

Adaptation of Mathematical Documents

Christine Müller

A thesis submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy in Computer Science

Date of Defense: 5 May 2010
Date of Publication: 10 May 2010

Jacobs University Bremen
School of Engineering and Science

Dissertation Committee

Prof. Michael Kohlhase, Jacobs University Bremen, Germany
Prof. Adalbert Wilhelm, Jacobs University Bremen, Germany
Prof. Cristian Calude, University of Auckland, New Zealand

To my parents.

I hereby declare that this thesis has been written independently except where sources and collaborations are acknowledged and has not been submitted at another university for the conferral of a degree.

Parts of this thesis are based on or closely related to previously published material or material that is prepared for publication at the time of this writing. These are [MK09a, CM09, KLM⁺09, MK09b, Mül09a, MK08a, MK08b, Mül08b, Mül08a, MK08c, KMR08, WM07, MK07, KMM07a, KMM07b]. Parts of this work are the result of collaborations with other researchers: for Part II with Michael Kohlhase, Christoph Lange, Normen Müller, and Florian Rabe and for Part III with Michael Kohlhase, Achim Mahnke, and Normen Müller. In each case the precise connection to this work is detailed in the relevant passages of the text.

Bremen, 10 May 2010, Christine Müller

Abstract

Modern web technologies have empowered users to create and share documents across the world. Today, users are confronted with an immense amount of documents, including documents in a more traditional understanding such as publications, manuals, and textbooks as well as documents in a wider interpretation such as forum postings, ratings, and tags. Confronted with this immense amount of information, users struggle to find appropriate documents and to acquire the essential knowledge conveyed in these documents. Essentially, the web's usability depends on whether or not it can respond to the individual user preferences and needs (called the user context) and personalise web documents respectively.

Personalising documents is widely addressed by applications in industry and research. For example, eLearning systems address the personalisation of online documents, particularly, the presentation and content planning of documents. They propose the most advanced adaptation services and are thoroughly discussed in this thesis. Unfortunately, all of these systems apply a topic-oriented approach: They can only handle self-contained information units (called topics or learning objects) that omit transitional phrases and cross-references. The resulting topic-oriented documents lack coherence and guidance that traditional, narrative writings provide. However, these narrative documents can not be easily modularised into reusable document parts that can be arbitrarily combined and arranged in a document.

This work applies the topic-oriented principles of modularisation and reuse to the narrative authoring paradigm. Narrative documents are modularised into infoms, for which transitional phrases and cross-references are marked. Infoms and their semantic/narrative dependencies as well as variant relations are modelled as graphs. These graphs are processed during the content planning of narrative documents during which appropriate infoms are selected and arranged according to the user's context. Since the narrative transitions are visualised by words and phrases, they can reduce the adaptability of infoms. To improve the exchangeability of infoms, infoms are thus enriched with alternative transitions and cross-references. With these enrichments, appropriate narrative transitions can be selected according to the combination and arrangement of document parts into user-specific documents.

To illustrate the proposed adaptation services for narrative documents, mathematical documents are used. This decision required the author to take an essential aspect of mathematical text into account: Mathematics is a mixture of natural language text, symbols, and formulae. Symbols and formulae can be presented with different notations. These notations can complicate communication and acquisition processes since notations are context-dependent and can considerably vary among different communities and individuals. These variations can cause ambiguities and misunderstandings.

In this situation, the author proposes a comprehensive framework that allows users to configure notations as well as to guide the content planning of mathematical documents regarding a semantic, narrative, and user context. To prioritise these contexts and to guide the adaptation, a combination of extensional and intensional options is provided. These give users full control over any step of the adaptation workflow: The specification of which document parts should be adapted and which should remain unchanged, the collection of adaptation objects (notation preferences, infoms, and context parameters), and the user-specific selection of the most appropriate objects to be applied in the rendering, substitution, and ordering of document parts. Since mathematics is the foundation of many other disciplines, the author expects that the findings of her work can be applied to other domains and a wide range of documents.

I am most grateful to my supervisor, Michael Kohlhase, for his continued encouragement and invaluable suggestions during this work. I am especially indebted to Cristian Calude whose insights and advice have been very compelling and fruitful. I also thank Adalbert Wilhelm for his support and feedback.

I would also like to include my gratitude to my colleagues that have provided the environment for sharing their experiences and theories about the problem issues involved as well as participated in stimulating team exercises to discuss and develop solutions to the identified problems. I want to particularly like to thank my colleagues and co-authors Fulya Horozal, Constantin Jucovschi, Andrea Kohlhase, Christoph Lange, Immanuel Normann, Achim Mahnke, Normen Müller, Florian Rabe, Heinrich Stamerjohanns, Marc Wagner, and Vyacheslav Zholudev. This is also extended to the students of the KWARC group who contributed to the development of my prototype system. These are Andrei Aiordachioaie, Stefania Dumbrava, Josip Dzlonga, Darko Makreshanski, Dimitar Misev, Alen Stojanov, and Jakob Ücker.

Finally, I would like to thank Jacobs University, the Germany National Merit Foundations, and the JEM-Thematic-Network ECP-038208 who have provided me with financial independence for the past three years.

Contents

I. Introduction & State of the Art	1
1. Introduction	3
1.1. The Document-Centered and Topic-Oriented Paradigms	6
1.2. Research Contribution	7
1.3. Adaptation of Mathematical Documents	9
1.4. Context-based Adaptation	12
2. Preliminaries & State of the Art	17
2.1. XML-based Markup Formats	17
2.2. Systems for Document Adaptation	31
II. Adaptation of Mathematical Notations	45
3. Introduction	47
3.1. Introduction into Mathematical Notations	48
3.2. Representation of Mathematical Notations	51
3.3. State of the Art	53
3.4. Requirements Specification	55
4. Pattern-based Rendering of Notations	59
4.1. Information Model	60
4.2. Specification of Notation Definitions	64
4.3. The Rendering Algorithm	66
4.4. Guiding the Rendering Algorithm	69
5. Summary & Evaluation of Notations	79
5.1. Services & Limitations	79
5.2. Proof of Concept	83
5.3. Chapter Summary	84
III. Content Planning for Mathematical Documents	85
6. Introduction	87
6.1. Representation of Variants/Alternatives	87
6.2. State of the Art	92
6.3. Requirements Specification	94
7. The Content Planning Approach	99
7.1. Modularisation of Mathematical Documents	100

Contents

7.2. Variants & Variant Relations	108
7.3. Specification for the Content Planning	114
8. Substitution of Document Parts	119
8.1. Information Model	119
8.2. The Abstractor	122
8.3. The Substitution Algorithm	124
8.4. Summary & Evaluation of the Substitution	132
9. Reordering of Mathematical Documents	139
9.1. Information Model	139
9.2. The Abstractor	141
9.3. The Reordering Algorithm	142
9.4. Summary & Evaluation of the Reordering	153
IV. Adaptation in Practice	159
10. The Panta Rhei System	161
10.1. The Adaptable Panta Rhei	163
10.2. The Social Panta Rhei	175
10.3. The Adaptive Panta Rhei	183
10.4. Chapter Summary	194
11. Conclusion & Future Work	195
11.1. Conclusion	195
11.2. Future Work	196
11.3. Epilog	198

Part I.

Introduction & State of the Art

1. Introduction

Information technologies have transformed our present era into an information age, in which individuals can freely transfer information and have instant access to data that was formerly difficult or impossible to access. In particular, the rise of the **World Wide Web** (WWW, W3 or Web) has led to an explosion of digital materials that are highly interlinked and available at our fingertips. Originally developed by Tim Berners-Lee in 1989 [BL90] to provide an easy way to exchange research results, the WWW evolved into a global source of information for everyone: *The WWW is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents* (Tim Berners-Lee).

Summarised under **Web 2.0** [O'R05], special technologies for the WWW changed the rule of communication: Web tools are no longer restricted to few highly-skilled (power) users in creating and distributing content. Instead, they involve a large number of (average web) users to collaboratively provide information using software tools such as wikis, blogs, social bookmarking and tagging tools, forums and social network applications like TWITTER [twi], FACEBOOK [Fac], YOUTUBE [You], etc.

In consequence to the rise of modern web technologies, computers and the Internet have penetrated every aspect of our lives. People around the globe are connected to one another and have access to huge amounts of data, including videos, pictures, and music but also documents of various kinds, such as news, advertisements, product descriptions, articles in encyclopaedias, tutorials, user manuals, learning materials, software documentations, and scientific publications. Confronted with this immense amount of information, the WWW faces a dilemma: Internet users can easily create and exchange information but struggle to find the appropriate resources as well as to acquire the essential knowledge conveyed in these resources.

Essentially, the usability of the web depends on whether or not it can respond to the individual user with his specific expectations, preferences, skills, or environmental constraints. These individual characteristics are henceforth summarised as the user's context. The **user context** defines which content is most appropriate for a user and how it needs to be presented to be most easily absorbed by him.

Personalising documents according to a user context is widely addressed by applications in industry and research. Recommender systems, usually using some form of collaborative or hybrid filtering, have been used for well over a decade. Major eCommerce players such as AMAZON [LSY03], NETFLIX [Neta], and LAST.FM [las] pay a lot of attention to personalisation. NETFLIX, for example, awarded a one million dollar prize to a team, which was able to improve the accuracy of the system's recommendations by more than 10% [Netb]. Researchers in the field of adaptive hypermedia study how systems can change the human-computer interaction from a one-size-fits-all attitude to a personal, more humanised experience. Respective services include navigation support, ranking of search results, adaptive presentation of resources, and the content planning of documents.

Favourite demonstrators for such hypermedia systems are adaptive, web-based **eLearning systems**, which aim at the personalisation of educational resources (lecture notes, assignments, text books, etc) to the preferences and competencies of individual learners and their changing levels of understanding. They not only support learners in finding appropriate information but also in acquiring the conveyed knowledge, eventually improving the efficiency

1. Introduction

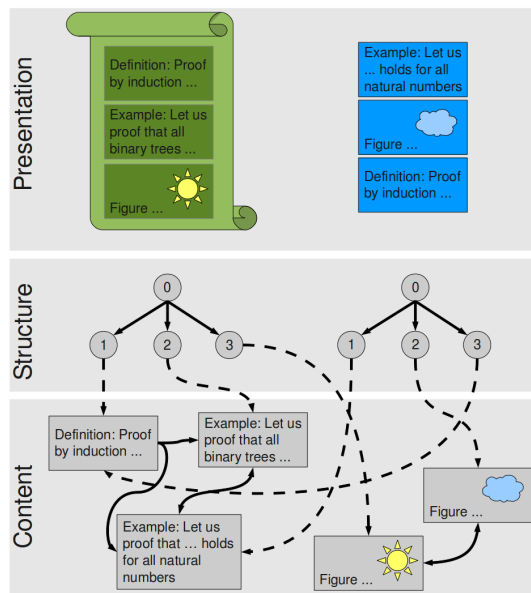
and effectiveness of learning. Respective services include the *content planning* of learning resources as well as the *presentation* of learning resources. While the former selects and arranges resources, the latter generates a user-specific version from one resource, e.g., taking layout preferences such as colour or fonts into account.

Many approaches implemented by eLearning systems struggle to personalise documents while assuring coherent outputs, in which the meaning and sequence of ideas relate to each other. **Coherence** is the product of many different factors, which are combined to make every paragraph, every sentence, and every phrase contribute to the meaning of the whole document. One important aspect is the **cohesion**, which defines the flow and connectedness of the document. For example, cohesive devices include *transitional words and phrases* (however, in addition, for instance, as we have seen above, etc), which help to sequence a text and to clarify the relationship among ideas and arguments, as well as *cross-references* (Figure ..., Section ..., ect), which refer to previous or subsequent ideas in the document.

This work proposes an advanced approach for the content planning and presentation of web documents, which assures the coherence of documents by preserving their connectedness via transitions and cross-references. Henceforth, **web documents** are defined as documents that can be converted into one of the two WWW formats, the Hypertext Markup Language (HTML [Gro99]) or the eXtensible Hypertext Markup Language (XHTML [Gro02]). The two services (content planning and presentation) are henceforth referred to as **adaptation services**.

In order to adapt web documents, we have to understand and handle their constituents. According to the W3C, a web document has three layers: the document content, structure, and its presentation.

The content of a document refers to what it says to the user through natural language, images, sounds, movies, animations, etc. The structure of a document is how it is organized logically (e.g., by chapter, with an introduction and table of contents, etc.). The presentation of a document is how the document is rendered (e.g., as print, as a two-dimensional graphical presentation, as a text-only presentation, as synthesized speech, as Braille, etc.) [CVJ99].



Consider the figure to the left. The content, structure, and presentation layer of two documents is illustrated. The content layer includes several document parts: a definition, two examples, and two figures. The structure layer presents two document structures that combine the document parts in different orders. The presentation layer displays two renderings of these structures, as manuscript (to the left) and as slides (to the right).

Adaptation of documents can take place on any of the three layers. The content planning approach as proposed by this work implements services on the content and structure layer. *Services on the content layer* address the substitution of document parts with alternative ones. They support the per-

sonalisation of documents to different user backgrounds. In our example, the manuscript includes an example on binary trees and the slide presentation an example on natural numbers. The bidirectional arrows between the two examples denotes that they are equivalent. They both illustrate a proof by induction but focus on different audiences. A computer scientist might prefer the example on binary trees and a mathematician the example on natural numbers. *Services on the structure layer* support the restructuring of documents and the reordering of document parts. They support the adaptation of documents to different learning styles: While one learner might prefer visual content and concrete, practical examples, another learner might like verbal content, abstract concepts, and generalisations. In our example, the manuscript starts with the definition of a proof by induction, while the slide presentation first includes an example and an illustration to introduce the topic. *Services on the presentation layer* adapt the appearance of documents without changing content or structure. For example, a print-out and a digital version of a document can be rendered differently.

The most important prerequisite for adaptation services is a representation of documents, which separates the three document layers (content, structure, and presentation) so that they are comprehensible to a computer system. Presentational information has to be distinguished from the structure and content of documents. The arrangement of document parts has to be marked and the content has to be unlocked and structured. Such machine-processable document representations are developed in the scope of the **Semantic Web**. The main objective of the Semantic Web is to explicate the meaning of web resources so that they can be understood by a machine. For this, the Semantic Web postulates *annotations* (or *metadata*) to classify and describe web resources. Several initiatives aim at standardising metadata: the IEEE Learning Object Metadata [WG102], Sharable Content Object Reference Model (SCORM [Lea]), the general purpose scheme Dublin Core [Dub], the instructional ontology by ACTIVEMATH (Section 2.2.3), the system ontology by [KBLL⁺04], the rhetorical ontology by [KMRW07], OMDOC's ontology for mathematical knowledge (Section 2.1.5), and friend-of-a-friend (FOAF [FOA]).

Metadata can be embedded in the document by extending the underlying document format with respective attributes and elements. Such markup is called **inline markup**. For example, the ACTIVEMATH group has extended OMDOC with the instructional ontology. Alternatively, an integrative approach, such as implemented by OMDOC's new metadata syntax [LK09], can be applied to embed a variety of metadata schemes as RDFa [RDF08] with only little modifications of the document format.

Relations and properties of document concepts can also be provided in separation to the document. Such markup is called **stand-off markup** (or remote markup). Tags (Section 4.4.1), Cascading Style Sheets [BLLJ08] but also rule systems (Section 7.2.2) are stand-off markup options.

Document formats that embed annotations are often instances of the eXtensible Markup Language (XML [BPSM⁺08]) and are referred to as **markup languages**. From now on we assume that all documents are representable in an XML-based markup language. The annotations of markup languages explicitly distinguish (and accordingly 'mark up' within a document) the structure, properties, and relations of text paragraphs. They can be computed by adaptation engines to personalise the respective document. The quality of the adapted document depends on the richness of the underlying representation format: The more annotations are provided and the higher the quality of the format, the better the processing of documents and, thus, the better the adaptation result. Since markup languages are an essential prerequisite for the adaptation of documents, a number of markup languages is analysed in Section 2.1. These markup languages are characterised based on two paradigms, the *document-centered* paradigm and *topic-oriented* paradigm.

1.1. The Document-Centered and Topic-Oriented Paradigms

The **document-centered paradigm** has been used for a long time. It builds on ‘English 101’ principles: Respective documents usually include an introduction, a successive exploration of new ideas, reviews, and references [Wal09d]. The information units of the writing are usually supported by previous units. Often transitional phrases and cross-references are included.

The document-centered approach follows a *top-down* authoring style, e.g., from a document, to chapters, to sections, to subsections, to visual document parts like examples or definitions, and, finally, to paragraphs. In analogy to the terminology introduced by [Mül06], all of these document components are called **narrative information units** (short **infoms**) and the assembled document is called **narrative document**. An infom can be nested and include smaller infoms. It is not *self-contained* since it can include cross-references and transitions to other infoms. Without these infoms it can not be understood and the information contained in the infom can not be disambiguated. The cross-references and transitions between the infoms guide the reader and produce a coherent flow through the document, reflecting the narrative practice of the author.

Although transitions and cross-references improve the coherence of documents, they also reduce their modularity and thus hamper the reusability of document parts and increase costs due to redundancies. This is particularly critical for the content planning of documents, during which documents have to be modularised so that their parts can be easily substituted with alternative ones as well as arbitrarily arranged.

The **topic-oriented paradigm** is based on the principles of reuse and modularisation and was originally developed for managing technical writings, such as product descriptions, user manuals, or software documentations. For these documents a flexible management infrastructure was needed that reduces authoring costs as well as change management issues, e.g., due to redundancies and hard-to-trace inconsistencies caused by changing some but not all copies of a document paragraph.

The topic-oriented approach follows a *bottom-up* authoring style (sometimes called aggregation), e.g., from paragraphs, to subsections, to sections, to blocks, to topics, and further to topic groupings, chapters, and, finally, the document. The assembled document is called **topic-oriented document**. **Topics** are defined as **self-contained** — i.e., isolated, stand-alone, and context-independent — information units. They are usually assembled into documents by a **transclusion mechanism**: Rather than copying information units into a document, they are reused (from one central source) by placing content references into the document. Most XML markup languages provide such references or include operators, e.g., by integrating the generic XML inclusion operator XINCLUDE [MOV06]. Some of these formats also support the specification of the document’s **tree structure**, where the leaves are content references to topics. In order to produce a complete document from such XML skeletons the referenced topics are retrieved (from other files or a database) and copied into the output document. Consequently, topics can be arbitrarily reused from a single source, thereby omitting redundancies and inconsistencies.

However, the topic-oriented approach has to impose constraints on the maintained units: They have to omit transitions and cross-references to other topics. This restriction is problematic because the assembly of documents from such independent texts can reduce the coherence of the output.

Topics give the author the flexibility to pull loosely coupled components together for different purposes. Writing in the traditional, linear fashion gives the author the ability to provide continuity for the reader. Pulling a set of very loosely

coupled topics together into something that has a fluid, coherent end-to-end narrative that satisfies the readers expectations of continuity strikes me as seriously, really, very hard [Wal09a].

