the presentation model that update the information in the user model.

Within a MathDox interactive mathematical document we implement our presentation model as functions inside Jelly tags; see Section 4.3. Our system is set up in such a way that it also allows for input from other (external) systems that could rule the presentation of a page.

4. Interactive Mathematical Documents in MathDox

An interactive mathematical document within the MathDox system consists of a set of MathDox sources, together with some tables stored in an XML database supporting XQueries. Together they capture the domain, user, and presentation model of the interactive mathematical document.

The main tools to realize the envisioned interactivity and adaptivity in a MathDox interactive mathematical document are functions implemented in Jelly tags and so-called MathDox fragments. Fragments are (reusable) MathDox sources. Besides text, such fragments can contain calls to Jelly programs taking care of parts of the presentation model. The fragments are designed in such a way, that they can be included or called with parameters inside each MathDox source. Here, *calling* is used to indicate that the results of an invocation are used, and *including* means that the source of the fragment is dragged into the main document during processing.

In the remainder of this section, we explain how each separate part of the formal set-up of an interactive mathematical document as discussed in the previous section is realized in MathDox.

4.1. Domain

4.1.1. The theory graph. The XML source files of a MathDox interactive mathematical document can be divided into two parts, the nodal pages and fragments.

The *nodal pages* are in a one-to-one correspondence with the theories of the interactive mathematical document. They represent the theory graph of the interactive mathematical document. The relation $\sqsubseteq_T$ between these pages is captured

in a table, stored in an XML database. Here all nodal pages have a unique identifier (or name). Nodal pages can call and include further pieces of MathDox source, called *fragments*.

Below, a typical nodal page is displayed; it corresponds to a node named 'knot' in the theory graph specified by the value 'knottheorygraph.xml' of the parameter 'theoryGraphURL'. The page includes a fragment with the name 'knot.mdf', which contains a text on knot theory. Moreover, it calls the fragment 'successors.mdf' twice with parameters 'direction' set to Backward and Forward, respectively. This fragment computes the set of nodes in the theory graph that are on a directed edge of the theory graph, pointing to, respectively, starting in 'knot'.

```
<article>
  <c:set var='theoryName'>knot</c:set>
  <c:set var='theoryGraphURL'>knottheorygraph.xml</c:set>
  <mdc:include-fragment fragment='successors.mdf'>
     <c:set var='direction'>Backward</c:set>
  </mdc:include-fragment>
  <mdc:include-fragment fragment='knot.mdf'/>
  <mdc:include-fragment fragment='successors.mdf'>
     <c:set var='direction'>Forward</c:set>
  </mdc:include-fragment>
</article>
```

Inside this nodal page two fragments appear: 'successors.mdf' and 'knot.mdf'. These are called at the moment that a user is entering a page (pre-action). The fragment 'knot.mdf' is merely included. Inclusion may be considered to be a default action of the presentation model. The first and last includes are calls of the fragment 'successors.mdf', which contains a function that computes the set of pages that have been visited or can be visited and has input parameters 'direction' and 'theoryName', which, as is the case in our example, might be defined outside the fragment.

This function can be extended so as to work with more input parameters, like 'skills of user', that are part of the user model. The user can leave the page (post-action) by choosing a link generated by the 'successors.mdf' fragment. The example will be further discussed in Section 5.1.

The subdivision of an interactive mathematical document into theory graph, nodal pages, and fragments makes it possible to reuse MathDox sources in various interactive mathematical documents. If one wants to include some parts of an interactive mathematical document into another document, one just has to refer to the right subset of nodal pages into the theory graph attached to the new interactive mathematical document. Of course, this can only be done in a sound way, as described in Section 3.

4.1.2. The symbol graph. As we already indicated in the previous section, mathematical objects in the MathDox sources are represented by OpenMath expressions. The leaves of each OpenMath expression tree are variables and symbols. Symbols

are defined in *content dictionaries*, see [31]. Within the XML encoding of OpenMath as used in MathDox, the name of a symbol and the content dictionary in which it is defined are specified as attributes. The ordering $\sqsubseteq_S$ on the set of symbols S inside a MathDox interactive mathematical document takes into account the OpenMath symbols needed for a formal definition of each element of S. Since OpenMath does not provide such a definition, the individual choices need be made by the author. Figure 3 shows a small symbol graph related to an example recurring in Subsection 5.2.

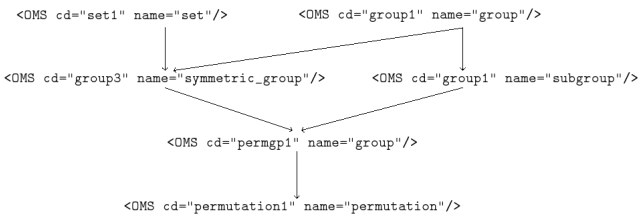

FIGURE 3. A part of a symbol graph on group theory

The map Θ restricted to the set S of symbols is stored as an XML table that can be produced by XSLT-transformations on the source documents. The conditions for an interactive mathematical document to be sound require that each symbol has a unique introducing theory. This introducing theory $\theta(s)$ for a symbol s can be obtained by computing the gcd of all elements in $\Theta(S)$ with respect to the relation $\sqsubseteq_S$. If this gcd is contained in $\Theta(s)$, then we have found the introducing element. Otherwise we add the root element of the theory graph to $\Theta(s)$ and consider the element s to be part of the prerequisites of our document.

4.1.3. The variable graph. The MathDox fragments can contain many mathematical variables. Together these form the set V of variables of the interactive mathematical document. The relation $\sqsubseteq_V$ between these variables is given by $v \sqsubseteq_V w$ for $v, w \in V$ if and only if, in the definition or expression of w, or in the restrictions imposed on it, a call is made to the variable v.

To illustrate this, consider the permutations g and h from the example in Section 3.2. They are variables in the variable graph. The variable `G` is defined by

```
<set name='G'>
    <OMOBJ>
      <OMA><OMS cd='permgp1' name='group'/>
           <OMS cd='permutation1' name='right_compose'/>
           <out name='g'/>
           <out name='h'/>
      </OMA>
    </OMOBJ>
</set>
```

As the variables g and h occur in the definition of G, we have $\mathtt{g} \sqsubseteq_V \mathtt{G}$ and $\mathtt{h} \sqsubseteq_V \mathtt{G}$. This relation between the variables defines the variable graph of the document. The variables together with the relations $\sqsubseteq_V$ and Θ are stored in a database table.

As in the case of the symbols, the information on the map Θ can be automatically retrieved from the MathDox sources. Furthermore, we can compute the introducing theory for all variables in V and store this information in a table. For practical reasons, we use the identifier of the introducing theory, as part of the information to identify our variables by unique global names.

The knowledge on the introducing theories, the information on the types of the variables and on Θ enables us to perform a soundness check on our interactive mathematical document.

Knowledge of the scope $\Theta(v)$ and its minimal element $\theta(v)$ is also valuable information for an author who wants to alter or extend the existing interactive mathematical document. Indeed, with this knowledge of the variables, he/she knows which variables can be used or have to be introduced.

The typing of the variables is done with the help of OpenMath. The type of a symbol is just a collection of OpenMath symbols (or combinations thereof) that may be present (or forbidden) in the value that will be assigned to the variable.

For example a variable of type integer may be of the form `<OMI>*</OMI>` or

```
<OMA><OMS name='unary_minus' cd='arith1'/><OMI>*</OMI></OMA>.
```

Type checking has been implemented by the use of Jelly [18] and XPath [40]. Indeed, checking whether a variable *var* is assigned a value of the type integer as described above can be checked by the Jelly tag

```
<x:if select='$var/OMA/OMS[@name='unary_minus']/following-sibling::OMI
      or $var/OMI'>True</x:if>
<x:if select='not($var/OMA/OMS[@name='unary_minus']/following-sibling::OMI
      or $var/OMI)'>False</x:if>
```

4.2. User

The user model consists of mathematical context, logistic user information and knowledge information. Needless to say, every user has his/her own instance of the user model, so that all his/her data stored in the model is private.

Inside the mathematical context we store information on the *values* of each of the variables inside the set V of variables in our document. These values may vary when the user progresses and can be called at each instance.