[Wal09d] illustrates the problem of the topic-oriented approach with an analogy to painting. Figure 1 presents the complete, coherent painting and a picture that was assembled from squares (or topics). Each square in the assembled painting is painted in relative isolation. To save time and effort, similar topics are used from other paintings. The final work is achieved by mapping all topics into a whole.

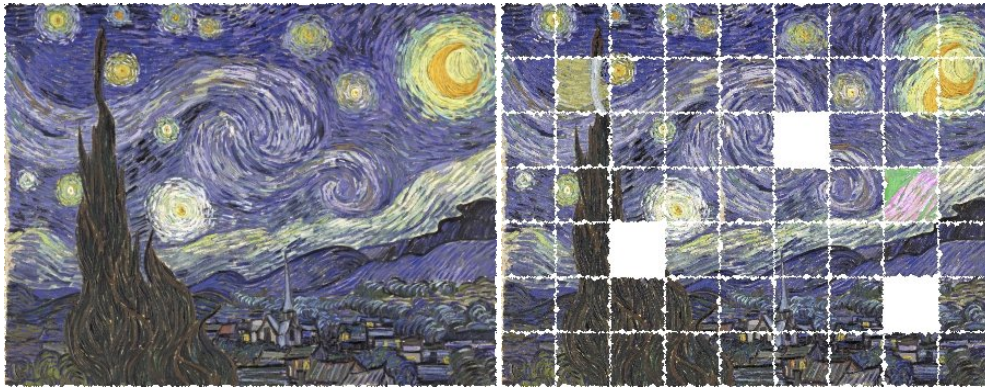


Figure 1.: Starry Night by Vincent Van Gogh [Wal09d]

Although topics are self-contained in terms of transitions and cross-references, they are not necessarily *semantically independent*. Instead, a *narrative* and *semantic self-containedness* of information units can be distinguished. Document parts are **narratively self-contained** (and thus considered as topics) if they omit **visual markers** such as narrative transitions and cross-references to other document parts. In contrast, document parts are **semantically self-contained** if they do not depend on each other. Infoms and topics are usually highly interdependent and rarely semantically self-contained.

1.2. Research Contribution

The topic-oriented and document-centered paradigm seem to be contradictory. Topics are authored independently as narratively self-contained units. Lacking transitions and cross-references they can be easily reused in various document structures: Topics can be freely substituted with equivalent topics and can be arranged arbitrarily in the document. However, the resulting documents lack *visual markers* that guide the reader through the document. In contrast, infoms include transitions and cross-references and are thus not narratively self-contained. They are authored, while keeping the whole document structure in mind, thus, providing a better coherence and guidance for the reader. However, narrative documents can not be easily modularised and are less suited for reuse. Both, topics and infoms, are usually highly interconnected and thus not semantically self-contained. These interrelations are valuable for the adaptation of documents.

For the content planning of documents it is essential that documents can be *modularised* so that their parts can be easily substituted with alternative ones as well as arbitrarily arranged, while preserving the *coherence* of the adapted documents. Neither topic-oriented nor document-centered paradigm lead to an adaptation infrastructure, which supports modularisation *and* coherence.

1. Introduction

To address this challenge, the author proposes to bridge the two paradigms and to combine aspects of both worlds. As this task is too big for one thesis, this work starts with the *adaptation of narrative documents*. It applies the topic-oriented principles of modularisation and reuse to the document-centered world. To evaluate and illustrate the proposed adaptation approach, mathematical documents are used. Since mathematics is the foundation for many other disciplines (software engineering, electrical engineering, etc), the author expects that the findings of her work can be applied to other domains and to a wide range of documents.

The adaptation of mathematical documents requires to take an additional aspect into account: Mathematics is a mixture of natural language texts, symbols, and formulae. Symbols and formulae can be presented with different notations, which underlie cultural conventions and individual preferences (Section 3.1). When planning the content and structure of documents, these notations have to be adapted respectively. In particular when applying content planning to multi-authored document collections, adaptation of notations becomes a central issues and helps to avoid notational inconsistencies.

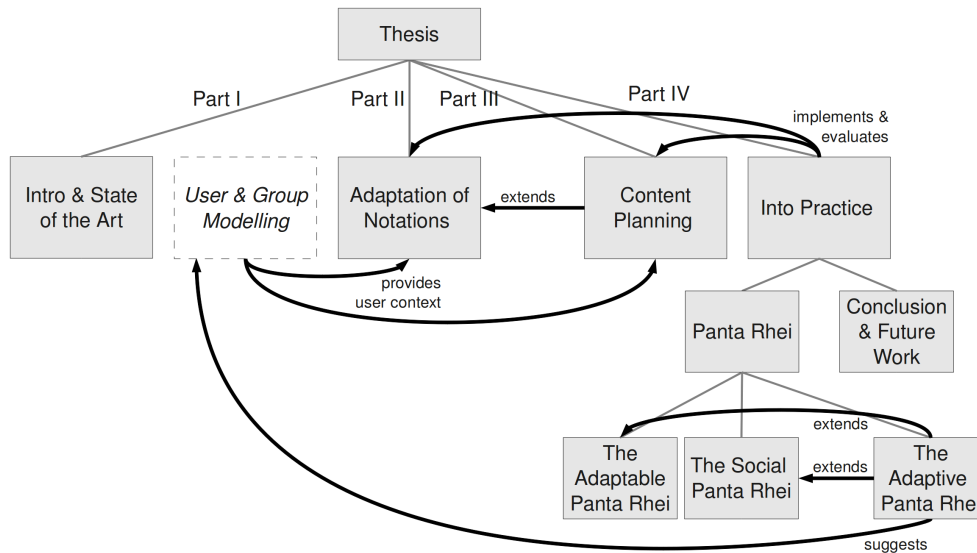


Figure 2.: Structure of this thesis

Figure 2 outlines the structure of this thesis. Part I contains the introduction and state of the art, which introduces several document formats and adaptation systems. Part II specifies the adaptation of mathematical notations. Part III extends the discussion in Part II for the content planning of mathematical texts and proposes the substitution and reordering of document parts. Part IV puts the theoretical discussions in Part II and Part III into practice. It introduces the adaptable *panta rhei* prototype (Section 10.1), which implements Part II and III and evaluates the proposed services.

Note that the adaptive hypermedia community distinguishes two adaptation principles: an adaptive and an adaptable approach [BKN07]. The former automatizes the adaptation by making assumptions on the user's preferences based on his interactions or his answers in questionnaires. This approach is also called **user modelling**. The inferred information on the user is stored in his user model from which parameters are derived that guide the adaptation. This work suggests an adaptable approach. Instead of memorizing user inputs or even inferring user characteristics implicitly, users are expected to enter their constraints explicitly for each adaptation request. However, user modelling techniques, in particular, collaborative and

social modelling approaches considerably reduce the effort for users and should be at least considered. A respective extension of this work can transform the herein described *adaptable, user-driven* approach towards an *adaptive, system-driven* one. As illustrated in Figure 2, the author would place the theoretical discussion on elaborated user modelling techniques before Part II.

Part IV presents the social *panta rhei* prototype (Section 10.2), which focusses on the discussion and rating of semantically marked-up documents. It supports user and community ratings and uses the latter to automatically rank search results. The social *panta rhei* has encourage the author to look into the extension of *panta rhei* towards an adaptive system, the adaptive *panta rhei*. Though not completely worked out, Section 10.3 presents the author's initial ideas on the modelling of user and group preferences. Section 11 concludes this thesis and summarises further research issues.

1.3. Adaptation of Mathematical Documents

The characteristics of mathematical knowledge make mathematical documents an ideal test tube. Mathematical knowledge is *precise, highly-structured*, and *extraordinarily interlinked*. The structure and interdependencies of mathematical information units can be modelled (using markup languages) and provide a rich machine-processable representation of mathematical documents. Mathematical document formats are particularly designed to explicate the **semantic context** of mathematical documents: They enable the representation of the implicit inheritance structure of mathematical knowledge, the classification of text paragraphs like proofs, definitions, examples, etc, the marking of mathematical relations between document parts, and the explication of the meaning of mathematical symbols and formulae as well as their notations. An analysis of explicit representations of mathematical information units and their interrelations reveals that they are usually not semantically self-contained: Axioms are the only independent units in mathematics, all other mathematical constructs depend at least on these axioms. Consequently, document parts of mathematical texts are seldom independent but rather highly interconnected.

Some document formats, such as OMDOC (Section 2.1.5), support a modularisation of mathematical documents that takes these semantic interrelations into account. They place document parts into larger structures that provide them with a *mathematical context*. These structures are referred to as **theories** and are linked via *theory morphisms*. This mechanism reifies a practice that has long been relatively overt in mathematical documents, e.g., the Bourbaki development of mathematics that starts with set theory [Bou68] and takes the mathematical practice of stating results with minimal preconditions to the extreme. The markup for theory objects extends the theoretically motivated accounts of inheritance and modularity in programming languages and mathematics to cover informal (but rigorous) mathematical practice. Intuitively, a **theory morphism** is a mapping between theories that allows to 'view' the source theory in terms of the target theory, if the mapping conserves truth. In the simplest case, theory morphisms model inheritance — the source theory can be viewed as an included part of the target theory — and thus allow to model the mathematical practice of modular/object-oriented development of knowledge in mathematics.

Figure 3 illustrates the theory graph of an introductory computer science course, where nodes correspond to theories and edges denote theory morphisms. This modularisation of documents can not only be applied to theories but rather to any addressable document part. Using XML technologies like XPATH [CD99] and XPOINTER [GMMW03], any specifically marked aspect of a document (including chapters, sections, paragraphs, sentences, single words as well as properties and relations between these) can be referenced and extracted.

1. Introduction

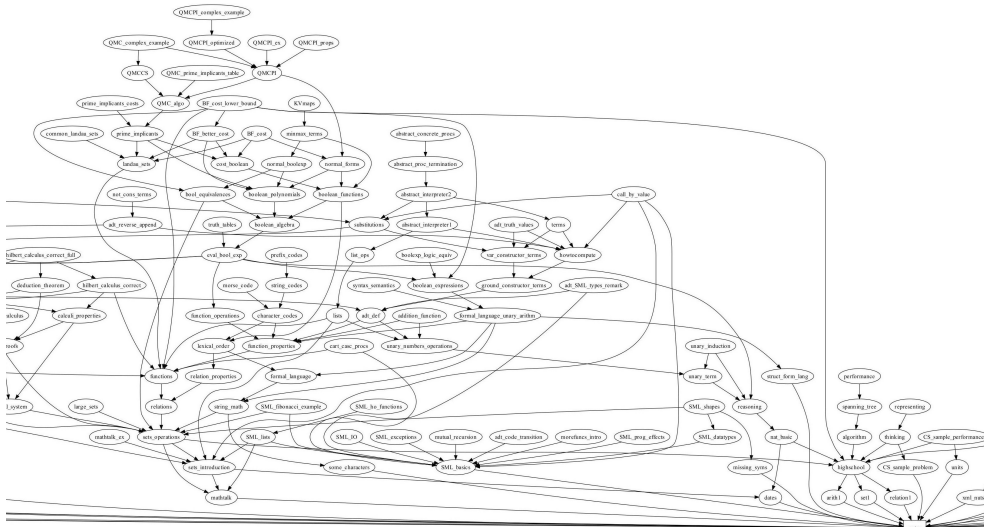


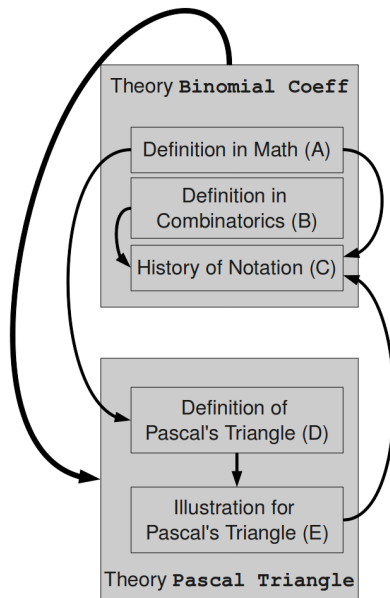
Figure 3.: The inheritance graph of an introductory computer science course [Koh08b]

Relations between document parts often reflect *mathematical dependencies*. Given a semantically rich document format, these underlying dependencies between document parts can be inferred (Section 7.1). XML technologies thus support the modelling of **dependency graphs**, where nodes correspond to addressable document parts and edges denote their dependencies.

A **dependency** associates a dependent document part (where the edge is *incoming*) with a supporting document part (where the edge is *outgoing*). The **supporter** is required in order to disambiguate and understand the information conveyed in the **dependant**. Having introduced dependency graphs, we can refine the definition of self-containedness: Document parts are **self-contained** if they are independent of any document part, i.e., if they have no incoming edges in a dependency graph.

The figure to the left illustrates a modularised mathematical document. The arrows denote dependencies between document parts. The boxes *A, B, C, D*, and *E* correspond to definitions and examples. The theories provide these document parts with a mathematical context and explicate the inheritance structure of the conveyed mathematical knowledge. The Pascal Triangle theory groups example *E* and definition *D*, while theory Binomial Coeff subsumes definitions *A* and *B* as well as example *C*.

The document parts are highly interdependent and form a dependency graph. For instance, the example *E* depends on the definition *D*. *E* is the dependant (it has an incoming edge from *D*) and *D* is the supporter (it has an outgoing edge to *E*). Since *D* depends on *A* it becomes the dependant in this relation. The definition *A* is independent in any relation (it has no incoming edges) and is thus semantically self-contained. The terminology of dependants and supporters can also be applied to mathematical theories, e.g., Pascal Triangle is the dependant and Binomial Coeff the supporter.



1.3. Adaptation of Mathematical Documents

We have seen that mathematical formats contribute an essential step towards the adaptation of mathematical documents: they provide a rich representation of documents as well as an approach to modularise them. Based on an explicitly marked-up semantic context, adaptation engines can take dependencies between document parts into account. Accordingly, document parts, which do not act as supporter, can be omitted since they do not foster the understanding of other document parts (e.g., *C* or theory ‘Pascal Triangle’). The removal of supporters is problematic since they are required by other information units (e.g., *A*, *B*, *D*, *E* or theory ‘Binomial Coeff.’). Dependencies also provide a notion of relevance and connectedness and can help to select appropriate content for a document: Dependants should not be added to a document if the required information units are not yet part of the document. The insertion of self-contained document parts should be handled with care as these document parts might not be relevant to the document’s content. Instead, document parts that depend on other document parts in the document can be preferred. The semantic context also allows adaptation engines to arrange document parts according to their mathematical dependencies, e.g., supporters can be placed before their dependants to assure that required information is acquired first.

Mathematical formats focus on mathematical aspects and do not yet consider the narrative flow of texts. They omit a marking of transitional words and phrases. The coherence of adaptation results is at risk, if such transitions do not correspond to the underlying mathematical dependencies. For example, a transitional phrase “*see Pascal’s triangle below*” in part *C* induces that *C* should be placed before *E* (Figure 4). This transition is not reflected by the underlying dependencies between *C* and *E* and can not be considered by an adaptation engine. An adaptation engine that prioritises all supporters and appends all dependants afterwards, will produce the sequence *A,B,D,E*, and *C* and reorder the parts respectively. The output document is not coherent. Figure 4 presents a more appropriate flow through the document parts, which considers the transition from *C* to *E* and places *C* before *E*.

<i>A</i>	In mathematics, the binomial coefficient $\binom{n}{k}$ is the coefficient of the x^k term in the polynomial expansion of the binomial power $(1+x)^n$.
<i>B</i>	In combinatorics, the binomial coefficient $\binom{n}{k}$ is the number of ways of choosing k objects from a collection of n distinct objects without regard to the order.
<i>C</i>	The notation $\binom{n}{k}$ was introduced by Andreas von Ettingshausen in 1826, although the numbers were already known centuries before that (<i>see Pascal’s triangle below</i>).
<i>D</i>	Pascal’s rule is the important recurrence relation $\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$, which can be used to prove by mathematical induction that $\binom{n}{k}$ is a natural number for all n and k , i.e., if $(x+y)^n = \sum_{k=0}^n \binom{n}{k} x^{n-k} y^k$ then $\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$.
<i>E</i>	Pascal’s triangle is a geometric arrangement of the binomial coeff. in a triangle: $\begin{array}{ccccccc} & & & & 1 & & & & \\ & & & & 1 & & 1 & & \\ & & & & 1 & & 2 & & 1 & \\ & & & & 1 & & 3 & & 3 & & 1 & \\ & & & & 1 & & 4 & & 6 & & 4 & & 1 & \\ & & & & 1 & & 5 & & 10 & & 10 & & 5 & & 1 & \end{array}$

Figure 4.: Coherent arrangement of the document parts A,B,C,D and E

This work distinguishes between a semantic context of document parts and a narrative context of documents. The **narrative context** of the document is formed when combining and arranging document parts into the document. The dependencies marked by the semantic and narrative context are henceforth called **semantic dependencies** and **narrative dependencies**, respectively. Examples for semantic dependencies are theory morphisms or mathematical dependencies between document parts. Narrative dependencies are triggered by *visual markers*. They improve the coherence of a document and guide the reader through the document.

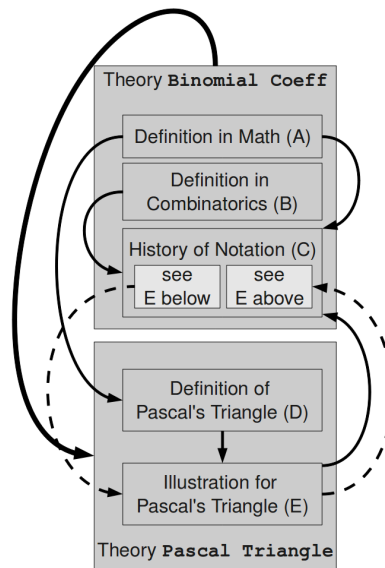
1. Introduction

The more visual markers are included in a document, the better the coherence of the document. Unfortunately, visual markers also reduce the reusability of document parts and reduce variations for the user-specific adaptation. Inserting a document part with cross-references and transitions into a document (at an inappropriate position) can cause inconsistencies.

In order to be considered during the adaptation, visual markers have to be represented in a machine-processable form. Given such markup, we can separate visual markers from the reusable content of an information unit. The reusable part and the marked up transitions and cross-references can also be enriched with alternative visual markers, allowing adaptation engines to select between them and to guide readers through various arrangements of document parts. Narrative dependencies between document parts thus become dynamic and can be selected according to the combination and arrangement of document parts into user-specific documents: The narrative context is dynamised and no longer restricts the content planning.

The figure to the right illustrates the refactoring of the above example. The dashed arrows denote narrative dependencies and the solid arrows semantic ones. The transition in part *C* is marked. Another option of the narrative context is provided that allows to place *C* after part *E*. An adaptation engine can now produce two document versions with different orderings of the paragraphs: The narrative dependency from *C* to *E* allows to create the document *A,B,C,D,E*; while the one from *E* to *C* leads to *A,B,D,E,C*.

When combining and arranging document parts, narrative dependencies are extremely critical since they are visualised for the user. In contrast, readers are not necessarily disturbed if semantic dependencies are neglected during the substitution and arrangement of document parts. Ignoring semantic dependencies results in more flexibility during the adaptation — an appropriate solution for some adaptation cases. However, in other scenarios users might wish to consider semantic dependencies as they, e.g., assure mathematical structured arrangements. Adaptation engine should thus distinguish between semantic and narrative dependencies.