The mathematical context controlled by a list of mathematical context variables, where all variables are represented by at least one tuple of the form *(Variable ID, Value, Timestamp)*. Each change of the value of a variable will lead to a new tuple being added to the user model, creating a list of tuples representing the mathematical context of any variable at any given time. The value and timestamp are dynamic information attached to the variable, which is uniquely identified by the variable ID. This *variable ID* is a reference to the variable in the variable graph in which additional static information like its definition and type, a list of usage by nodal pages, and conditions on type and value are stored.

The logistic information contains information about the user; it is stored as tuples of the form *(user's ID, date of birth, address, education, affiliation, profession, level of knowledge, Timestamp)*. After each change, a new tuple is added to the database. Of course this information can be extended by other fields if deemed necessary. Finally, knowledge information is the set that indicates which nodes from the theory graph are visited or mastered by the user. Also knowledge information is timestamped.

All information belonging to the user model is stored in, and can be retrieved from an XML database supporting XQuery [41]. The user model is updated each time a user performs an action. Indeed, once a visitor enters or leaves a nodal page, the information in the user model is updated. After a topological sort of the variables, such an update can be performed in a linear order.

The information inside the user model is input for the presentation model to create a view on particular parts of the document. The user information adapts this view to the user's knowledge, skills, and mathematical context. Moreover, as all information is timestamped, it is easy to replay previous views on the document.

4.3. Presentation

In the presentation model of our interactive mathematical document, input from the domain and user model are used to create a page that is being served to the user.

The composition of the page is ruled by various functions inside the fragments, which can be called using Jelly tags and take the information in the user model as input. We have already seen an example in Section 4.1.1. As a result a new MathDox source is created that is then transformed to an HTML document as explained in Section 2; in particular, see Figure 1. As only this HTML code is served to the user, the MathDox source code need not be available to the user.

5. Examples and use cases

In this section we present two larger examples and provide some comments on the use of MathDox within the system Wortel TU/e [37]. We start with navigation in knot theory course notes [19] and continue with a permutation group theory exercise from Interactive Documents on Algebra [8].

5.1. Knots

Recently, an existing MathDox document on knot theory was transformed into an interactive mathematical document enriched with context; see [19]. The theory graph of the interactive mathematical document is depicted in Figure 4.
To each node of this theory graph, we have attached a unique nodal page, which typically looks as follows.

```
<article>
  <mdc:include-fragment fragment='template.mdf' var='result' >
    <!-- all set variables in this body are parameters for the fragment -->
```