1.4. Context-based Adaptation

The proposed adaptation services are implemented by combining the semantic context of document parts, the narrative context of the document, and the user context¹. These contexts can be prioritised by users and allow them to overwrite semantic or narrative constraints. After recapitulating each context, this section explains how a varying prioritization of these contexts can influence the adaptation output. Since mathematical notations play an essential role during the adaptation of mathematical texts, they are used as illustrative example.

The **semantic context** represents the mathematical inheritance structure, dependencies, and properties of document parts. It supports the adaptation of notations according to the inheritance structure of mathematical knowledge: The notations of a document part (a dependant) are thus inherited from a required part (its supporter).

¹This understanding has been motivated by the thorough analysis of context-based semantic services and their contextual requirements in [KK08].

The **narrative context** defines the arrangement and presentation of document parts in the context of the whole document. It relates to a famous citation of Aristotle “*The whole is more than the sum of its parts*”. Accordingly, the context of the document does not only reflect the semantic contexts of its parts but may extend (or overwrite) these contexts. This is particularly important to assure a coherent flow and a consistent, fine-grained use of notations in the document. For example, authors might wish to make comparisons like “*The imaginary unit is denoted by i , in physics j is used*”, which should not be overwritten by inherited notations.

The **user context** specifies individual constraints such as a specific language, learning style, or level of detail. It allows users to overwrite the semantic and narrative context. For example, it supports users to define the adaptation of their notations according to specific notation conventions or individual preferences. Depending on the technologies used, a user context can be applied *globally* to all notations or *fine-grained* to only some notations in the document. The latter assures the adaptation of notations, which is as consistent and fine-grained as based on the narrative context.

In the following, the binomial coefficient symbol and its notations are used to illustrate different adaptation results. We assume that these are generated by an adaptation engine, which allows users to vary the prioritisation of contexts. The notation $\binom{n}{k}$ in the following paragraph was inherited according to the mathematical knowledge structure of a document. The fine-grained specifications of the narrative context have been ignored. The French notation is thus not displayed with \mathcal{C}_n^k but the inherited notation $\binom{n}{k}$. The notations in the paragraph are inconsistent. A globally specified user context would have caused similar problems.

The binomial coefficient $\binom{n}{k}$ is the number of ways of choosing k objects from a collection of n distinct objects without regard to the order. Alternative notations include the French notation $\binom{n}{k}$.

To prevent such inconsistencies, users can change the priority of the narrative context. For example, they can enforce that all notations are overwritten with inherited notations unless they are explicitly protected by the narrative context. In the below paragraph, the French notation is displayed appropriately as defined by the narrative context.

The binomial coefficient $\binom{n}{k}$ is the number of ways of choosing k objects from a collection of n distinct objects without regard to the order. Alternative notations include the French notation \mathcal{C}_n^k .

Depending on the user’s preferred notation choice, the notation comparison in the paragraph can become obsolete. For example, a user’s notation choice might match with the fine-grained specification of the narrative context. In the below example, the user’s preferences is the notation \mathcal{C}_n^k and, thus, makes the comparison with the French version obsolete.

The binomial coefficient \mathcal{C}_n^k is the number of ways of choosing k objects from a collection of n distinct objects without regard to the order. Alternative notations include the French notation \mathcal{C}_n^k .

To avoid confusions, the user can be presented with a different sentence, which compares his preferred notation with other nationalities, e.g., the English notation $\binom{n}{k}$.

The binomial coefficient \mathcal{C}_n^k is the number of ways of choosing k objects from a collection of n distinct objects without regard to the order. Alternative notations include the English notation $\binom{n}{k}$.

1. Introduction

As mentioned before, this work distinguishes between two adaptation services: the presentation and content planning of documents. Content planning of documents subsumes the substitution of document parts with the most appropriate alternative and the arrangement of these parts in the document. Presentation services are supported by generating user-specific versions of a document without changing its content or structure. The adaptation of notation is considered as presentation service. It is supported by generating a user-specific notation for each mathematical symbol and formula in a paragraph. However, the adaptation of notations can require content planning. For example, the last paragraph above presents the user with an alternative comparison of notations, a reference to the English notation instead of the French one is made. Based on XML technologies, such modifications can not be generated on the presentation layer of documents. Instead, the respective, alternative paragraph has to be selected. Subsequently, its notations are adapted according to the user's preferred choice.

1.4.1. Presentation Services for Mathematical Documents

Markup languages build the foundation for presentation services for topic-oriented and narrative documents: They provide a separation between the *actual knowledge conveyed* in the documents and the *way it is presented* to the user. Annotations in **content-oriented** formats mark the meaning of the document content. The content-oriented representation can be converted into **presentation-oriented** formats, such as the eXtensible Hypertext Markup Language (XHTML). Presentational information (like user-specific layout, colours, fonts, etc) are specified in separate documents [BLLJ08] and initiate a web browser to adapt the presentation of the XHTML document.

The generation of user-specific notations is far more challenging. To support user-specific notations, mathematical document formats have to support a fine-grained markup of mathematical formulae, symbols, and their notations. In particular, they have to separate the meaning of mathematical expressions from their presentational characteristics and provide a conversion from the latter to the former, which adapts to the semantic, narrative, and user context. The representation of mathematical formulae and notations is commonly based on two standards, MATHML and OPENMATH (Section 3.2). MATHML includes two standards: Content-MATHML, the W3C [W3C] recommendation for the content-oriented representation, and Presentation-MATHML, the standard for high-quality presentation of mathematical formulae on the Web. The OPENMATH format is a well-known and widely used alternative for Content-MATHML. A smart conversion of notations from OPENMATH/Content-MATHML to user-specific Presentation-MATHML is one objective of this work and discussed in Part II.

1.4.2. Content Planning for Mathematical Documents

Most content planning approaches focus on topic-oriented documents. For example, all adaptation systems, which the author analysed in the scope of her Ph.D., follow the *learning object paradigm* [WG102], which is an instance of the topic-oriented approach. In the terminology of the learning object paradigm, topics are called learning objects. These are defined as small, reusable, self-contained, and context-free units. Since they omit transitions and cross-references, learning objects can be easily substituted with other learning object and can be arbitrarily arranged in the document, which remains rather challenging for narrative information units. These are strongly interrelated and can often not be understood without other infoms.

In order to apply a topic-oriented content planning approach to narrative documents, they have to be *modularised* (see (1) in Figure 5) into narratively self-contained units (or topics).

Understanding and explicating the transitions and cross-references in narrative documents, enables machines to automatically extract their reusable parts and to arrange or substitute them.

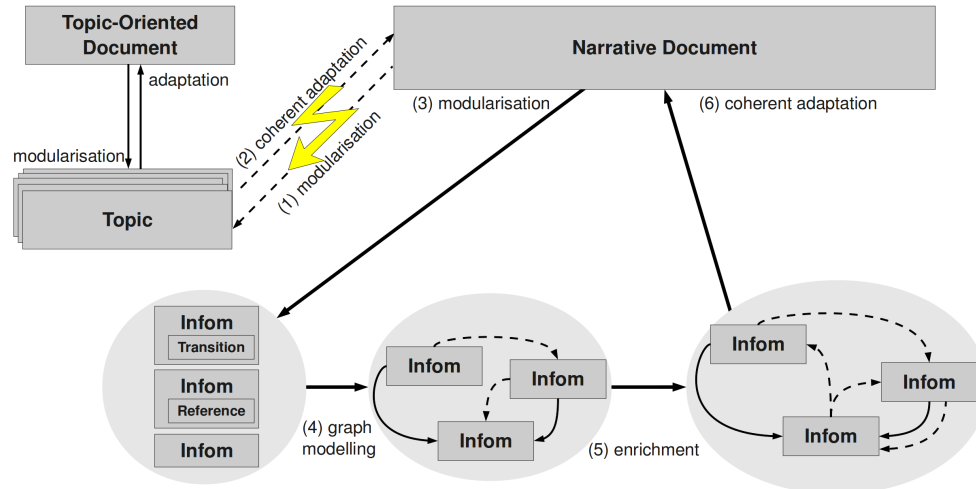


Figure 5.: Content planning for topics and infoms

Opponents of the topic-oriented style criticise such a decomposition of narrative documents as “an absurd promise to make, as most academics make liberal use of narrative style in their content, and the results of aggregation are often an incoherent mess” [Wal09d]. They underline the previously mentioned drawback of topic-oriented documents: Even if narrative documents can be successfully modularised into topics, a topic-oriented adaptation can not assure the *coherence* of the adapted document (see (2) in Figure 5).

Consequently, it is not sufficient to decompose narrative documents into narratively self-contained units. Instead, the adaptation of narrative documents requires a *new adaptation model*: Narrative documents are modularised into *infoms*, for which all transitional phrases and cross-references are marked (see (3) in Figure 5). Infoms and their interdependencies are modelled as dependency graph. Two dependency types are distinguished: semantic and narrative dependencies. These can overlap and oppose each other (see (4) in Figure 5). Narrative dependencies are *dynamised* by enriching infoms with alternative transitions and cross-references (see (5) in Figure 5). The narrative dependencies between document parts are thus no longer static but rather depend on the combination and arrangement of document parts in the user-specific document. Users can prioritise the semantic, narrative, and user context to guide the substitution and arrangement of the modularised components into narrative documents (see (6) in Figure 5). Details on the content planning of mathematical documents are discussed in Part III.

The next section analysis the state of the art of XML technologies. Since markup languages are an essential prerequisite for the adaptation of documents, several markup languages are analysed in Section 2.1. One format is selected and extended. Section 2.2 presents the state of the art of adaptation systems for mathematics and outlines weaknesses of these systems that can be improved by this work.

1. Introduction

2. Preliminaries & State of the Art

The essential prerequisite for adaptation services is a representation of documents that unlocks and structures the content of documents so that they are comprehensible to a computer system. Various research addresses the transfer of document content into a form suitable for further processing and web presentations. This work focus on semantic web technologies and represents documents with XML markup languages.

The following sections discuss the state of the art for representing and adapting web documents. Section 2.1 starts with a comparison of XML-based (mathematical) document formats. The thorough analysis of XML formats is indispensable in order to select and extend one of these formats. The in-depth discussion of adaptation systems in Section 2.2 introduces the state of the art and verifies, whether these systems support the following adaptation services: the adaptation of notations, the reordering of document parts, and the substitution of document parts with alternatives that differ in, e.g., level detail, expertise, formality, or language.

2.1. XML-based Markup Formats

We have seen that markup languages can be characterised based on two paradigms: the topic-oriented paradigm and the document-centered paradigm. On a technical level the former is characterised by the markup of self-contained information units (i.e., topics), a markup of document structures, and the integration of a transclusion mechanism to embed topics in these document structures. The document-centered paradigm supports coherent documents.

Some markup languages explicate the *semantic context* of document parts, which includes a markup for

- the meaning of mathematical symbols and formulae (by integrating MATHML or OPENMATH),
- the dependencies between paragraphs and their properties,
- the implicit inheritance structure of mathematical knowledge.

For the adaptation of narrative documents, markup formats should also explicate the narrative context and user context. A representation of the *narrative context* can include a markup of cross-references (references to figures, tables, sections, etc) or transitions that allows to omit or insert these words and phrases according to the content of documents and its arrangement. Markup for the *user context* includes, e.g., annotations and conditions that allow users to personalise a document.

The following sections introduce the formats DITA, CNXML, DOCBOOK, MATHLANG, and OMDOC and analyse whether they support the topic-oriented or document-centered paradigm as well as the representation of semantic, narrative, and user context. The summary in Section 2.1.6 recapitulates the formats in a table and concludes which format is most appropriate in the context of this work.

2. Preliminaries & State of the Art

2.1.1. DITA

The Darwin Information Typing Architecture (DITA [PAH07]) defines an XML architecture for designing, writing, managing, and publishing various information in print and on the Web. It is based on principles of inheritance and specialisation (Darwin), supports a *topic-oriented authoring paradigm* (Information Typing), and expresses principles for creating and delivering topics (Architecture). In contrast to document-centered writings, DITA is designed for the creation of modular topics and to move away from books as the primary entity through which technical knowledge is conceived, created, and disseminated.

Based on XML, DITA distinguishes content and presentation markup and facilitates the conversion into various output formats like DOCBOOK (Section 2.1.3), PDF or XHTML. Reuse of content is a central concern in DITA: all content is only maintained once (stored in DITA topics) and reused by transcluding content into structural entities, called DITA maps. DITA supports three standard topic types: *concept*, *task*, and *reference topics*. Tasks represent an instruction or numbered list of instructions. Concepts include definitions and descriptions of terms. References provide a lexicon-like assembly of links for an efficient lookup of information. All topics share the same basic structure (title, description, prolog, body, related links, and nested topics). Listing 1 presents the markup for a DITA concept topic. The `concept` element is the top-level element. Every concept contains a `title` and a `conbody` and optional `titlealts`, `shortdesc`, `prolog`, and `related-links` section.

Listing 1: Markup of a DITA concept topic

```
<concept id="A.dita">
  <title>Natural numbers</title>
  <conbody>
    <p>
      The set of <term>natural numbers</term> is either the set {1, 2, 3, ...} (positive integers) or
      the set {0, 1, 2, ...} (the non-negative integers). More information can be found in <cite>
      Mathematics. A Very Short Introduction</cite> or <xref href="nat.dita#nat1"></xref>
    </p>
    <para conref="#topic1/p2" />
  </conbody><related-link>...</related-link>
</concept>
```

DITA provides different markup for cross-references and links within topics. Cross-references only occur in the body of a topic (e.g., the `conbody`) and are represented with an `xref`, `term`, or `cite` element. The `cite` element is used to mark bibliographic cross-references that refer to a book or article. Its content identifies the title of the resource. An optional `keyref` attribute allows the citation to be associated to other possible bibliographic processing, e.g., to specify the location of the cited material. The `term` element is used to mark a text (a definiendum) that requires further definitions. An optional `keyref` attribute allows to link to the respective definiens (e.g., in a glossary). The `xref` element allows users to point to other DITA topics in the same file (e.g., `#nat1`), to a specific element inside the topic (e.g., `#nat1/el.id`), to a topic in another file (e.g., `nat.dita#nat1`), or to a web resource (e.g., `http://www.openmath.org/cd/setname1.xhtml#N`). Note that DITA requires globally unique XML ids [MVW05]. This is assured by *scoped IDs*, which are represented with a *fragment identifier syntax*. For example `#nat1/el.id`, points to an element with `xml:id el.id` inside a topic with `xml:id nat1`. Authors can supply the label for a cross-reference by embedding text into the `xref` tag or leave it blank to use the title of the referenced topic as text for the link. An optional `type` attribute indicates whether a figure, table, footnote or topic is referenced and an optional `format` attribute

specifies the format of the target resource. Transclude operators within topics are represented with the `conref` attribute. In DITA, the `conref` operator includes only the content of the referenced element and has to point to an element of the same type as its parent element. For example, `#topic1/p2` in Listing 1 has to point to a paragraph.

The `related-links` section of a DITA topic is a special structure that supports the navigational rules from a topic to its related neighbour topics. These links (represented by `link` elements) are different from cross-references. They only represent topic-to-topic connections or connections to non-DITA-topic resources. Links can be grouped into container elements such as `linkgroup`, `linklist`, or `linkpool`. `linkgroup` allows authors to define groups of references with common attributes, `linklist` supports the markup of ordered sequences of references, and `linkpool` defines a group of links with common characteristics. Listing 2 illustrates a `related-link` section. The `linkpool` element groups two references (represented with a `link` element) with common type `concept` (they are both concept topics) and common target audience `students` (they are both useful for students). The `link` element can only express a syntactic relationship between two topics. It indicates that there is a relationship but does not mark its meaning.

Listing 2: Grouping of variant topics

```
<related-links>
  <linkpool type="concept" audience="students">
    <link href="set.dita"/>
    <link href="integers.dita"/>
  </linkpool>
</related-links>
```

DITA also supports the markup of alternative content within topics. Respective alternatives are annotated with conditions, which specify when an alternative text should be included. DITA provides several built-in attributes to hold the values for conditional filter criteria, these include `type`, `audience`, `platform`, `product`, `importance`, `revision`, and `status`. Currently, these conditional values are predefined. However, the DITA 1.1 specification proposes to enable authors to define their own metadata attributes in future releases of the format. Listing 3 illustrates the DITA conditional representation of two examples for different audiences: A computer scientist might prefer the first example on binary trees, while a mathematician might prefer an example on natural numbers.

Listing 3: Annotation of conditions in DITA.

```
<ph audience="compscience">
  Let us proof that all binary trees of height n, in which all non-leaves have 2 sons,
  have at least n+1 leaves. We proof the lemma by 'proof by strong induction' on the height ...
</ph>
<ph audience="math">
  Mathematical induction can be used to prove that the statement  $0 + 1 + 2 + \dots + n = \frac{n(n+1)}{2}$ 
  holds for all natural numbers n. ...
</ph>
```

To define which conditions apply when generating an output document, rules can be defined in separate `ditaval` files. Listing 4 illustrates an example of two rules: The first rule includes the example for computer scientists and the second one the example for mathematicians.

Listing 4: Extract of a ditaval file to define results for mathematicians.

```
<prop action="exclude" att="audience" val="compscience" />
<prop action="include" att="audience" val="math" />
```

2. Preliminaries & State of the Art

Listing 5 shows the markup of a DITA map, an organized collection of references used to represent the structure of documents. The top-level `topicref` element organises a set of topics in a hierarchy, which can be used to define print output, online navigations, or parent/child links. The `topicref` elements can point to a DITA topic, map, or any other processable resource.

Listing 5: Markup of a DITA map

```
<map title="title">
  <topichead navtitle="navi-title" audience="math" />
  <topicmeta></topicmeta>
  <topicref href="A.dita" collection-type="sequence">
    <topicref href="A1.dita"/>
    <topicref href="A2.dita"/>
  </topicref>
  <reltable>
    <relrow>
      <relcell>A.dita</relcell>
      <relcell>B.dita</relcell>
    </relrow>
  </reltable>
</map>
```

The `relate-links` element of concept topics allows authors to link within a topic to another topic. However, hard-coding of links in a topic creates dependencies between topics and reduces their reusability. Since the links are hard-coded within the topic, they may not apply to other contexts in which the topic is reused. If a topic is renamed or if its path changes, the link will have to be recreated in all topics which reference this link. Hard-coded links thus result in excessive maintenance. Relation tables (represented with `reltable` elements) provide an alternative for interlinking topics. They allow to create and maintain links independent from their topics. Links can be created both between topics of the same information type and between topics of different information types that are not directly related through parent/child relationships. The DITA specification suggests that the best practice for linking in DITA is to use a relationship table within a map. This supports a markup of the narrative context of documents that facilitates arbitrary combinations and arrangements of document parts, though not assuring coherent adaptation results.