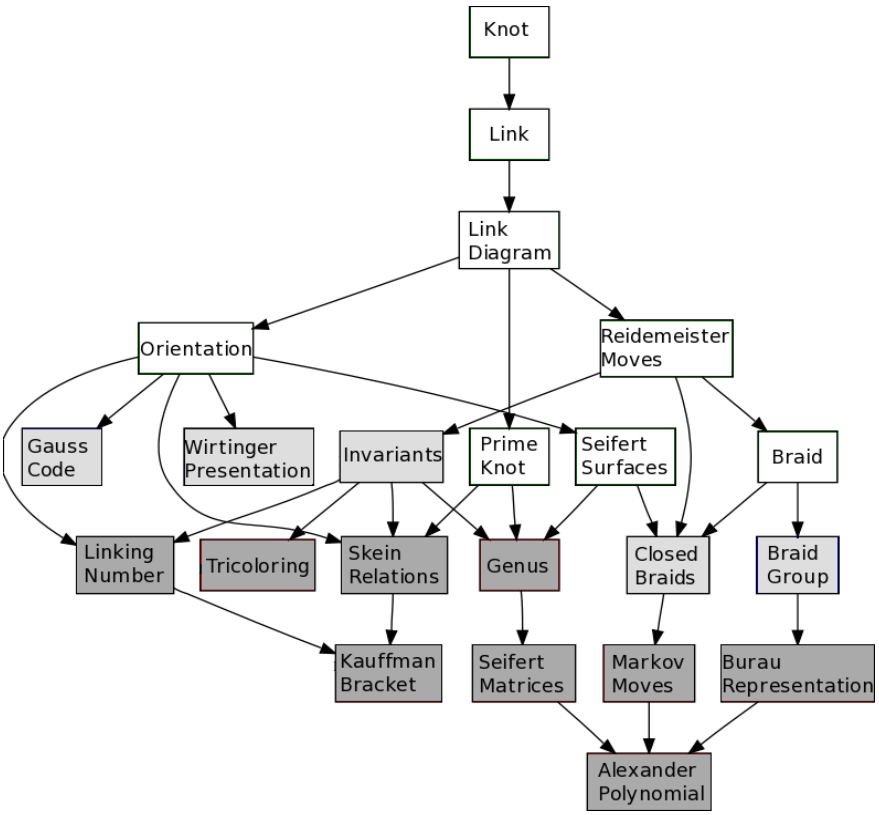

FIGURE 4. A theory graph customized for a user in such a way that the theories mastered by the user are white, those that the user can visit are gray, and the remaining theories are dark gray.

```
    <c:set var='theoryName'>primeknot</c:set>
    <c:set var='theoryGraphURL'>knottheorygraph.xml</c:set>
    <c:set var='fragmentAddress'>primeknot.mdf</c:set>
    <c:set var='passExercise'>primeknotpassexercise.md</c:set>
  </mdc:include-fragment>
</article>
```

This nodal page calls only one fragment, 'template.mdf', which takes care of the following tasks for the logged in user:

It determines the position of the nodal page in the theory graph using the values of the parameters 'theoryGraphURL' and 'theoryName', which are pointers to the theory graph and a node in this graph, respectively. Then it calls a fragment defined by the value of the parameter 'fragmentAddress'. This fragment contains the actual contents to be displayed to the user. Moreover, an exercise related to the contents of the page using the parameter 'passExercise' is specified. With input

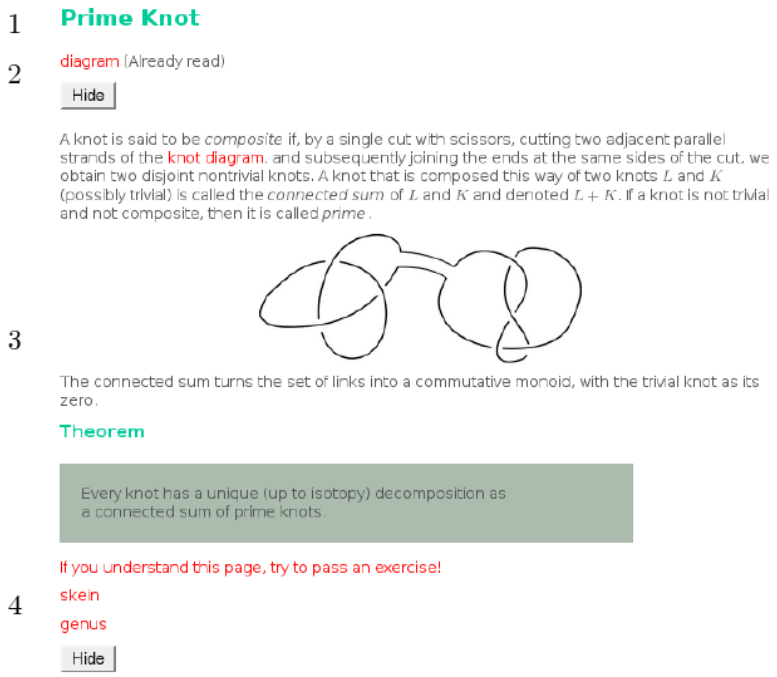

FIGURE 5. A view on a nodal page

from the user model the nodes of the theory graph that have been accessed (and mastered) by the user and those that the user is now ready to visit are determined. And finally, upon each action of the user the necessary calls to the databases and updates of the databases are performed.

In the user model of this interactive mathematical document we store the user ID. The knowledge information contains the names of the theories visited by the user, as well as those mastered by the user.