To conclude, DITA is not primarily intended to represent narrative documents. It follows the topic-oriented paradigm and is mainly used to assemble topic-oriented documents. The document structure is stored in a separate file, a DITA map. All document content is organised in self-contained topics, each stored in a separate topic file. DITA maps include relationship tables to define relationships between topics. This stand-off markup of interdependencies minimises the number of embedded cross-references and, thus, increases the reusability of content. Conditional markup and various groupings of texts support the markup of alternative (or equivalent) content. The annotation of conditional filter criteria and specification of rules for the selection from conditional texts according these parameters, supports first notions of a user context. Authors can adapt their documents for different audiences, readers have no means to adapt the respective output further.

The DITA format does not explicate the full semantic context of mathematical documents. Neither the meaning of mathematical symbols and their notation nor the inheritance structure of mathematical documents can be explicated. All cross-references are syntactic, apart from the `term` element that interlinks a definiendum with its definiens. These limitations are due to the fact that DITA has not been designed for mathematics, where such structures are relatively overt. Nevertheless, its *specialisation module* promotes the creation of new standard

information types (or topics) and domain-specific markup vocabularies. Lately, the OPEN-MATH community became active in gathering requirements for a *math domain* for DITA, i.e., an extension of DITA for mathematics. They propose to introduce a `math` and a `mathph` element analogously to the `p` and `ph` DITA elements to embed mathematical expressions within DITA topics as well as a markup for associating these expressions with symbol definitions and descriptions in DITA [Nor09a, Nor09b]. As long as such a math domain has not been specified, DITA remains of little use for mathematics.

2.1.2. CNXML

The CONNEXIONS markup language (CNXML [Bar]) is a lightweight XML-based markup language for educational documents, developed for the CONNEXIONS project (see Section 2.2). The goal of CNXML is to convey the content of materials and not a particular presentation to provide smarter search and transformation of content into different output media via XSLT stylesheets.

Listing 6: CNXML representation of a document

```
<document id="angelica">
  <title>ANGELICA</title>
  <content>
    <section id='intro'>
      <para id='intro.p1'>
        Angelica is a European perennial plant.
        <note type='info'>It is sometimes grown in this country as a culinary herb.</note>
      </para>
    </section>
  </content>
</document>
```

Listing 6 illustrates a CNXML document (or CONNEXIONS module). Each CNXML document has a title, an optional metadata section, and a content section, represented with a `content` element. Structural elements are used to give structure to the document. They are defined as container for almost all other elements and include `content`, `section`, `example`, `meaning`, `proof`, `statement`, `problem`, and `solution` elements. A markup of document structures is not provided, transclusion of document parts is not supported. For example, authors that wish to add another proof to Listing 7 have to explicitly manipulate the `rule` element. They cannot simply add an operator to transcribe previously created content.

Listing 7: Markup for mathematical texts

```
<rule id='murph' type='law'>
  <title>Murphys Law</title>
  <statement>
    <para id='murph1'>
      If there are two or more ways to do something, and one of those
      ways can result in a catastrophe, then someone will do it.
    </para>
  </statement>
  <proof>
    <para id='murph2'>
      Edward A. Murphy, Jr. was one of the engineers . . .
      called <term url="http://murphy-law.info">Murphys law</term>.
    </para>
  </proof>
</rule>
```

2. Preliminaries & State of the Art

```
For further information see <cite><cite–title pubtype="book">A History of Murphys Law
</cite–title> by Nick T. Spark</cite>
<link url="http://www.amazon.com/Murphys–Law–Spark">(order here)</link>.
</para>
</proof>
</rule>
```

CNXML is not designed to represent mathematical structures. It only provides light-weight markup of mathematical content. Content-MATHML is used to markup the semantics of expressions and to facilitate conversion into arbitrary notations in Presentation-MATHML based on the XSLT-based workflow proposed by [NW03, HRB⁺02]. In addition, CNXML partially classifies content according to the mathematical terminology: Elements such as `equation`, `proof`, or `definition` allow authors to classify mathematical text segments. Listing 7 presents the markup of a rule of thumb, which can include one or more `statement` elements and zero or more `proof` and `example` elements. The arrangement of these statements in the `rule` element expresses their semantic relationships.

CNXML specifies different types of cross-references to interlink documents and CNXML elements: *module links*, *hyperlinks*, *citations*, and *concept references*. Module links and hyperlinks are represented with `link` elements. Module links are represented by specifying a `document` attribute (specifying a module or collection), a `target-id` attribute (specifying the target element within a module), and a `strength` attribute (defining the relevance of the link with value 1, 2 or 3). For representing hyperlinks the `link` element takes an optional `url` attribute. For example, in Listing 7 the `link` element marks a hyper reference to `http://www.amazon.com/History–Murphys–Law–Nick–Spark`. Citations allow users to reference non-electronic materials. They are represented with a `cite` element, which shares most of the attributes specified for `link` elements.

The `term` element represents concept reference, in particular, it marks words or phrases which are being defined. Its use is confined to either a `para` or `definition` element. The `term` element has several optional attributes: `url` specifies the source or definition of the term, `document` provides the id of the CONNEXIONS module or collection, `target-id` specifies the element (`para` or `definition`) in the current or another CNXML document, `reference` points to a file that is associated with the term in question, `version` provides the version of a module or collection.

In contrast to other markup formats, CNXML does not provide a sophisticated transclusion infrastructure for content reuse – it is solely designed to support a document-centered authoring style and can not support the editing of self-contained, atomic document parts¹. The mathematical support is limited: The inheritance structure of mathematical knowledge is not represented. The markup of interrelation between mathematical content snippets is limited, the meaning of dependencies can not be expressed. CNXML does also not support groupings of content with common characteristics or marking of parametrised texts.

2.1.3. DocBook

DOCBOOK [WM99] was originally developed for writing technical documents related to computer hardware and software but can be used for any other sort of documentation. Users create their documents in a presentation-independent form and can publish them in a variety of output formats.

¹One could argue that the CONNEXIONS system (Section 2.2.2) implements a mixture between topic-oriented and document-centered approach. The most coarse entity in CNXML are modules, which are authored in a book-like fashion but can be interpreted as large, topic-like entities. The CONNEXIONS system supports users to assemble these modules into larger entities.

DOCBOOK provides three categories of elements: structural, block-level, and inline elements. *Structural elements*, such as `set` (representing a document collection), `book`, `article`, `title`, `part`, `chapter`, `glossary`, and `appendix`, specify the document structure. *Block-level elements* include `paragraph` or `lists`. The markup of paragraph sequences is entirely neutral to the actual rendering, e.g., allowing to display block-level elements below each other for western languages and from right to left for Japanese. *Inline-level elements* are elements like `emphasis` or `hyperlink`. They wrap text within a block-level element. These elements typically cause the document processor to apply some kind of distinct typographical treatment to the enclosed text, by changing the font, size, or similar attributes. However, these presentational information are not encoded in the DOCBOOK format but can be attached to the inline-level elements, e.g., via an XSLT stylesheet [Wal09b].

Listing 8 provides a simple DOCBOOK document. The `xref` element forms a cross-reference from chapter `chapt1` to `chapt2`. The element can be empty to initialise a processing system to generate an appropriate label for the reader from the element referenced by the `linkend` attribute. Alternatively, if the `endterm` attribute is specified, the content of the referenced element must be used as label for the cross-reference. Hyperlinks are represented by `ulink` elements. Citations are represented with a `citation` element, where the content is assumed to be a reference string, perhaps identical to an abbreviation of an entry in a bibliography.

Listing 8: A simple DOCBOOK document

```
<book id="doc1">
  <title>A simple book</title>
  <chapter id="chapt1">
    <title>A simple chapter</title>
    <para>
      Please see <link linkend="chapt2">Chapter 2</link> or
      Chapter <xref linkend="chapt3" endterm="title3" />.
      Further details are given in <citation>Graזור 2009</citation>,
      a demo can be downloaded at
      <ulink url="http://mydemo.com">http://mydemo.com</ulink>.
    </para>
  </chapter>
  <chapter id="chapt2"></chapter>
  <chapter id="chapt3">
    <title xml:id="title3">A more detailed explanation</title> . . .
  </chapter>
  <xi:include xmlns:xi="http://www.w3.org/2001/XInclude" href="capt4.xml" />
</book>
```

DOCBOOK supports the modularization of documents into parts. These can be assembled using the W3C XINCLUDE [MOV06]. In Listing 8 all of chapter 4 is stored in a separate file. A processing system can copy the content of the file into the main document to provide the user with a self-contained document. The `href` value in an XINCLUDE can be an absolute path, a relative path, an HTTP-URL that accesses a web server, or any other URI. Granular embedding is supported by adding an XPOINTER attribute. If no XPOINTER is given the whole file starting from the root element is copied to the parent during the flattening [Sta07, chapter 23].

DOCBOOK focuses on the creation and publishing of self-contained documents and was not designed to support a topic-oriented authoring such as provided by DITA. [Wal09c] specifies an extension of DOCBOOK 5.0 that implements the key features of DITA: A topic-oriented authoring paradigm, the fragment identifier syntax for a more flexible cross-

2. Preliminaries & State of the Art

referencing scheme, transcribe operators, and an extensible approach. This extension shows that DOCBOOK can behave like DITA although its original specification follows a different way of creating and thinking about technical content, i.e., using *documents as interfaces* to capture, transfer, and present knowledge. Consequently, the elements in DOCBOOK carry meaning that is tied to document-centered terms such as `article`, `chapter`, and `section`. In contrast to DITA, where authors first create a number of topics and then assemble these in larger structural entities, authors of DOCBOOK files usually start with creating a document outline, which is filled with content. In this process cross-references and transitions are added to improve the coherence of the writings. Consequently, although DOCBOOK can theoretically support a topic-oriented writing, it has been designed to support traditional, document-centered authoring approach. The extension of DOCBOOK for DITA remains experimental.

A major drawback of DOCBOOK for the herein described approach is the lack of markup for mathematical structures and interdependencies. Though the integration of Presentation-MATHML is supported, DOCBOOK does not integrate Content-MATHML or OPENMATH to mark the functional structure of mathematical expressions², lacks the marking of semantic relations between document parts, and was not designed to reflect the inheritance structure of mathematical knowledge. In addition, DOCBOOK does not address conditional markup or grouping of texts with common characteristics as well as the annotation of user preferences for the adaptation of documents.

2.1.4. MATHLANG

The MATHLANG [KWZ08] project provides a representation language for mathematical documents, i.e., the MATHLANG format, as well as a framework for writing mathematical texts, i.e., a set of software tools for the authoring of MATHLANG. The project aims at connecting different representation languages for mathematics, from languages for the mathematical vernacular such as \LaTeX , to semantic representations like OPENMATH, Content-MATHML, or OMDOC (Section 2.1.5), to fully formal representation languages of proof assistants like Isabelle [NPW02] or COQ [BC04] by providing various degrees of formalisation and compatibility to different logical frameworks, such as set theory, category theory, and type theory.

The format supports the editing of mathematical texts in an XML format and using the $\text{\TeX}_{\text{MACS}}$ editor [TeX05]. Supported with the MATHLANG editors [KWZ08], authors can mark the roles of chunks of their text (theorem, definition, example, section, etc.) and can indicate the relationship between them (uses, contradicts, follows from, etc.). These annotations are used to convert the document into automatically generate proof skeletons, which can be imported into proof assistants for formal verifications. In this sense, MATHLANG suits as an interface between professional mathematicians and proof assistants.

The MATHLANG format specifies three layers for the representation of mathematical texts: the core grammatical aspect (CGa), the text and symbol aspect (TSa), and the document rhetorical aspect (DRa).

CGa assigns categories (term, statement, noun, etc.) to parts of the texts, deals with binding names to meaning, and checks that a kind of grammatical sense is maintained. TSa supports author to integrate mathematical authoring representation, such as \LaTeX , XML or $\text{\TeX}_{\text{MACS}}$ with CGa data, while structuring their text according to the mathematical vernacular (or the common mathematical language). MATHLANG distinguishes two kinds of structural document units: *division elements* that express the textual structure, e.g., `part`, `chapter`, and

²Note that the MATHDOX project (Section 2.2.5) has extended DOCBOOK with OPENMATH. Since this extension is not officially permitted by the DOCBOOK group, it is omitted from this overview.

section, and *mathematical units* expressing the mathematical structure of a text, such as theorem, lemma, or proof. According to these structural units, MATHLANG distinguishes two roles of document units: *ordinary document sectioning roles* and *mathematical roles*.

DRa provides an ontology, which defines these two roles as well as relations (`inconsistentWith`, `justifies`, `subpartOf`, `uses`, `relatesTo`, and `exemplifies`) between the structural units of a document. Mathematicians have to mark these relations between the part of their documents (Figure 6).

From this markup, the MATHLANG processing systems can automatically extract the *dependency graph* for a document. The document dependency graph is a directed labelled graph, where vertices represent structural units (mathematical or structural rhetoric roles) and the edges represent their relations. Figure 7 presents a dependency graph for the example document in Figure 6. For each kind of relation in the dependency graph, MATHLANG computes its *textual order*. In particular, MATHLANG distinguishes three kinds of textual order [KWZ08]: *strong textual order* \prec (A succeeds B), *weak textual order* \preceq (A is a subpart of B), and *common textual order* \leftrightarrow (two nodes use at least on common symbol or statement). These orders are used to generate a *graph of textual order* (GoTO [KWZ08]) between the annotated parts of a text. GoTO is a directed graph with labelled edges, where vertices correspond to the vertices in the dependency graph. The direction of the edges and their labels express the logical precedence of two vertices and corresponds to the relations in the dependency graph.

The GoTO is used to generate proof skeleton for formal verification in a proof assistant. The generation of proof skeleton has two parameters: an input document with DRa annotations and a configuration for the proof assistant. These inputs are used to rearrange the nodes in the graph and to convert them into the input format of the respective proof assistance system. Though intended for the communication with proof assistants, the MATHLANG GoTO graph is used to generate a reordered version of the document.

MATHLANG places less focus on adaptation of notations or document content. Content reuse and modularisation are supported by a granular-transclusion architecture [KWZ08, p.17]. Annotation of user preferences, markup of document structures, and the markup for conditional texts is not in the scope of the project. Nevertheless, the conversion of a dependency graph into a GoTO graph has inspired the discussion in Section 7.1.2.

2.1.5. OMDoc

The XML-based, web-scalable Open Mathematical Document Format (OMDOC [Koh06]) serves as document markup format and ontology language for mathematical documents on the World Wide Web. Similar to MATHLANG, the OMDoc project provides a plethora of libraries and system to support the authoring and maintenance of documents in OMDoc (Section 10.3.6). OMDoc focuses on the presentation-independent markup of documents and supports the publishing of documents in presentation-oriented formats such as XHTML. Presentational information are not embedded in the format but can be attached to any OMDoc element, e.g., via XSLT stylesheets [Koh]. The OMDoc project provides a \LaTeX -based authoring interface. Authors can draw on a semantic extension of \LaTeX , called $\mathcal{S}\TeX$ [Koh08c], to markup their \LaTeX documents. These can be transformed into PDF using the standard PDF \LaTeX workflow or XML (i.e., OMDoc) drawing on the L \TeX ML translator [Mil]. The author's annotations are embedded into the document and are not provided as external specification. OMDoc does not enforce a specific granularity for the markup of documents but offers various levels of formality — from minimal markup to fully formalised representations [Koh06, chapter 4].

2. Preliminaries & State of the Art

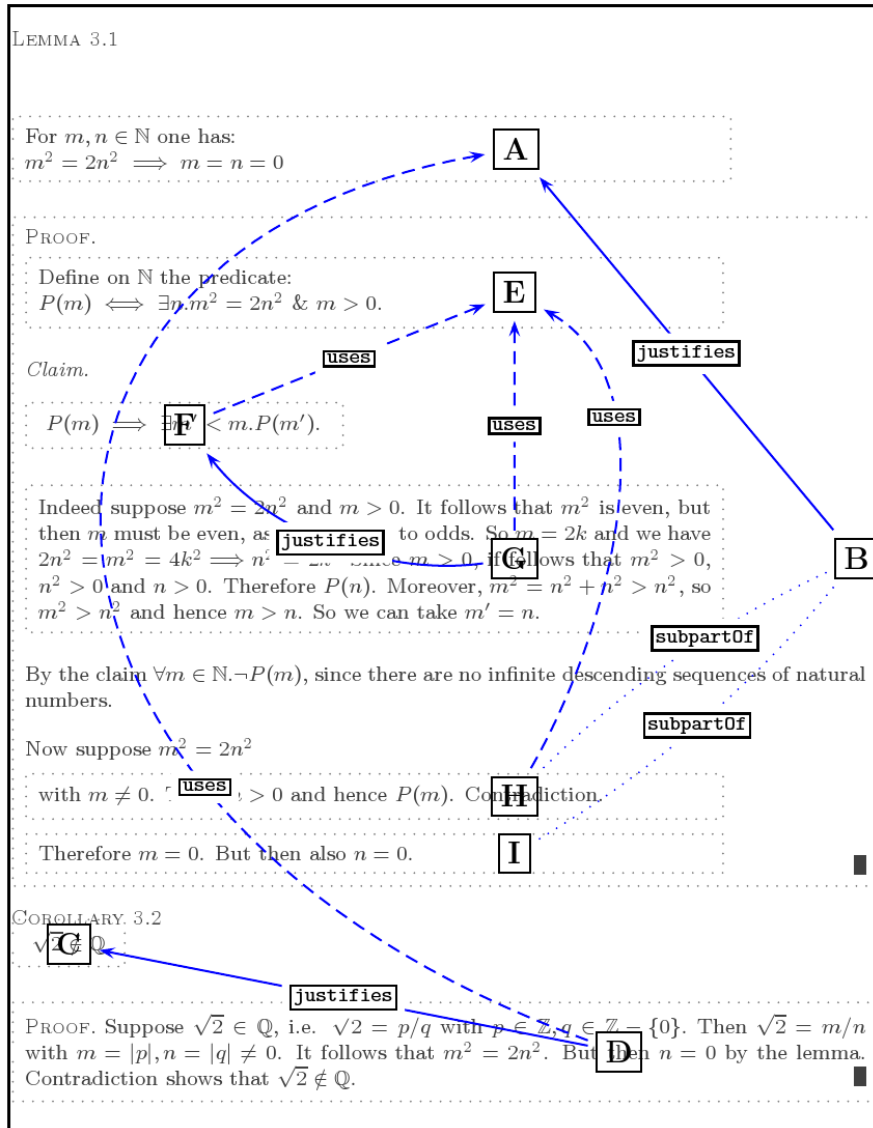


Figure 6.: Markup of texts [KWZ08, p.18]

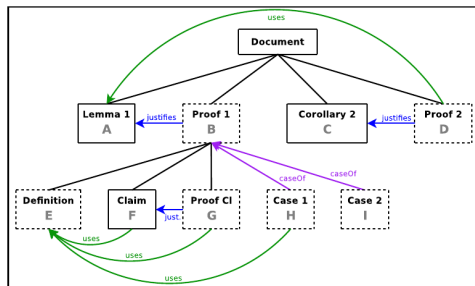


Figure 7.: Dependency graph [KWZ08, p.19]

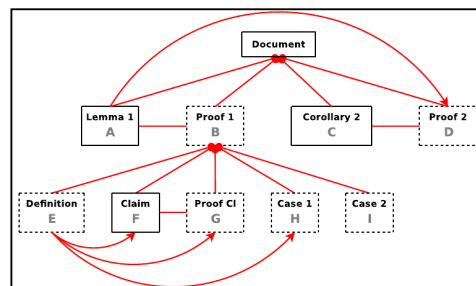


Figure 8.: GoTO [KWZ08, p.25]

OMDOC allows authors to classify and interlink all parts of their documents according to the mathematical terminology. In particular, OMDOC distinguishes three layers of mathematical knowledge for the representation of document content: the *symbol*, *statement*, and *theory layer* [Koh06, chapter 3.2].

Symbols are objects we talk and write about when we do mathematics and are presented by mathematical notations. OMDOC embeds OPENMATH and Content-MATHML for the representation of symbols and Presentation-MATHML for the representation of notations. It provides an XSLT-based conversion mechanism that allows to convert OPENMATH and Content-MATHML into Presentation-MATHML, while adapting the generated notations according to context parameters such as language or output format.

Mathematical symbols are embedded into paragraphs of mathematical texts, such as formulae, proofs, lemmas, or definitions. In OMDOC, these text fragments are referred to as mathematical *statements*. The meaning of statements is annotated by specific XML elements, such as `definition`, `example`, or `proof`. Statements are interlinked by ontological relations (`defines`, `illustrates`, `proves`, etc), which are represented with `for` or `xref` attributes. These attributes take a URI reference as value, which identifies another mathematical statement or symbol. For allowing other elements to reference them, most OMDOC elements carry a unique identifier, represented with the `xml:id` attribute [MVW05]. OMDOC symbols are referenced by pointing to the value of their `name` attribute.

Listing 9 presents the markup of a `definition` that defines the subset symbol. It is illustrated by an `example`. The relations of both statements is given in their `for` attributes, the semantics of the relation is encoded in the OMDOC ontology. According to this ontology, the `for` attribute of a `definition` element indicates the *defines* relation. It takes a whitespace-separated list of URI reference to `symbol` elements. The `for` attribute of an `example` is interpreted as *exemplifies*.

The `ref` and `link` element in Listing 9 represent cross-references to other resources. The processing of the `ref` element with type `cite` is application specific. The OMDOC specification recommends to generate an appropriate label and (optionally) supports hyper references to electronic materials. Citations of non-electronic material are not mentioned in the specification. The `link` element represents hyper references to web resources.

Listing 9: Markup of mathematical statements

```
<symbol name="symA" /><symbol name="symB" /><symbol name="symC" />
<definition xml:id="def.abc" for="symA symB symC">
  A is B + C.
  More information can be found in
  <ref type="cite" xref="http://www.maa.org/reviews/mathvsi.html" /> or
  <link href="http://www.maa.org/reviews/mathvsi.html">Mathematics.
  A Very Short Introduction</link>.
</definition>
<example for="#def.abc">illustration for def.abc</example>
```

Fig. 9 illustrates an extract of the OMDOC ontology, which specifies categories for mathematical texts (symbols, statements, and theories) as well as the semantics of their mathematical relations.

OMDOC arranges symbols and statements into larger entities, referred to as mathematical *theories*. Similar to content dictionaries (CD) in OPENMATH [OMC] these theories serve as an explicitly represented context for statements and mathematical symbols. However, OMDOC extends the functionality of OPENMATH CDs with a very expressive and extensible infrastructure for relations within and among theories that facilitate concept inheritance, parametric reuse, and multiple views on mathematical symbols and statements.

2. Preliminaries & State of the Art

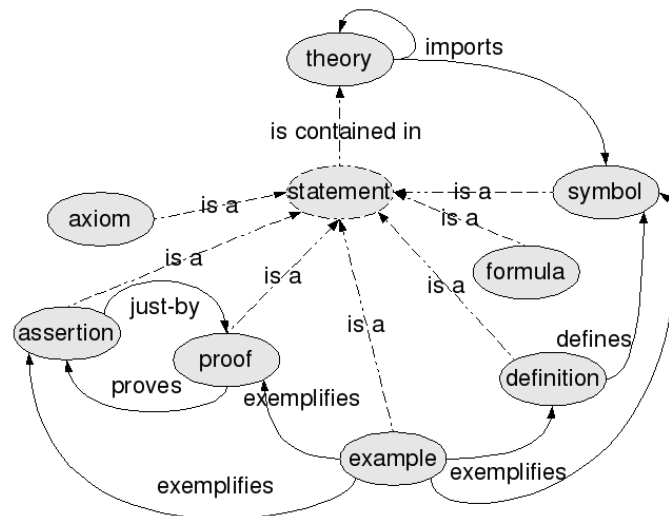


Figure 9.: An extract of the OMDOC ontology on the basis of [Lan06, Koh06].

Listing 10 illustrates the markup of two theories, for convenience the example does not mark mathematical expressions in OPENMATH or Content-MATHML but draws on \LaTeX . Theory `subset.T` defines the subset symbol and theory `prop.T` defines the proper subset. Both theories embed definitions and examples that illustrate these concepts. The definition of the proper subset builds on the subset concepts but excludes the identity of sets, it is a specialisation. This relation is explicated by the `imports` element in its theory. The imported theory, i.e., theory `subset.T`, is the more general one.

Listing 10: Generalisation and specialisation of symbols

```
<omdoc xml:id="my.cd">
  <theory xml:id="subset.T">
    <symbol name="subset"/>
    <definition xml:id="defs" for="subset">
      A subset is a portion of a set.  $B$  is a subset of  $A$  (written  $B \subseteq A$ ) iff every member of  $B$  is a
      member of  $A$ .
    </definition>
    <example for="#defs">...</example>
  </theory>
  <theory xml:id="prop.T">
    <imports from="#subset.T"/>
    <symbol name="prosubset"/>
    <definition xml:id="defps" for="prosubset">
      A proper subset  $S'$  of a set  $S$ , denote  $s' \subset S$ , is a subset that is strictly contained in  $S$  and so
      necessarily
      excludes at least one member of  $S$ .
    </definition>
    <example for="#defps">...</example>
  </theory>
</omdoc>
```

OMDOC supports the markup of alternative texts within mathematical statements, but is limited to the markup of language and formality variants. Listing 11 shows the markup of a mathematical statement, represented with an `omtext` element. It includes two alternative

text fragments, one in English and one in German. These are embedded in a `CMP` element. The default language is English, an `xml:lang` is used to indicate the alternative language (e.g., German). OMDOC provides a set of XSLT stylesheets [Koh], which take as input parameter the target language and select the respective `CMP` during the conversion from OMDOC to XHTML. Similarly, OMDOC marks formal variants of a text using a `FMP` element, which takes an optional `logic` attribute to indicate the formal language of its content. In general, the content of an `FMP` element is represented in `OPENMATH`. `FMP` elements can not yet be selected via the XSLT transformation.

Listing 11: Markup for language variants

```
<omtext>
  <CMP>This is an English text.</CMP>
  <CMP xml:lang="de"> Die ist ein deutscher Text.</CMP>
  <FMP><OMOBJ>...</OMOBJ></FMP>
</omtext>
```

OMDOC supports the creation of self-contained documents as well as the specification of stand-alone document structures. For this, OMDOC separates the markup of content and structure: Content OMDOCs are “*knowledge-centered documents that contain the knowledge conveyed in a document*” [Koh06, p. 72], this includes all elements defined by the OMDOC content ontology. In contrast, narrative OMDOCs are used to “*reference the knowledge-centered documents] and add the theoretical and didactic structure of a document*” [Koh06, p. 72]. Listing 12 illustrates the markup of a document structure in OMDOC. The `omdoc` and `omgroup` element are container elements. They embed content, other container elements or pointer to elements, represented by `ref` elements (of type `include`). The `ignore` element represents comments. These can be arbitrarily nested and may occur as an OMDOC top-level elements or within mathematical texts. The `ignore` elements are usually used to comment the OMDOC representation and to omit the respective document content in the final presentation of the document. In Listing 12 they are used to embed content that is referenced in the document structure.

Listing 12: OMDOC representation of document structures.

```
<omdoc xml:id="doc1">
  <omgroup>
    <ref type="include" xref="#enum"/>
    <ref type="include" xref="#t3"/>
  </omgroup>
  <ref type="include" xref="#st1"/>
  <ignore type="targets" comment="content referenced by ref elements">
    <omgroup xml:id="enum" type="enumeration">
      <ref xref="#t1"/>
      <ref xref="#t2"/>
    </omgroup>
    <omtext xml:id="t1">T1</omtext>
    <omtext xml:id="t2">T2</omtext>
    <omtext xml:id="t3">T3</omtext>
  </ignore>
</omdoc>
```

The original motivation of OMDOC was to provide a format for the integration of proof assistants (similar to `MATHLANG`). It thus thoroughly marks the semantic context of mathematical knowledge, including the inheritance structure of knowledge, semantic relations

2. Preliminaries & State of the Art

and dependencies, and the representations of symbols and formulae. Fortunately, OMDOC has been extended with an authoring and publishing infrastructure similar to DITA. It thus bridges mathematical markup languages like MATHLANG and document formats like DOCBOOK or CNXML.

Nevertheless, OMDOC currently does not consider the narrative context of documents. A topic-oriented approach can only be supported if authors omits transitions and cross-references. Moreover, the markup of user-specific conditions is limited to metadata like language and formality. User can also not yet conveniently guide the selection from conditional texts, but are required to adapt the OMDOC XSLTs.

2.1.6. Summary

The table below summarises the discussed formats. DITA is designed for a topic-oriented authoring approach but lacks the support for coherent documents. CNXML and DOCBOOK focus on a document-centered approach but lack the markup of self-contained information units, document structures and/or transclusion mechanisms. OMDOC and MATHLANG can support a topic-oriented approach if authors focus on writing self-contained mathematical modules and omit narrative transitions and informal cross-references. DITA, CNXML, and DOCBOOK do not fully explicate the semantic context of documents. In particular, they lack the markup of interdependencies between document parts as well as the inheritance structure of mathematical knowledge. DITA and DOCBOOK do not represent the meaning of mathematical symbols and formulae. CNXML, DOCBOOK, and MATHLANG neglect a markup of user preferences, e.g., in form of the conditional texts in DITA and OMDOC.

Format supports ...	DITA	CNXML	DOCBOOK	MATHLANG	OMDOC
markup of topics	yes	no	no	yes	yes
markup of document structures	yes	no	no	no	yes
transclusion mechanism	yes	no	yes	yes	yes
coherent document structures	no	yes	yes	yes	yes
semantic context:					
integration of Content-MATHML or OPENMATH	no	yes	no	yes	yes
markup of semantic dependencies/properties	yes	yes	no	yes	yes
implicit inheritance structure	no	no	no	yes	yes
narrative context	yes	yes	yes	no	yes
user context	yes	no	no	no	yes

The OMDOC format supports most requirements and is thus selected as example³. It provides the best infrastructure for a topic-oriented as well as document-centered authoring approach. However, the authoring of self-contained, independent units in OMDOC depends on the author. The format can not yet sufficiently support authors to modularise narrative documents and lacks a markup for transitions. The format is further limited in the rendering of mathematical notations. Part II proposes a new rendering workflow for OMDOC. In Part III, the modularisation of narrative documents in OMDOC is discussed and the rendering workflow is extended to support the substitution and the reordering of document parts.

³This thesis is based on OMDOC 1.2 [Koh06]. Note that an important criteria for choosing OMDOC was also a pragmatic one. As the author is part of the OMDOC group, proposed extensions and revisions could be more easily discussed and verified. Most of the proposed extensions will be included in the subsequent versions of OMDoc.

Note that the proposed adaptation services are not limited to OMDOC but can be applied to any other XML based mark-up format, if the format is extended respectively. For example, future work could extend DOCBOOK with annotations that explicate the semantic context of the conveyed knowledge, that support the modularisation of documents, and that add conditions, which allow users to guide the substitution and reordering of document parts.

2.2. Systems for Document Adaptation

Acknowledgement: The author would like to thank all developers and researchers of the systems in this section. Special thanks go to George Goguadze, Jan Willem Knopper, Christoph Lange, Elvira Popescu, Carsten Ullrich, Rikko Verrijzer, and Marc Wagner for their valuable feedback.

This section analyses several systems and verifies whether they implement the adaptation services in the scope of this work. These include the adaptation of mathematical notations, the reordering of document parts, and the substitution of document parts with alternatives that differ in level detail, expertise, formality, or language. In addition, the following aspects are observed:

- Does the system follow a document-centered or topic-oriented approach?
- Is the system web-accessible?
- Does the system focus on mathematical knowledge management (MKM⁴)?
- Does it exploit the semantic context, user context, and narrative context for its adaptations?

In the following sections, we look at the PLAT Ω , CONNEXIONS, ACTIVEMATH, SWIM, MATHDOX, and WELSA system and analyse whether they fulfil the above requirements or not. The choice does not express any judgement on remaining systems, of which some could have been chosen analogously. The summary in Section 2.2.7 recapitulates the systems.

2.2.1. PLAT Ω

The mediator PLAT Ω [AFNW07] integrates the proof assistant system Ω MEGA [ABD⁺06, HKK⁺96] with the standard scientific text-editor $\text{T}_{\text{E}}\text{X}_{\text{M}}\text{A}^{\text{C}}\text{S}$ [TeX05] and Microsoft Word [Cor]. Figure 10 illustrates a screenshot of the Word interface. In contrast to the remaining systems, which can only handle documents, represented in one of the formats in Section 2.1, PLAT Ω processes the fully formalised content in the Ω MEGA system. The system supports authors to write documents in a *document-centered* fashion. Doing so they can draw on services of the Ω MEGA backend, such as automatic verification of formal proofs. The system is implemented as stand-alone application and does not provide an online interface, thus, can not yet provide web accessible contents.

PLAT Ω considers the notation contexts of documents as a dynamic parameter in separation to the document's content. By processing the marked up notations in a document, the PLAT Ω system is able to extract the semantics of the mathematical notations contained in the document and, thus, can model the author's notation practice. The document can be automatically adapted to the author's preferences in the case of notational changes, thus, implementing adaptable notations [WM07].

⁴Mathematical knowledge management (MKM) is a relatively new field of research, which aims at developing better ways to articulate, organise, disseminate, and access mathematical knowledge [CF09]. Adaptation systems, which contribute to this field, are closely related to this work as they also exploit mathematical structures to improve adaptation services.

2. Preliminaries & State of the Art

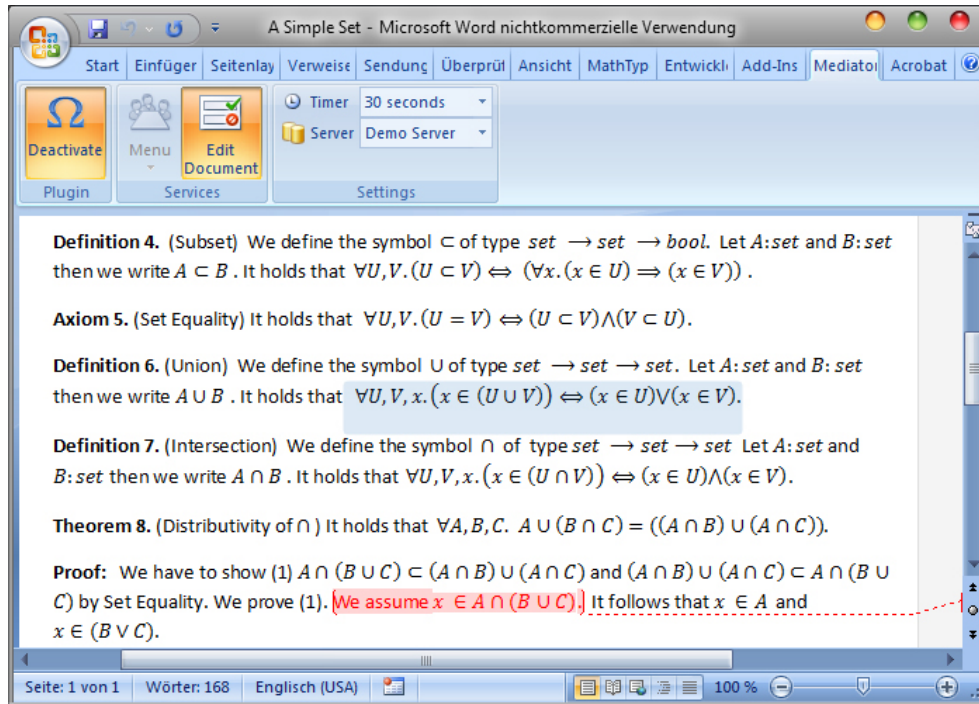


Figure 10.: PLATΩ integration into Word kindly provided by Marc Wagner

Based on [Fie01], PLATΩ can generate user-adaptive proof explanations that vary in order, level of detail, and formality. The adaptation approach in PLATΩ is not based on marked up documents but draws on information management techniques, such as natural language processing. For example, in order to provide different details, an XML-based markup approach can only select between two paragraphs with different level of detail. In contrast, PLATΩ can generate the required output from a single, formal source in the ΩMEGA system. Both approaches have their advantages and drawbacks. The PLATΩ approach tremendously reduces redundancies in a knowledge base. In return, it requires fully formalised materials and is limited to outputs that can be generated (e.g., multilingual presentation are not yet supported). Systems drawing on markup techniques reduce the formalisation effort of authors to markup of text categories, relations, and inheritance structure. They are limited to selecting from this content, which have been manually provided by the author. Alternative presentations with varying level of detail or formality have to be provided redundantly and cause consistencies as well as change management issues in the underlying document repository [MW07].

2.2.2. CONNEXIONS

The educational knowledge repository CONNEXIONS [BBH⁺02, Hen04] facilitates the aggregation of course material in form of coarse-grained knowledge chunks called *modules* that can be organised as *collections* – the content object that define books, courses, and other complex course structures within CONNEXIONS. The modules are represented in the CONNEXIONS markup language (CNXML, Section 2.1.2) and are stored in a versioned repository, called the CONNEXIONS content commons. Each item in the content commons is associated with metadata, such as language, subject, keywords, or author. While the modules can be authored in a book-like fashion, their assembly into collections is similar to the embedding of

DITA topics into maps (Section 2.1.1). We could thus say that CONNEXIONS implements a mixture between topic-oriented and document-centered approach. However, though written as self-contained units, modules are very large in comparison to topics in DITA. Adaptation services based on these coarse-grained units are rather limited.

CONNEXIONS supports three user groups: Authors, who collaboratively write the modules, instructors, who create collections from these modules, and learners, who can search for and explore materials. Authors create modules, which are (ideally) standalone, self-contained pieces of learning content. While there are methods for referencing elements in other modules (via links and numbered references), there is no method for automatically transclusion content from another document. Thus, while there are tools to help the author to provide links and references to other web resources, all content needs to be defined within the module and cannot be dynamically included from other sources.

The *course composer* supports instructors to create collections by stringing a series of CNXML modules together. Instructors first create several sections and subsections and form the table of contents (toc) of a collection. They then add modules to any of the toc's components. The course composer also lets instructors provide additional *imposed links* to reference *supplemental* or *prerequisite* items. These links are presented to the students to help them understand the relationships between materials and to encourage individual exploration. They can be interpreted as some form of recommendation. Finally, the Course Composer creates a course description file out of the imposed links along with the selected course toc.