Next we explain how the page is constructed from the code in the template fragment. We have omitted parts of the code to improve readability. Figure 5 shows the resulting page. It begins with the title of the page to be rendered, which is derived from the theory node in the theory graph. Next it assigns a Boolean value to the variable 'CanGoOn' that indicates whether the user has the required preknowledge. According to the value of this Boolean, it either serves the contents (in 'fragmentAddress') or directs the user to a theory closer to the root in the theory graph.

```
<title>
  <!-- 1 Title of this topic-->
  <mdc:getTheoryProperty theoryGraphURL='${theoryGraphURL}'
        theoryName='${theoryName}' propertyName='verbose'/>
</title>

<!-- if all required knowledge is present CanGoOn is set to true-->
<mdc:available-theories theoryGraphURL='${theoryGraphURL}'
          theoryName='${theoryName}' direction='backward' var='CanGoOn'/>
  <c:choose>
    <c:when test='${CanGoOn}'> <!-- preknowledge is present -->
      <!-- 2   show previous topics leading up to the current topic-->
      <mdc:include-fragment fragment='successors.mdf'>
        <c:set var='buttonName'>Predecessors</c:set>
        <c:set var='direction'>Backward</c:set>
      </mdc:include-fragment>

      <!-- 3   show the content identified by the fragment address -->
      <mdc:include-fragment fragment='${fragmentAddress}'/>
      <para> <!-- link to exercise -->
        <ulink url='${passExercise}'>Try to pass an exercise!</ulink>
      </para>

      <!-- 4   shows follow up topics available after current topic -->
      <mdc:include-fragment fragment='successors.mdf'>
        <c:set var='buttonName'>Successors</c:set>
        <c:set var='direction'>Forward</c:set>
      </mdc:include-fragment>
    </c:when>
    <c:otherwise>
      <!-- preknowledge not present, redirect -->
      <para>Sorry, You first need to study the following sections.
        <mdc:include-fragment fragment='successors.mdf'>
          <c:set var='buttonName'>Predecessors</c:set>
          <c:set var='direction'>Backward</c:set>
          <c:set var='list'>true</c:set>
        </mdc:include-fragment>
      </para>
    </c:otherwise>
  </c:choose>
```

Here, a theory will be mastered by the user who passes a designated exercise attached to the theory.

The functions implemented in the 'successors.mdf' fragment are a part of the presentation model. Since the user has to log in to the system, these functions have access to the information in his/her personal user model. In this way we have realized adaptive navigation in the knot theory course notes.

5.2. Interactive Documents on Algebra

Subsection 5.1 on the knot theory course notes is a simple example of context enrichment. It focuses mainly on the theory graph and the prerequisite knowledge for viewing a certain text, i.e., it provides adaptive navigation. Of course, the applications of context are much wider. Some other features appear in the context enriched version of IDA (Interactive Documents on Algebra) under construction [8]. Both in the knot theory example and IDA, adaptivity and interactivity are being added in such a way, that users can work within the same mathematical context between various sessions and pages.

By way of example, we consider a student of IDA working with permutation groups. Somewhere during his/her visits of the document, the student has chosen two permutations, g and h inside the group Sym_6. This information is stored in the mathematical context of the student.

In a next session, the student might be ready to do a multiple choice exercise on permutation groups. The student is offered the following exercise

What is the G-orbit O of i?

Here G is a subgroup of Sym_6 generated by the permutations g and h as in Section 3.2 and i an integer in $\{1, \ldots, 6\}$. We can use the values of the variables of the variable graph displayed in Figure 2. The values of the variables O and H inside this graph will be calculated at the moment that the student needs them. For the actual computations with these variables, the computer algebra system GAP [15] is being used.

To construct an alternative to the correct answer O, we randomly select elements r_1 and r_2, respectively, in the set $\{1, \ldots, 6\}$ and form the set $Y = (O \setminus \{r_1, r_2\}) \cup (\{r_1, r_2\} \setminus O)$, which certainly is different from O. This set Y is stored in the mathematical context as a variable named 'wrong_orbit'.