Currently, the structuring of modules into collections has to be done manually by the instructors and can not be automatised based on semantic interrelation, competencies, or user-specific input parameters. To allow for more flexible reuse of modules, the CONNEXIONS group is currently working on a new XML model for collections (COLLXML) that will replace the current methods and support dynamic and automatised embedding of modules on the collection level [Emm09].

Assembling materials — manually or automatically — from different authors can cause inconsistency, e.g., if mathematical notations mismatch. CONNEXIONS addresses this problem by utilizing Content-MATHML. The repository stores only semantic information, leaving notation and presentation specifics up to the instructor, which are encourage to specify notational parameters for displaying their course. The course composer stores these choices in the course description file and then uses a limited implementation of the techniques described by Naylor and Watt [NW03] to apply the notational preferences to all modules of the course's roadmap.

To access the course collections, students use the *roadmap navigational software*, implemented as a browser add-on using the eXtensible User-Interface Language (XUL [Pro]). It supports students to study the module, request materials with *similar* content, materials that *embed* the currently viewed module, and *prerequisite* as well as *supplemental* links. The importance of the related material is indicated by visual markers [Hus09]. CONNEXIONS does not implement any user modelling techniques to adapt the presented materials to the students preferences or competencies.

To express the approval and authorship for modules, CONNEXIONS utilises the concept of *lenses* [KBB08]. These are selection of content in the CONNEXIONS repository to help readers find content that is related to a particular topic or focus. Three types of specialized lenses are currently supported: *endorsement lenses* for reviewed materials, *affiliation lenses* for material created by members of the lens creator's organization, and *member list lenses* for all other purposes. Lenses can be private or public. Figure 11 shows the 'General Computer Science lens' (a lense created for test purposes), which currently includes references to the two modules 'different kinds of numbers' and 'numbers – where do they come from'.

2. Preliminaries & State of the Art

PUBLIC LENSES
All lenses
▪ Endorsements (5)
▪ Affiliations (12)
▪ Member lists (47)
What's a Lens?

TAGS
Select a tag to narrow the contents of this lens.
gencs

RECENTLY VIEWED
Modules
[x] Numbers - where do they come from?
[x] Different kinds of numbers
[x] Help Using the Connexions Roadmap
See all recently viewed...

General Computer Science Lens (edit lens)
Lens by: Christine Müller

Selected content (what's this?)
2 modules (What are [x] modules and [y] collections?)

Add selected content to: Personal Workspace Add
Sort by: Type Results per page: 10

View: Lens | Detail | Compact | Statistics

Different kinds of numbers (m31086)
Author: gert bezuidenhout
Lens Tags: gencs
Lens Comments: Lens for the General Computer Science Course

Numbers - where do they come from? (m31203)
Author: gert bezuidenhout

Add selected content to: Personal Workspace Add

Popularity is measured as percentile rank of page views/day over all time

Feeds: [Subscribe to this feed](#) | [All Connexions Feeds](#) | [About RSS Feeds](#)

Figure 11.: The general computer science lens [Mül09b]

The CONNEXIONS system provides a discussion and annotation infrastructure for authors, instructors, and learners. Each CONNEXIONS module has its own discussion forum, which allows authors to collaboratively improve their content or students to pose questions. All forum entries are public. In addition, all users can enter annotations, i.e., notes that are associated with a specific point in the text of a course or module. These notes do not appear in the text but appear in a dialogue box that is opened if users click on an annotation icon in the text. Annotations of authors and instructors are public and are not connected to a specific user, while annotation of students are private and user-specific. The implementation of the CONNEXIONS annotation infrastructure is based on ANNOTEA [Koi05].

To conclude, CONNEXIONS is a comprehensive eLearning system but does not focus on adaptation. It provides means to adapt notations but these can only be used by the creators of course materials, they are not accessible to students. Limitation of the implemented rendering approach are discussed in Part II. Instructors can adapt document structures and content manually by arranging modules into their course materials. An automatised reordering or sequencing of modules according to user preferences is not provided. Discussion and annotation features support formation of discussion groups. User and group specific pre-selection of content in from of lenses provide a notion of relevance. Apart from this, only rudimentary recommendations of alternative, supplemental materials are available.

2.2.3. ACTIVEMATH

The eLearning system ACTIVEMATH [MS04, MAF⁺01] is a web-based adaptive hypermedia system that adapts learning objects to individual user preferences, learning goals, and competencies stored in a user model [Mel01]. Figure 12 presents a screenshot of the main menu of ACTIVEMATH. The entries on the left hand side correspond to manually authored books. The menu on the right hand side allows a learner to start the course generation wizard [Ull08]. With a click on one of the items to the left, users receive a predefined course. A click on 'create a book' to the right, initialises the generation of a new course that adapts to a manually entered area of interest (differential calculus, fractions, etc), the type of book (discover, rehearse, train competencies, simulate an exam, etc), and a number of explicitly provided topics. The book type encodes a pedagogical objective (called *scenario*).

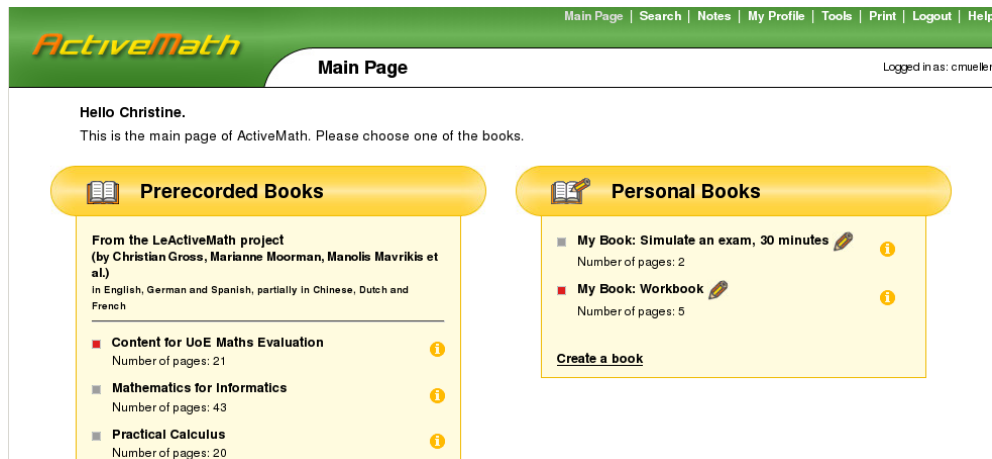


Figure 12.: Screenshot of the main menu of ACTIVE MATH, retrieved from [Act]

ACTIVE MATH implements a mixture of explicit and implicit user modelling: Information on users is either manually specified by the user or is inferred by the system by analysing the user's interactions. The user model includes static and dynamic information, such as the students' interaction history, the learning goals, field, scenario, knowledge mastery data (for each learning object in the system), and appearance preferences. During the registration, users are asked to describe their language preferences, field, educational level, and computer skills. The field (computer science, mathematics, etc) has an impact on the type of examples and exercises the system includes in a course. The learning goal corresponds to a concept in the underlying mathematical knowledge base and influences the content of a course. The knowledge mastery data is updated with training and assessment data from the exercise system [Gog09b, Gog09a]: The assumptions about the mastery of an exercise is based on the correctness of its solution. The mastery value of a section is the average mastery of the exercises related to the respective concepts of the section. The system's assumptions on the user's current knowledge are visualized by differently coloured boxes of each course section (e.g., red means failures in Figure 12).

The course material in the system consists of **learning objects** (or educational resources), which are defined as

“an atomic, self-contained learning object that is uniquely identifiable and addressable [...]: An educational resource must consist of the smallest possible (atomic) but still understandable and complete learning material (self-contained). If any content is removed from such an educational resource, then it can not longer be grasped without referring to additional resources. [...] an educational resource is accessible through the Web (addressable), identified using an [Uniform Resource Identifier [BLFM05]] URI” [p. 12][UII08].

The same learning object can be used in multiple courses. This reuse of learning objects imposes constraints on the content: transitions and cross-references have to be omitted, which reduces the coherence and readability of the assembled documents.

“In comparison to a standard textbook, absolute references to previous or latter content have to be avoided, because it is impossible to tell in advance whether the referenced educational resource will be presented at all and at which positions they will be presented. For the same reason, authoring introduction to a course

2. Preliminaries & State of the Art

or summaries is difficult: at authoring time, the educational resources contained in a particular course are unknown. But introductions, summaries and similar text have pedagogical purposes that is relevant for a successful learning process, and which a simple sequence of educational resources lacks” [UII08, p. 101].

For the representation of learning objects, the system draws on OMDOC (Section 2.1.5). The fine-grained markup and the separation of document structure and content fragments in OMDOC, allows the ACTIVEMATH system to dynamically adapt the arrangement of text fragments inside the user-specific documents and, thus, to flexibly reuse them. The adaptation is not based on the mathematical context of learning objects but on didactic requirements in form of pedagogical rules and competencies. These competencies corresponds to the mastery data in the learner models and are used to identify appropriate content for the user. They are specified by the ACTIVEMATH competency scheme, which evolved from Bloom’s taxonomy of learning goal levels, to the PISA standard methodology [KAB⁺04], to an interoperable competency scheme [MFEN08]. The pedagogical metadata is defined in the *ontology of instructional objects* [UII08] as illustrated in Figure 13.

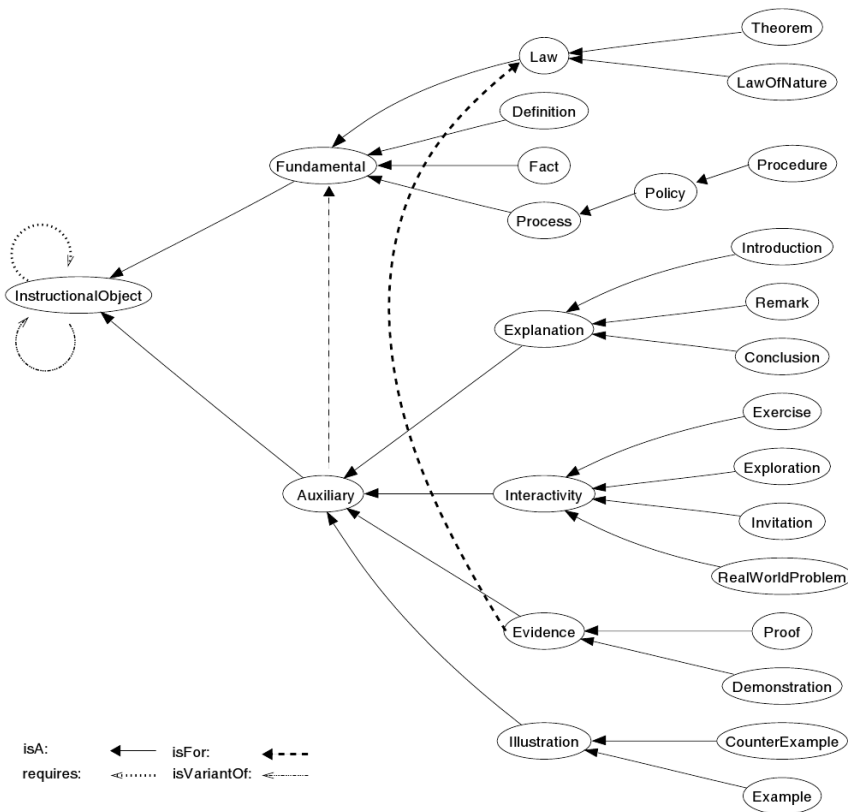


Figure 13.: Ontology of Instructional Objects [UII08, Figure 4.2, p. 50]

The ACTIVEMATH *presentation system* [ULWM04] implements a translation workflow from OMDOC to XHTML in two stages: In the first stage, the requested content is collected, pre-processed, and transformed into XHTML using XSLT stylesheets. In the second stage, the content fragments are assembled into a complete document page and enriched with personalised and dynamic data. The assembly is based on page templates provided by a tem-

plate engine [Vel]. The major function of the template engine is to replace meta variables by corresponding values, e.g., the user's name. During the personalisation, the language of the content and all mathematical notations [Lib07, MLUM05] are adapted to the user model, e.g., to the user's national background [LW06].

The central component of ACTIVEMATH is the *course generator* [UII08], which takes over all personalisation of the course materials. When using the course generation wizard (Figure 12), the course planner generates the initial input for the presentation system, i.e., a stable of contents that adapt to the learner's competencies. The course generator is also called by the presentation system to take over all personalisation in the second stage of the presentation process. The most relevant factors for both personalisation processes are the educational level of the learner and his competencies. Both should correspond to the respective properties of the selected learning objects, i.e., their competency level and learning context.

The course generator is implemented as hierarchical task network planner (HTN [UII08]) and thus handles a *planning domain* (encoded and evaluated with pedagogical experts) and *planning problem*. The planning domain is composed of operators, methods, axioms, and external functions. Planning problems are composed of an initial state with the goal of the learner and possible operators (or tasks) that can be performed. The HTN planner returns a plan, i.e., a sequence of operators (or tasks). When applied these operators generate a structured list of references to educational resources and learning-support services, which are passed to the presentation system and resolved to create a course. During the planning, the user model is accessed to conditionalise the learning context of the selected exercises and their difficulty or to select from multilingual alternatives.

To overcome the lack of transition phrases in the authoring phase, the course generator generates and inserts bridging texts, which explain the purpose of a learning object, include cross-references to other learning objects, and smoothen the transitions between them. Based on empirical studies, different kinds of transition phrases have been identified and are provided for the corresponding scenarios (discover, rehearse, training competencies, etc). The generation of section titles also follows the pattern of the bridging texts generation.

To conclude, ACTIVEMATH is an adaptive eLearning system, which applies user modelling technique to adapt the assembly and presentation of learning object. It imposes constraints on these learning objects (transitions have to be omitted), which reduce the coherence of the assembled material and enforces a topic-oriented approach on the authored materials. In order to produce coherent documents (and to implement a document-centered approach for readers) the system's generates missing transitions between the assembled self-contained learning objects. The ACTIVEMATH approach neglects the mathematical structure and dependencies of these learning objects, though, these are already represented in the underlying XML representation. The adaptation routines are encoded in the system and solely draw on the user's learner models. Consequently, users have no control and can not guide the adaptation. The approach is focused on an educational scenario and can not be easily applied to other domains.

2.2.4. SWIM

The semantic wiki SWiM [Lan10, Lan08] facilitates the collaborative editing of mathematical content represented in OMDOC (see Section 2.1.5). As common in some wikis, each SWiM page is associated with a discussion page, providing a space for questions, answers, and comments about the respective topic. In [LK09], an argumentation ontology was introduced to annotate the semantics of these discussion entries and their relations.

Wiki pages and links can be tagged with ontological concepts. For this, SWiM integrates the most common semantic web ontologies, such as the Descriptive Ontology for

2. Preliminaries & State of the Art

Linguistic and Cognitive Engineering (DOLCE [MBG⁺03]), the Suggested Upper Merged Ontology (SUMO [PNL02]), friend-of-a-friend (FOAF [FOA]), learning object metadata (LOM [WG102]), SIOC Core Ontology Specification [BB07], and the Simple Knowledge Organization System (SKOS [IS09]), and supports users in importing any other RDFS [BG04] or OWL [SWM04] ontology.

Semantic wikis such as SWIM are usually not used for authoring whole documents but rather for creating self-contained modules (or concepts), thus, following a topic-oriented approach. These modules are displayed, edited, and maintained in the smallest unit of the wiki, i.e., a wiki page. Wikis do usually not provide a linear navigation through these pages but allow users to browse a network of concepts, which are highly interlinked by explicating their syntactic cross-references and, in semantic wikis, semantic interrelations. Based on self-contained modules, the content of SWIM can be easily assembled into user-specific documents drawing on existing workflows such as proposed by the ACTIVEMATH system (Section 2.2.3). The wiki can also import existing topic-optimised material. However, SWIM's content can not be easily converted into a coherent document such as a textbook or lecture notes. Also the import of such narrative documents remains rather challenging.

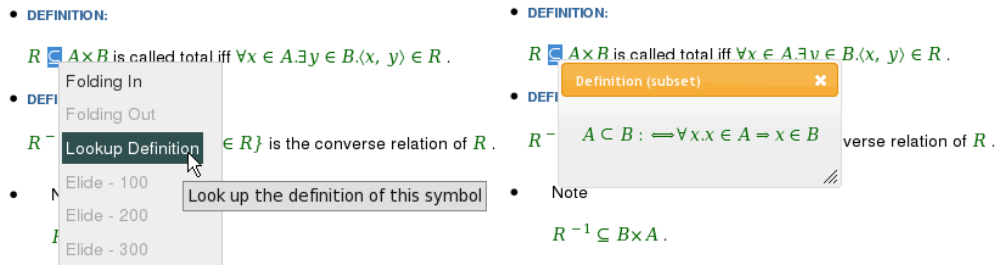


Figure 14.: Definition lookup with JOBAD, kindly provided by Christoph Lange [GLR09]

The semantic power of SWIM, which makes SWIM superior to other semantic wikis for mathematics, is the explication of the ontological concepts and interrelation defined by the OMDOC format and ontology. These are extracted from the OMDOC content [Lan09b] and explicitly stored as RDF triples [MM04]. In SWIM, one page can describe a mathematical statement or theory. The choice for the appropriate granularity is left to the user. However, the semantic navigation is richer if only small concepts are maintained. Following the paradigm of ‘little theories’ [FGT92], large theories should thus be partitioned into smaller ones that do not contain more statements than necessary. Ideally, each statement should be maintained on one wiki page and be linked to the theory it belongs to, i.e., by pointing to its home theory.

The SWIM project does not address the user-specific adaptation of document parts. Instead, one aspect of SWIM is the integration of interactive features to make the presented materials more *active* and on supporting users to interactively change the content and form of a wiki page. For this purpose, the JAVASCRIPT Framework JOBAD [GLR09] has been developed. Among others it allows users to retrieve definitions for the symbols in a page. Figure 14 illustrates the lookup of definitions with JOBAD [KGLZ09]. Users can right click on a notation (here the subset symbol) and choose the menu item ‘Lookup Definition’. On this selection a request is sent to the XML repository TNTBASE [ZK09], which retrieves the respective definitions from a corpus of OMDOC documents. Figure 15 illustrates the folding of notations. A right click on a notation opens a different menu, from which users can choose the ‘Folding In’ item. Doing so, the notation is enriched with details and $W_{pot}(R)$ is not displayed as $\frac{e^2}{2\pi\epsilon_0 R}$.

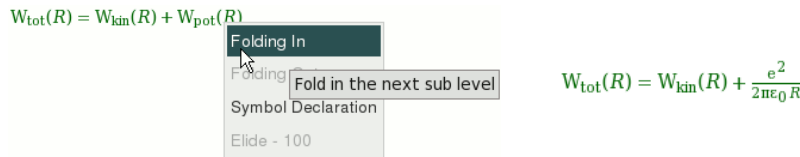


Figure 15.: Folding of notations with JOBAD, kindly provided by Christoph Lange

The SWIM project has also addressed the interactive adaptation of mathematical notations, e.g., supporting users in changing the amount of brackets. The formulae to the left of Figure 16 was rendered without any user configuration. Consequently, only the default brackets are displayed. For the formulae to the right, the user manually increased the precedence level, thus, more brackets than needed are inserted.