Finally, the multiple choice question is created by use of the following template.

```
<mdc:include-fragment fragment='mcfragment.mdf'>
  <x:parse var='question'>
    <question> What is the orbit of <out name='i'/> under the action
               of the group <out name='G'/> generated by the permutations
               <out name='g'/> and <out name='h'/>?
    </question>
  </x:parse>
  <x:parse xmlns:x='jelly:xml' var='answer'>
    <wrapper>
      <mc-answer><mc-option>Answer A</mc-option>
                 <mc-text> <out name='wrong_orbit'/></mc-text>
                 <mc-verdict>Incorrect</mc-verdict>
                 <mc-feedback>Wrong!</mc-feedback>
      </mc-answer>
      <mc-answer><mc-option>Answer B</mc-option>
                 <mc-text><out name='O'/></mc-text>
```

```
                <mc-verdict>Correct</mc-verdict>
                <mc-feedback>Yes, you are right!</mc-feedback>
      </mc-answer>
    </wrapper>
  </x:parse>
</mdc:include-fragment>
```

Any change of the variables g, h and i triggers not only a new exercise, but also new examples as described in Section 3.2. As the information in the user model is timestamped, the student (or his/her teacher) can replay the examples and exercises anytime.

5.3. Wortel TU/e

Wortel TU/e [37] is a mathematical elearning environment offering students a wide range of automatically graded exercises. The system is built on the MathDox software.

Wortel TU/e contains a set of exercises on linear algebra in which a limited and preliminary version of our context is implemented. By the use of this context, these exercises provide detailed feedback to the users. A study of the impact of this extended feedback has shown that it has positive effects on motivation and learning of the students. For the results of this study, we refer the reader to [5].

6. Related work

There are various systems offering interactive mathematical documents, ranging from exercises systems like MapleTA [25], STACK [35], Aplusix [3] or the Digital Mathematics Environment [13], and the notebooks and worksheets offered by CAS like Mathematica [27], Maple [24], or Sage [33], to learning environments based on SCORM [34] included in Learning Management Systems. We will compare our system with those that, to the best of our knowledge, also provide interactive and adaptive presentation of mathematical documents for intelligent tutoring.

The OMDoc format [7, 20] is a semantics-oriented representation format and ontology language for mathematical knowledge. It is, just as MathDox, based on OpenMath and MathML. It is especially tailored for capturing the structure and interrelations of mathematical objects and statements, i.e. definitions, theorems, and proofs. The use of OMDoc in mathematical elearning materials is discussed in [21].

The OMDoc format is also used in the ActiveMath and LeActiveMath systems, see [30, 23], as well as Panta Rei [22]. Within ActiveMath and LeActiveMath automatic course generation is based on hierarchical network planning producing personalised courses that adapt to the user's learning goals and competencies.

These OMDoc based systems offer a rich format for structuring mathematics. They also offer various forms of interactivity. However, the way MathDox provides interactivity and, in particular, adaptation by the use of Jelly and Monet

queries and the discussed mathematical context, seems to be new. In particular, our running examples are a novel feature.

ConneXions [4] is an educational knowledge repository. Documents within ConneXions are written in CNXML [6], a lightweight XML markup language for educational content. CNXML embeds MathML as well as OpenMath for the representation of mathematical objects, but does not provide markup for the mathematical structure of the document. Adaptivity is provided by so called 'lenses'. These lenses are used to select appropriate content for a user according to his membership to a specific mathematical community.

The Trail Solutions project [10] also aimed at developing a technology for the generation of personalized teaching materials, notably in the field of mathematics. However, its starting point was existing static documents. The background of this project is the 'Slicing Book Technology'. Existing (linear and static) documents are sliced into pieces and then presented in a nonlinear fashion.

Both ConneXions as Trail Solution only provide tools for static documents. Their dynamics are restricted to navigation through the document.

Emilea-Stat [14] is an elearning environment in applied statistics. It contains a large database of small modules containing for example a single theorem, example or exercises. As the relations between these modules are stored in the database, it is possible to automatically create (personalized) courses. The various concepts are accessible in three layers of abstraction: elementary, basic, and advanced. The interactivity in Emilea-Stat is mainly offered by Java applets.

The various systems described above all offer quite similar possibilities for automatic creation of interactive mathematical documents and ways of navigation through these documents. However, the ways to interact with the various parts of such a document vary a lot. The main features that distinguish MathDox from these other systems seems to be the notion of (mathematical) context and the three graphs.