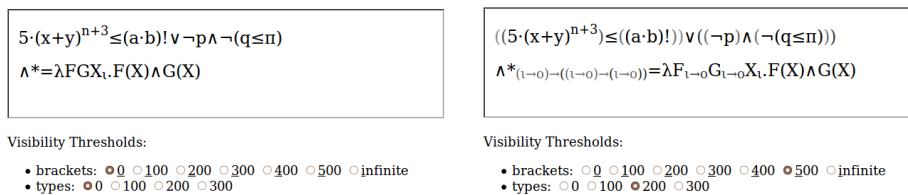


Figure 16.: Flexible elisions of notations, kindly provided by Christoph Lange [KLR07]

To conclude, SWIM exploits the semantic context of mathematical content to provide various interactive services, such as the enrichment of wiki pages with additional examples or the lookup of definitions. Its notation services are based on the rendering workflow described in Part II. SWIM considers what the author calls semantic context, e.g., by explicating didactic or rhetoric prerequisites of texts. However, the system does not focus on adaptive techniques and, thus, can not consider user constraints in any of its services. Moreover, SWIM follows the topic-oriented paradigm and can not support the export or import of coherent documents. It is thus limited to topic-optimised material. This work helps to extend the import facilities of SWIM: By modularising narrative documents into infoms, each infom can be represented as separate wiki page. Cross-references and transitions can be used to interlink these pages.

2.2.5. MATHDOX

The MATHDOX system [CCK+08] allows lecturers to augment their course material with interactive online assignments. Course materials are represented in the MATHDOX XML format and can be transformed to interactive mathematical web pages using the MATHDOX-PLAYER [Matc]. Lecturers can use these web pages to demonstrate algorithms, to test their students' mathematical skills, or to introduce new concepts with dynamic, on-screen calculations. Students have to enter the correct results for an exercise in the online material. They can provide several steps and use symbols from the palette of the MATHDOX formula editor [Matb]. When submitting their exercises, the result is sent to a computer algebra system (CAS) for verification. Finally, the students receive a feedback that either the exercise was solved correctly or, if wrong, which step is flawed.

The MATHDOX format extends DOCBOOK (Section 2.1.3) with OPENMATH [BCC+04] and the LeACTIVEMATH Exercise Language (LEAMEL [CCJS04]). OPENMATH supports the marking of mathematical expressions and LEAMEL captures the structure of mathematical exercises, which are represented as automaton with an inscribed problem solution strategy inscribed. To fine-tune reactions of a MATHDOX document to user-input,

2. Preliminaries & State of the Art

authors additionally need to embed programming instructions. For this, the MATHDOX format supports executable XML in JELLY [Jel]. Among others, JELLY supports conditional statements, calls to Java objects, and calls to web services. For example, JELLY instructions allow MATHDOX to randomise specific numbers in the exercises. The XML tag `<mdu:random var="a" minimum="1" maximum="9"/>` marks a randomisable variable a that can take a value between 1 and 9. The namespace initialises the Jelly component to randomise the numbers by calling a respective Java object. For more complex calculation, calls to a CAS are used.

Figure 17.: The SCORM package manager for MATHDOX [Mata]

MATHDOX provides a package manager [Mata] that allows users to combine MATHDOX documents into Shareable Content Object Reference Model (SCORM [Lea]) packages. The package manager maintains more than 1000 parametrised exercises in basic high school math, undergraduate calculus, linear algebra, and discrete mathematics. Authors can manually structure their course into different sections and explicitly select appropriate exercises from the corpus, either by directory or separately. When selecting a whole directory of exercises, these are automatically sorted by name in ascending order. The sorting is optional for separately selected exercises. Figure 17 shows the package manager, which includes one section with all the exercises of the *calculus/functies/kwadratisch* directory.

A recent, adaptive extension of MATHDOX [CCV10] focuses on context-sensitive, user-specific adaptations of interactive MATHDOX documents based on user models. The user model consists of *logistic information*, such as user name, affiliation, address, and student level, *knowledge information*, which is represented as mastery level attached to every node to the theory graph, and *mathematical context*, i.e., information about the mathematical context that the user has created by visiting the document and setting variables. In contrast to ACTIVEMATH, which solely considers didactic information about learning object, [CCV10] exploit the mathematical structure of the document. In particular, three graphs — a theory, a symbol, and a variable graph — are modelled for each document. The theory graphs support users to browse along the interconnection of large-scale structures of the mathematical doc-

uments, while the symbol and variable graphs are used to adapt the mathematical objects in the documents. A common context of the documents supports to provide consistent views of the documents in the system.

To conclude, MATHDOX is a mathematical eLearning system similar to ACTIVEMATH. In contrast to the interactive exercise module in ACTIVEMATH, the MATHDOX exercise system does not provide any user-specific adaptations. The underlying knowledge representation (DOCBOOK+OPENMATH) does not explicate the full semantic context of mathematical documents, only few dependencies are modelled. The inheritance structure of mathematical knowledge is not encoded. Although OMDOC-based systems offer a richer format for structuring mathematics than MATHDOX does, they do not offer the interactivity as provided by MATHDOX: A recent extension of MATHDOX addresses the interactive browsing of mathematical documents and models the semantic context in form of theory, symbol, and variable graphs. Users can use these graphs to explore materials in arbitrary order. During the user's interaction, his mathematical context is preserved in a user model and applied to the documents to initialise variables in the interactive formulae. This allows to repeat running examples and to provide a coherent flow of mathematical expressions in the documents. The adaptation in MATHDOX is a valuable supplement to this work. While [CCV10] support the instantiation of variables in formulae with user-specific values, this work focuses on the adaptation of mathematical notation as well as the content planning of mathematical documents. The integration of both approaches can offer valuable synergies and should be addressed in further research projects.

2.2.6. WELSA

The Web-Based Educational System with Learning Style Adaptation (WELSA [PBM09, WEL]) adapts courses to best suit the learning style of each student. These learning styles (or learning preferences) have been extracted from several learning style models to form a Unified Learning Style Model (ULSM [Pop09a]) and relate to perception modality, way of processing and organising information, as well as motivational and social aspects [PBT08]. The pedagogical goal of the system is to offer students *recommendation* regarding the most suited learning objects and learning paths for a predefined course, but to let the students decide whether they want to follow these guidelines or not [PBM09]. The adaptation provided by the system does not guide the creation of new documents from a collection of text fragments (e.g., as provided by ACTIVEMATH) but reorders and renders a given course by applying adaptive annotation techniques.

Analogous to other eLearning systems, WELSA manages *learning objects*, which represent any reproducible, reusable, and addressable digital learning resource and which have to omit cross-references and transitions [Pop09c]. The latter restriction can reduce the overall coherence of the course documents.

WELSA organises learning material hierarchically: each course consists of chapters, which consists of sections and subsections, the lowest level of subsections includes the elementary resources [PBT08] (the learning objects). The structure tree of the course (formed by chapters, sections, subsections) is stored separately from the actual content, i.e., the learning objects are solely referenced by (but not embedded into) the leaves of the structure tree. This representation provides the technical prerequisites for the automatic combination of learning objects as well as their reuse in different context.

Each learning object in WELSA is associated with metadata (maintained in a separate file). The metadata includes the instructional role, the media type, the level of abstractness and formality, and the type of competence. For this, the Dublin Core metadata standard [Dub] and Ullrich's instructional ontology [Ull08, Section 4.1] have been extended. The proposed

2. Preliminaries & State of the Art

descriptors include `LoType*`, which describes the instructional role according to Ullrich's instructional ontology, `isFor` and `inverseIsFor`, which relate an auxiliary learning object to the fundamental one it completes, `requires` and `isRequiredBy`, which relate a learning object to its prerequisite, `isA` and `inverseIsA`, which relate a learning object to its parent concept, `isAnalogous`, which relates two equivalent learning objects with similar content but differing in media type or level of formality [PBT08]. Note that metadata is only associated with learning objects, thus relation between aggregated, coarse-grained items of a course structure are not represented. Moreover, the WELSA's adaptation approach fails if all learning objects are connected via these relations. Consequently, the authoring of semantically self-contained (non-related) learning objects is considered a best practice [Pop09c].

The screenshot displays the WELSA AI course interface. At the top, there is a blue header with the 'Welsa' logo on the left and 'Account Logout' and 'Welcome Student1' on the right. Below the header, navigation links for 'Home | Courses | Forum | Chat' are visible. The main content area is titled 'Constraint Satisfaction Problems' and 'Solving a CSP'. On the left, a sidebar lists 11 chapters, with '5. Constraint Satisfaction Problems' highlighted. The main content area shows a list of sections: 'Introduction', 'Definitions', 'Graphical example of domain-consistent constraint network', 'Example of domain-consistent constraint network', 'Example of achieving arc consistency', and 'Algorithms for achieving network consistency'. Each section has a blue arrow icon on the left and a star icon on the right. The 'Introduction' section is expanded, showing a paragraph of text: 'Although backtracking is usually a substantial improvement over generate-and-test, it still has various inefficiencies that can be overcome. Consider, in the schedule example, variables C and D . The assignment $C=4$ is inconsistent with each of the possible assignments to D since $D_D=\{1, 2, 3, 4\}$ and $C < D$. In the course of the backtrack search this fact can be rediscovered for very many different assignments to A, B and possibly E . This inefficiency can be avoided by simply deleting 4 from D_C , once and for all. This idea is the basis for the consistency algorithm.'

Figure 18.: The WELSA AI course, kindly provided by Elvira Popescu [WEL]

WELSA applies an implicit, dynamic user modelling approach. The system (in particular the course player) observes the student's interaction by monitoring behavioural indicators such as navigational, temporal, and performance indicators. These indicators as well as the corresponding metadata of learning objects is used to infer learning preferences as specified by ULSM. This is done by applying a set of inference rules [Pop09b] for the computation of ULSM preferences onto the available data, i.e., the indicators and resources metadata: For example, a visual preference is inferred if students spend a high amount of time on contents

with graphics, images, and video. The inferred learning preferences are stored in the student’s user model. A possible model for a student could indicate that he prefers visual content, concrete, practical examples, serial approach, reflexive observation. Alternative, it could specify that the students likes verbal content, abstract concepts, and generalizations, global approach, active experimentation. For the adaptation, the user models are matched to metadata of the learning objects, which are rearranged and annotated, respectively.

Figure 18 provides a page of an artificial intelligence course from the WELSA demo [WEL] including a sequence of learning objects, which can be expanded and collapsed by the students. Each learning object is associated with an icon reflection its instructional role, e.g., example, definition, or algorithm. The arrangement and presentation of the learning objects adopts to the user model of the student. However, no learning objects are omitted to allow students to explore all resources of the lecture. Instead, colours are used: Recommended learning objects have a green title, less preferred ones have a dimmed light grey title.

The student’s preferences to generate the page in Figure 18 are *visual* and *example-oriented*, thus, all graphical examples are recommended: The page includes two equivalent examples for ‘domain-consistent constraints network’, a graphical and a text example (see Figure 19). Only the graphical example is recommended (thus has a green title). Since no equivalent graphical example for ‘achieving consistency’ exists, the text example is recommended. On contrast, for the preferences *abstract* and *global*, WELSA would recommend algorithms and definitions and mark the examples as less preferred.

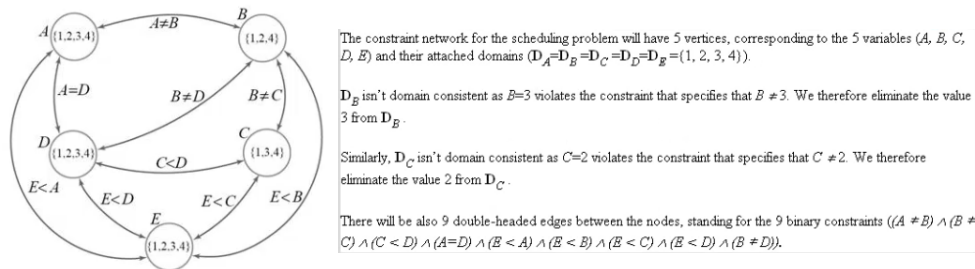


Figure 19.: Two equivalent examples, kindly provided by Elvira Popescu [WEL]

To conclude, WELSA follows a non-intrusive learning strategy. Learning objects are only sorted, but not replaced, removed, or inserted. A focus is placed on making recommendations in form of visualisations and orderings. Content planning services such as varying level of detail, expertise, formality, or multilingual presentations are not supported. The system has not been developed with a focus on mathematics, though its content (for the Artificial Intelligence course) is based on mathematics. An explicit representation of the underlying mathematical structure, dependencies, and properties could improve the system’s control over its content. Respective extensions require the introduction of new ways to guide the adaptation. Currently, the *requires* and *isRequiredBy* relations are used by authors to prevent the ordering of resources: If two learning objects are related in either relation, the system does not change their order, regardless of the learner preferences.

2. Preliminaries & State of the Art

2.2.7. Summary

System	PLATΩ	CNX	ACTIVEM.	SWIM	MATHDOX	WELSA
document-centered	yes	yes	no	no	no	no
topic-oriented	no	yes	yes	yes	yes	yes
exploitation of semantic context	yes	no	yes	yes	yes	yes
exploitation of user context	yes	yes	yes	no	yes	yes
exploitation of narrative context	no	no	no	no	no	no
focus on MKM	yes	yes	yes	yes	yes	no
web-accessible	no	yes	yes	yes	yes	yes
adaptation of notations	yes	yes	yes	yes	no	no
reordering of document parts	yes	no	yes	no	no	yes
vary level of detail, expertise or formality	yes	no	yes	no	yes	no
multilingual presentations	no	no	yes	no	no	no

The table above recapitulates the discussed systems. PLATΩ and CONNEXIONS follow a document-centered approach and allow authors to write coherent documents. All other systems focus on self-contained learning objects (or wiki pages) and are limited to a topic-oriented approach. ACTIVEMATH aims at bridging the topic-oriented and document-centered world by generating transitions between the self-contained learning objects. The quality of these transitions, however, can not compete with manual written texts (Section 7.1). CONNEXIONS aims at a topic-oriented infrastructure by supporting users to assemble document modules into collections. However, since the CONNEXIONS modules are very large, they do not comply with the common understanding of topics.

ACTIVEMATH and WELSA only exploit didactic metadata for their adaptations services, mathematical relations, properties, and structures are neglected. In return, MATHDOX and PLATΩ exploit the mathematical structure and dependencies of content but do not consider didactic aspects. CONNEXIONS stores user preferences (e.g., in form of lenses) but does not consider the semantic context of its content. SWIM exploits the semantic context but does not process information on users. None of the system exploits the narrative context of documents for their services.

The adaptation of notations is addressed by PLATΩ, CONNEXIONS, ACTIVEMATH, and SWIM. The limitation of the former three approaches is discussed in Part II. The approach in SWIM is collaborative work with the author of this thesis. Note that PLATΩ is not considered in further discussions as it does not apply XML technologies for its adaptation, but rather information management techniques. It can thus not be directly compared with this work. From all content-oriented systems (i.e., systems, which draw on markup techniques), only ACTIVEMATH and WELSA provide user-specific content planning services. They will thus be compared to the proposed content planning approach in Part III.

Part II.

Adaptation of Mathematical Notations

3. Introduction

Acknowledgement: The results presented in this part are collaborative work with Michael Kohlhase, Christoph Lange, Normen Müller, and Florian Rabe. They have been published in [MK09a, KLM⁺09, KMR08].

Over the last three millennia, mathematics has developed a complicated two-dimensional format for communicating formulae [Caj93, Wol00]. Structural properties of operators often result in special presentations, e.g., the scope of a radical expression is visualised by the length of its bar. Their mathematical properties give rise to placement (e.g., associative arithmetic operators are written infix), and their relative importance is expressed in terms of binding strength conventions for brackets. Changes in notation have been influential in shaping the way we calculate and think about mathematical concepts, and understanding mathematical notations is an essential part of any mathematics education. All of these make it difficult to determine the functional structure of an expression from its presentation.

Content Markup formats for mathematics, such as OPENMATH [BCC⁺04] and Content-MATHML [ABC⁺08], concentrate on the functional structure of mathematical formulae, thus allowing mathematical software systems to exchange mathematical objects. For communication with humans, these formats rely on a presentation process that transforms the content objects into the usual two-dimensional form used in mathematical books and articles. Many such presentation processes have been proposed, and all have their strengths and weaknesses. This work conceptualises the presentation of mathematical formulae as consisting of three components: the context-dependent *selection* of appropriate rules (called **notation definitions**) for presentation, the two-dimensional *composition* of visual sub-presentations to larger ones, and the *interaction* with rendered content.

Most current presentation processes concentrate on the relatively well-understood composition aspect. They control the notation selection by relying on simple metadata, which do not fully represent the context of a formula. Their output is mostly static and meant to be read, not to be interacted with, except for some applications that allow for changing notations afterwards.

In this situation, the author and her colleagues [KLM⁺09, KMR08] proposes to encode the presentational characteristics of symbols (their context, their compositional behaviour, and information accessible by interactive services) declaratively in *notation definitions*, which are part of the representational infrastructure and consist of **prototypes** (patterns that are matched against content representation trees) and **renderings** (that are used to construct the corresponding presentational trees).

This work proposes an elaborated mechanism to collect notation definitions from various sources and to guide the selection of appropriate renderings according to the user's notation preferences. This brings the separation of *function* from *form* in mathematical objects and assertions in mathematical representation formats to fruition on the document level. This is especially pronounced in the context of dynamic presentation media (e.g., on the screen), for which we can now realise *adaptable documents* that adapt to a user's notation preferences. Note that since notations have been reified, we can now devise a flexible management process for notations. For example, we can capture the notation preferences of authors, aggregators, and readers and adapt documents to these.

3. Introduction

3.1. Introduction into Mathematical Notations

Acknowledgement: *The examples in this section are taken from a survey on mathematical notations conducted with a voluntary group of students at Jacobs University Bremen. They have been discussed with Michael Kohlhase, Christoph Lange, Normen Müller, and Florian Rabe and have partially been published in [KLM⁺09, KMR08]. The author would like to thank André Nies from the University of Auckland, NZ, for his feedback on this section.*

Looking at the history of the mathematical language, we can observe a tendency to increased formalisation: Early publications include more natural language as many notations had not yet been developed. For example, around 250 AD the Greek mathematician Diophantus could not draw on symbols such as $=$ for equality, $<$ and $>$ for less/greater than, as well as \leq and \geq for less/greater or equal than. During his productive period, number theory did not yet provide these notations. Throughout the years, more and more mathematical symbols and notations have been added. Nowadays, mathematical language is a mixture of highly specialised notations and natural language, where notations make up 30-60% depending on the type of mathematical text. Mathematical notations have become an essential part of mathematical language, similar to musical notation systems, which are fundamental for the creation and communication of compositions. Modern science is inconceivable without a precise notation system: Notations ease communication of mathematical practitioners as they *reify* mathematical ideas into compact and precise forms, which, conversely, have to be interpreted by the recipients.