7. Conclusion

In this paper we discussed a formal model for interactive mathematical documents and explained how it is realized in the MathDox system. In line with the Dexter model [17] and the AHAM model of [11], it is divided into three parts: the domain, the user, and the presentation model. The main new ingredients in our formal model are the three graphs that are at the core of the domain: the theory, symbols, and variable graphs. These graphs describe the relations between the main theories, the mathematical symbols, and the variables in an interactive mathematical document. Context is then interpreted as both the information regarding the user visiting an interactive mathematical document and the state of the mathematics that is active during that visit. The mathematical context can conveniently be defined by means of the three graphs. Laws regarding relations between these graphs are described that capture soundness of the mathematical

context. This implies control over the proper order of introduction of mathematical notions and variables.

We have also discussed the realization of the model in the MathDox system, which is being developed at the Eindhoven University of Technology. Here the theory graph is the backbone for navigation through a document. This is exemplified by interactive navigation features through notes of a knot theory course. The use of variables consistent with the mathematical context is illustrated in a group theory exercise from IDA, an Interactive Documents on Algebra set up in MathDox.

Although the construction of tools for the context handling in MathDox is still under development, it shows promising advantages over the usual writing of a document. For instance, an overview of where new symbols are introduced can be fully automated so that, when a teacher compiles his/her favourite treatment of the text for a course from a MathDox interactive mathematical document, he/she can invoke a check whether this choice is mathematically sound (in the sense that no symbols or variables are used before they are properly introduced) at any desired moment.

Various aspects need further elaboration. We mention a few of the many unresolved issues involved in the writing of powerful interactive mathematical documents that we would like to address. At the end of the Subsections 3.1.3 and 4.1.3 we discussed the concept of typed variables, but the management of possible types and their dependencies has not been worked out. Also the various presentation models have to be worked out. Moreover, as our context software is still in a development phase, we have not been able to evaluate usability. But the MathDox software has been successfully used within the mathematical elearning environment Wortel TU/e [37] of the TU/e.

References

[1] O. Caprotti, A.M. Cohen, H. Cuypers, M.N. Riem, H. Sterk, *Using OpenMath Servers for Distributing Mathematical Computations.* Proceedings of the Fifth Asian Technology Conference in Mathematics (eds. Wei Chi Yang, Sung-Chi Chu, Jen-Chung Chuan) (2000), 325-336.

[2] O. Caprotti, J. Davenport, M. Dewar, J. Padget,
Mathematics on the (Semantic) NET. Lecture Notes in Computer Science **3053**, (2004). 213-224. `http://monet.nag.co.uk/monet/`.

[3] H. Chaachoua, J. Nichaud, A. Bronner, D. Bouhineau, *APLUSIX, a learning environment for algebra, actual use and benefits.* Proceeding of ICME-10: 10th International congress on Mathematical Education (2004). `http://aplusix.imag.fr`.

[4] ConneXions, `http://cnx.org`.

[5] G. Corbalan, H. Cuypers, F. Paas, *Het Leren van Lineaire Algebra: Effecten van Feedback op Motivatie en Efficiëntie van het Leren.* Tijdschrift voor Didactiek der β-wetenschappen **26 1 & 2** (2009), 21-36.

[6] CnXML, ConneXions markup language,
http://cnx.org/aboutus/technology/cnxml.

[7] A.M. Cohen, H. Cuypers, E.R. Barreiro, *MathDox: mathematical documents on the web*. OMDoc: An Open Markup Format for Mathematical Documents (ed. M. Kohlhase) (2006), 262-265.

[8] A.M. Cohen, H. Cuypers, H. Sterk, *Algebra Interactive*, Springer-Verlag, 1999.
An interactive version is under construction at
http://dam02.win.tue.nl/mathadore/ida/bookframe.html.

[9] H. Cuypers, A.M. Cohen, J.W. Knopper, R. Verrijzer, M. Spanbroek,
MathDox - A System for Interactive Mathematics. Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (EDMEDIA) (2008), 5177-5182.
MathDox, http://www.mathdox.org.
Manual, http://www.mathdox.org/player/.
MathDox Formula Editor, http://mathdox.org/formulaeditor/.

[10] I. Dahn, *Management of Informal Mathematical Knowledge - Lessons Learned from the Trial-Solution Project*. Electronic Information and Communication in Mathematics, LNCS (2003), 29-43.

[11] P. De Bra, G.J. Houben, H. Wu, *AHAM: a Dexter-based reference model for adaptive hypermedia*. Proceedings of the tenth ACM Conference on Hypertext and Hypermedia: returning to our diverse roots (1999), 147-156.

[12] P. De Bra, *Web-based educational hypermedia*. Data Mining in E-Learning (eds. C. Romero and S. Ventura). WITpress, 2006, 3-17.

[13] Digital Mathematics Envirmonment, http://www.fi.uu.nl/dwo.

[14] Emilea-Stat, http://www.emilea.de.

[15] GAP —Groups, Algorithms, Programming— a System for Computational Discrete Algebra, http://www.gap-system.org/gap.html.

[16] J. Gosling, B. Joy, G. Steele, *The Java Language Specification*. 3rd Edition, Addison-Wesley Professional, 2005. http://java.sun.com.

[17] F. Halasz, M. Schwartz, *The Dexter Hypertext Reference Model*. Communications of the ACM **37 2** (1994), 30-39.
http://java.sun.com/products/servlet/.

[18] Jelly, http://commons.apache.org/jelly/.

[19] Knot Theory, http://dam02.win.tue.nl/mathadore/knots/.

[20] M. Kohlhase (ed.), *OMDoc: An Open Markup Format for Mathematical Documents*. Springer-Verlag, 2006.

[21] M. Kohlhase, C. Müller, *Semantic technologies for Mathematical elearning*, preprint, JEM workshop 6 (2009).

[22] M. Kohlhase, C. Müller, *Panta Rhei*. Wissens- und Erfahrungsmanagement LWA (Lernen, Wissensentdeckung und Adaptivität) conference proceedings (Alexander Hinneburg, ed.), (2007), 318-323.

[23] LeActiveMath, http://www.leactivemath.org.

[24] Maple, http://www.maplesoft.com.

[25] MapleTA, http://www.maplesoft.com.

[26] Mathadore, http://www.mathadore.nl.

[27] Mathematica, Wolfram Research, Inc., *Mathematica Edition: Version 7.0*, Wolfram Research (2008). http://www.wolfram.com.

[28] Mathematical Markup Language (MathML) Version 2.0, http://www.w3.org/TR/MathML2/.

[29] Mathematical Markup Language (MathML) Version 3.0 (proposal), http://www.w3.org/TR/MathML3/.

[30] E. Melis, J. Siekmann, *Activemath: An intelligent tutoring system for mathematics*. Proceedings of the International Conference on Artificial Intelligence and Soft Computing, 91-101. http://www.activemath.org.

[31] OpenMath, http://www.openmath.org.

[32] Orbeon, http://www.orbeon.com.

[33] SAGE, http://www.sagemath.org.

[34] SCORM, http://www.adlnet.gov/Technologies/scorm/default.aspx.

[35] C. J. Sangwin, M. J. Grove, *STACK: addressing the needs of the neglected learners*. Proceedings of the First WebALT Conference and Exhibition (2006), 81-95.

[36] N. Walsh, L. Muellner, *DocBook: The Definitive Guide*. 1st Edition, O'Reilly, 1999. http://www.oasis-open.org/docbook/.

[37] Wortel TU/e, http://wortel.tue.nl.

[38] XForms, http://www.w3.org/MarkUp/Forms/.

[39] XInclude, http://www.w3.org/TR/xinclude/.

[40] XPath, http://www.w3.org/TR/xpath/.

[41] XQuery, http://www.w3.org/TR/xquery/.

A.M. Cohen
Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513
5600 MB, Eindhoven
The Netherlands
e-mail: amc@win.tue.nl

H. Cuypers
Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513
5600 MB, Eindhoven
The Netherlands
e-mail: hansc@win.tue.nl

R. Verrijzer
Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513
5600 MB, Eindhoven
The Netherlands
e-mail: r.verrijzer@tue.nl