Nevertheless, the increased formalisation and need for interpretation of mathematical language also bears its challenges: Mathematical notations can complicate communication and acquisition processes, in particular, for less experienced users. This is due to the fact, that mathematical notations are *context-dependent* and can considerably vary among different communities and individuals. Consequently, notations can cause ambiguities and misunderstandings and, thus, may hamper learners and collaboration. Even though notations are an essential part of mathematical texts, we are still not able to fully control them, e.g., to adapt them to a reader's background and preferences. In the following, mathematical notations and their variations are illustrated.

Understanding Mathematical Notations Mathematical notations denote mathematical concepts, i.e., the objects we talk and write about when we do mathematics. This includes rather simple objects like numbers, functions, triangles, matrices, and more complex ones such as vector spaces and infinite series. Mathematical notations are no separate entities but highly interdependent. In mathematics we speak of *notation systems*, i.e., collections of notations that depend on each other. Consequently, the choice of a specific notation for a concept requires to use notations from the same system for all other concepts. For example, if we look at the notation for *subset* and *proper subset*, we can use \subseteq and \subset versus \subset and \subsetneq . In the first combination, \subset denotes the *proper subset*, while in the second combination it denotes *subset*. Consequently, when adapting the notation of subset from \subseteq to \subset , we also need to change the notation for proper subset from \subset to \subsetneq . Otherwise, we end up with the same notation for two different mathematical concepts, which eventually destroys the semantics of the mathematical formula.

Mathematical Communities and their Notations We can observe mathematical communities, which prefer different notation systems. For example, Figure 20 provides an example of two notation systems in the area of sentential logic: the core of Jan Łukasiewicz's

Mathematical Concept	Conventional Notation	Polish Notation
Negation	$\neg\varphi$	$N\varphi$
Conjunction	$\varphi \wedge \psi$	$K\varphi\psi$
Disjunction	$\varphi \vee \psi$	$A\varphi\psi$
Material conditional	$\varphi \rightarrow \psi$	$C\varphi\psi$
Biconditional	$\varphi \leftrightarrow \psi$	$E\varphi\psi$
Sheffer stroke	$\varphi \psi$	$D\varphi\psi$
Possibility	$\diamond\varphi$	$M\varphi$
Necessity	$\square\varphi$	$L\varphi$
Universal Quantifier	$\forall\varphi$	$\Pi\varphi$
Existential Quantifier	$\exists\varphi$	$\Sigma\varphi$

Figure 20.: The conventional and polish notation system

notation for sentential logic [Łuk67] to the right and the conventional notation, which was developed in the 1970s and 80s, to the left.

Different notation systems can also be observed when looking at Russian and Western mathematical journals. Partly, the notations between Russian and Western researchers differ as they build on different concepts. However, also the overlapping concepts used by both groups are denoted with very different sets of notations. Moreover, even if Western researchers used and defined concepts solely used by Russians, they would denote them very differently staying conform to the type of notations in their systems.

Mathematical areas are further divided into schools that originally evolved based on individual styles of single mathematicians. For example, Chaitin [Cha87] and Li/Vitani [LVon] use different notations to denote the same concepts (plain and prefix free complexity) in the field of algorithmic information theory (AIT): Chaitin uses $K(x)$ and $H(x)$, while Li/Vitany use $C(x)$ and $K(x)$.

Contextualisation of Mathematical Notations Various researchers have provided examples in which mathematical expressions are presented differently depending on the context they are used in. [SW06] introduce possible reasons for multiple notations of the same mathematical concept, namely area of application, national conventions, level of sophistication, the mathematical context, and the historical period. In contrast to the five reasons in [SW06], [MLUM05] distinguishes four context categories that influence the adaptation of notations, namely language, different patterns of the argument, the author's style, and notations of the same collection. [MUGL09] emphasis on the culturally communities to which a learner, teacher, or author can belong to: language, country, region, and community of practice [Wen05b], e.g., a group of chemistry or electrical engineering students. In the following, alternative notations are illustrated according to the context dimensions: *individual (author) style, level of expertise, display style, language, and area of application*.

Individual Styles. We can observe individual styles that differ within schools or communities. For example, some mathematical authors are more formal, while others prefer to include more natural language terms. For example, consider the mathematical statement “Let n equal 2 times m square. Choose a natural number k so that k is less than or equal to n .” in contrast to the more compact and formal “Let $n = 2m^2$. Choose a number $k \leq n$.”. Some mathematicians feel that the latter is more easier to read, while others reject it, as they believe that symbols should not be part of the prose text.

3. Introduction

Level of Expertise. Mathematicians gloss over parts of the formulae, e.g., leaving out arguments, if they are non-essential, conventionalised, or can be deduced from the context. Indeed, this is part of what makes mathematics so hard to read for beginners, though, also so efficient for the initiates. The below examples illustrate notations that can be classified based on different levels of expertise of readers and authors.

- While $a \div b$ is mostly used in elementary school, a/b and $\frac{a}{b}$ are used in higher education [SW06].
- To denote the logarithm function $\log_{10}(x)$ alternatives such as $\log(x)$, $\log x$ or $\lg x$ can be used.
- The natural logarithm is denoted by either $\ln y$ or $\log_e y$. Teacher may choose to use the latter notation to lead to the former.
- Experienced users refer to $\sin^y x$ rather than $(\sin x)^y$ to denote the sinus function to the power of y .
- Experienced users omit the times operator, if it can be deduced from the context, e.g., writing ab rather than $a * b$.
- Authors choose to insert brackets that are not necessarily needed or to omit them if their readers are more experienced [KLR07].
 - $x^2 - 3yx + 1$ can be illustrated with $x^2 + (-3y)x + 1$ [NW03].
 - $ax + y$ is actually $(ax) + y$, since multiplication binds stronger than addition,
 - $5 + x * y^{n+3}$ rather than $5 + (x * (y^{(n+3)}))$.

Display Style. Authors select notations for a specific output format.

Text Output	Display Output
$\sum_{k=1}^n \dots$ $\bigcup_{k=1}^n \dots$	$\sum_{k=1}^n \dots$ $\bigcup_{k=1}^n \dots$

Language and Cultural Differences. We can also think of notations that vary among different cultures or languages. For example, for decimal numbers Germans use a comma where English use a decimal point: 4,5 in German is equivalent to 4.5 in English. In contrast, for structuring large numbers the German notation uses a point (1.000), while in English a comma is used (1,000).

Another example is the binomial coefficient, which is denoted with C_n^k in French/Russian speaking countries and with $\binom{n}{k}$ in German/English speaking countries [MUGL09]. Alternative notations are $\mathcal{C}(n, k)$, ${}_n\mathcal{C}^k$, ${}^n\mathcal{C}_k$, and ${}_nC_k$ as well as system-specific notations, such as $\text{binomial}(n, k)$ in MAPLE and $\text{Binomial}[n, k]$ in MATHEMATICA [NW03].

Area of Application. The same mathematical concept can be expressed by different notations depending on the area it is used in. For example, a mathematician uses the symbol i to denote the *imaginary unit* $\sqrt{-1}$. In contrast, an electrical engineer uses j to avoid confusion with the symbol I for electric current.

Another example for the contextualisation by different area (and language) refers to the *natural numbers*, which are defined and presented differently in various areas: A natural number is either an element of the set $\{1, 2, 3, \dots\}$ (the *positive integers*) or an element of the set $\{0, 1, 2, 3, \dots\}$ (the *non-negative integers*). Table 1 illustrates different notations for both concepts, which were partially gathered from [Nat09] and partially identified during discussions with colleagues/students.

Notation	Explanation of use
$\mathbb{N}^+, \mathbb{N}^*$	notation for positive integers in English
\mathbb{N}	notation for non-negative integers in English and in set theory in Germany
\mathbb{N}	notation for positive integers in number theory in Germany
\mathbb{N}_0	notation for non-negative integers in number theory in Germany
\mathbb{N}^*	The notation $*$ is standard for non-zero or rather invertible elements, i.e., elements that can 'undo' the effect of combination with another given element.
\mathbb{W}, \mathbb{P}	Some authors who exclude zero from the naturals use the term whole numbers, denoted by \mathbb{W} , for the set of non-negative integers. Others use the notation \mathbb{P} for the positive integers.
\mathbb{N}	Russia notation for natural numbers (defined as positive integers)
\mathbb{Z}^+	Russian notation for non-negative integers
ω	Set theorists often denote the set of all natural numbers by a lower-case Greek letter omega. When this notation is used, zero is explicitly included as a natural number (thus denotes non-negative integers).

Table 1.: Notations for non-negative and positive integers

3.2. Representation of Mathematical Notations

To support the adaptations of mathematical notations in online documents, mathematical objects are represented in MATHML [ABC⁺08] and OPENMATH [BCC⁺04].

OPENMATH Representation	MATHML Representation	Presentation
<pre> <OMOBJ> <OMA> <OMS cd="combinat1" name="binomial"/> <OMV name="n"/> <OMV name="k"/> </OMA> </OMOBJ> </pre>	<pre> <mrow> <mo fence="true"></mo> <mfrac linethickness="0"> <mi>n</mi> <mi>k</mi> </mfrac> </mrow> </pre>	$\binom{n}{k}$

Figure 21.: OPENMATH and MATHML representation of the binomial coefficient.

Figure 21 provides the OPENMATH and MATHML representations of the binomial coefficient. The OPENMATH expression on the left captures the functional structure of the expression by representing it as the application (using the OMA element) of the binomial coefficient function (represented by an OMS element) applied to two variables (OMV). Note that the `cd` and `name` attributes characterise the binomial function by pointing to a definition in a **content dictionary** (CD) [OMC], a specialised document that specifies commonly agreed definitions of basic mathematical objects and allows machines to distinguish the meaning of included mathematical objects. In contrast to this, the Presentation-MATHML expression in the middle marks up the appearance of the formula when displayed visually (or read out aloud for vision-impaired readers): The formula is represented as a horizontal row (`mrow`) of two stretchy bracket operators (`mo`) with a special layout for fractions (`mfrac`), where the line is

3. Introduction

made invisible by giving it zero thickness. The numerator and denominator are mathematical identifiers (`mi`). The aim and strengths of the two formats are complementary: OPENMATH expressions are well-suited for information retrieval by functional structure and computation services, while MATHML is used for display: MATHML-aware web-browsers will present the expression to the right as $\binom{n}{k}$.

<pre> <m:semantics> <m:mrow id="top"> <m:mo></m:mo> <m:mfrac linethickness="0"> <m:mi id="left">n</m:mi> <m:mi id="right">k</m:mi> </m:mfrac> <m:mo></m:mo> </m:mrow> ... </pre>	<pre> <m:annotation-xml> <om:OMOBJ> <om:OMA xref="top"> <om:OMS cd="combinat1" name="binomial" /> <om:OMV name="n" xref="left"/> <om:OMV name="k" xref="right"/> </om:OMA> </om:OMOBJ> </m:annotation-xml> </m:semantics> </pre>
--	--

Figure 22.: Parallel markup – combining OPENMATH and MATHML

In order to combine both markup aspects, MATHML allows *parallel markup* [ABC⁺08] with fine-grained cross-references of corresponding sub-expressions. Figure 22 provides the parallel markup for the example in Figure 21: The `semantics` element embeds a Presentation-MATHML expression and an `annotation-xml` with the respective OPENMATH expression. The `id` and `xref` attributes specify corresponding sub-terms. An application of this would be that a user can select a sub-term in the Presentation-MATHML rendered in a browser, so that a context menu option could send the corresponding OPENMATH sub-expression to, e.g., a computer algebra system for evaluation, simplification, or graphing [GLR09].

Parallel markup only provides a one-to-one mapping between OPENMATH and MATHML expressions. However, in mathematics we have to deal with multiple alternative notations that denote the same mathematical object, such as various presentations of the binomial coefficient C_n^k or $\binom{n}{k}$. Moreover, we can also select a MATHML representation that is more suited for further processing, such as by Braille or screen readers. While the former reader supports visually impaired users, the latter is of use to any learner as it supports to read notations out loud and thus to foster the user’s understanding. Figure 23 provides two alternative MATHML representations for the binomial coefficient $\binom{n}{k}$. The expression to the right is also referred to as *canonical representation* [AM06] and can more easily be accessed by Braille readers. A mathematical Braille translator, such as [ASFM07], will recognise the MATHML expression to the left as fraction $frac(n, k)$, since presentation attributes such as `linethickness` are ignored. One could argue that Braille translators should not rely on Presentation-MATHML (but rather OPENMATH or Content-MATHML), as Presentation-MATHML only provides a layout tree and no insights on the semantics of the notation. This works aims at supporting existing implementations and thus provides an adaptable selection between alternative MATHML expressions tailored to the needs and preferences of the users as well as further processing services.

In order to automatically adapt mathematical notations, we need to be able to vary the displayed Presentation-MATHML. This is usually done by parametrising the process by which notations are generated from a given content representation. This approach represents the mappings between an OPENMATH expression and all alternative MATHML representations. Conceptually, these mappings represent mathematical notation practices as they explicate the


```

<mrow>
  <mo></mo>
  <mfrac linethickness="0">
    <mi>n</mi>
    <mi>k</mi>
  </mfrac>
  <mo></mo>
</mrow>
<mrow>
  <mrow><mo></mo>
  <mrow>
    <mtable>
      <mtr><mtd><mi>n</mi></mtd></mtr>
      <mtr><mtd><mi>k</mi></mtd></mtr>
    </mtable>
  </mrow>
  <mo></mo></mrow>
</mrow>

```

Figure 23.: Two valid MATHML expressions.

choice of mathematical notations of the user. In order to make adaptation context-aware, a conversion workflow has to be implemented, which takes notation practices and concrete context as input and adapts the respective notation for the user.

3.3. State of the Art

This approach focuses on the adaptation of notations in web documents. However, we must not neglect the plethora of proof assistance systems, where support of specific mathematical notations is a major concern. In the following, techniques used in ISABELLE, MATITA, and Ω MEGA are discussed.

The theorem prover ISABELLE [Pau05, NPW02] provides one of the most advanced notation frameworks among all proof assistants. It offers type-setting facilities of formulae for \LaTeX and supports the declaration of the notations for symbols as prefix, infix, postfix, and mixfix as well as the translations between different notations [Pau05]. However, [KLR07] argue that the ISABELLE model is not directly applicable to presentation of OPENMATH objects, as ISABELLE crucially depends on the assumption that terms have a fixed arity. In OPENMATH objects, applications can have flexible arities depending on the operator, i.e., application nodes can have different numbers of children, even if they coincide in the first child (the function). This is used to model a flexary addition function, i.e., a function that takes any number of arguments, e.g., $\text{\@}(add, 1, 2, 3, 4)$. In ISABELLE, we would have to model addition as a binary, associative operator, and the term above as $plus(plus(1, 2), plus(3, 4))$ which would have the same presentation $1 + 2 + 3 + 4$, but a different structure. Similarly, ISABELLE only supports unary binding constructions.

The interactive theorem prover MATITA [ACG⁺06] provides a bidirectional conversion between the presentation-oriented encoding in MATHML and the fully formal representation of the system. Thus, in contrast to this work, MATITA handles fully formal representations and is able to disambiguate presentation markup. The notational system of MATITA [Cla09, PZ06] is based on three different languages for the representation of formulae and proofs: fully formal representations, content, where Content-MATHML and OMDOC is used, and presentation, where Presentation-MATHML within a simple layout language called BOXML [Pad04] is used. The rendering workflow is based on a predecessor system called HELM [hHELoM]. HELM implements a multi-stage process of generating XSLTs from meta-XSLTs to convert COQ [Tea04] into Content-MATHML and further into Presentation-MATHML. When the HELM project started, support for MATHML in web browsers was insubstantial, resulting in the development of a new MATHML rendering engine as a widget [Pad04, Vie] for the Gimp Toolkit (GTK). Following this approach, MATITA does not provide an web-accessible interface and, thus, doesn't produce XHTML. Instead, content is rendered into BOXML,

3. Introduction

the presentation format of the MATITA widget, and Presentation-MATHML. To conclude, MATITA and HELM provide an elaborated rendering workflow but do not focus on the management of mathematical notations on the web. Moreover, they do not allow users to guide the rendering process with individual context parameters.

The interactive mathematical mediator $\text{PLAT}\Omega$ [AFNW07] integrates the scientific text editor TeX_{MACS} [TeX05] with the Ω MEGA system [ABD⁺06, HKK⁺96]. The user interface supports a document-oriented authoring approach and considers the notation contexts of documents as a dynamic parameter in separation to the document's content. By processing the marked up notations in a document, the $\text{PLAT}\Omega$ system is able to extract the semantics of the mathematical notations contained in the document and, thus, can model the author's notation practice. Moreover, the document can be automatically adapted in the case of notational changes to the author's preferences. In [WM07], an extension of the mediator is proposed, which aims at supporting communities of practices (Section 10.3.1). The individual notation practice of authors can be compared and used to identify communities that share specific notation preferences. Once having identified communities, $\text{PLAT}\Omega$ can actively support the community members, e.g., by suggestion the community's standard notation, by notifying about conflicts or even by translating documents between communities.

Unfortunately, literal reuse from proof assistance systems will not work as their rendering workflows are commonly tied to the systems' internal data structures. In the following, related work on the conversion from Content-MATHML and OPENMATH to the web-accessible Presentation-MATHML format is discussed.

OPENMATH/ Content-MATHML is commonly rendered by implementing one XSLT template [Kay07] per symbol, which specifies how to transform this symbol to the output format, e.g., to Presentation-MATHML. Appropriate templates for the arguments are usually applied recursively. To save authors from the tedious, error-prone task of writing similar templates for every symbol and notation variant, different facilitations have been invented. The probably earliest examples are the presentation architectures in OMDOC 1.0 [Koh00] and Naylor's&Watt's OPENMATH conversion [NW03]. Both supply XML-based notation definitions that can be transformed into XSLT-conversion stylesheets based on meta-stylesheets.

[NW03] propose meta (XSLT) stylesheets that utilises a MATHML-based markup of arbitrary notations in terms of their content and presentation. Based on the manual selection of users, user-specific XSLT stylesheets [Kay07] are generated and can be used to adapt the notations in a document. A one-dimensional context annotation of content expressions is introduced, which allows to intensionally select an appropriate notation definition. The authors claim that users also want to delegate the styling decision to some defaulting mechanism and propose the following hierarchy of default notation definitions (from high to low): command line control, input documents defaults, meta stylesheets defaults, and content dictionaries.

[MLUM05] emphasise the need for maintaining uniform and appropriate notations in collaborative environments, in which various authors contribute mathematical material. They address this problem by providing authors with respective tools for editing notations as well as by developing a framework for the consistent, user-driven presentation of symbols. In particular, they extend the approach of [NW03] by an explicit language markup of mathematical content expressions and prioritise (from high to low) the different notation styles as follows: individual style, group, book, author or collection, and system defaults .

[KLR07] present an infrastructure for declarative notation definitions building on ideas from OMDOC 1.2 [Koh06] and the presentation system of the ISABELLE theorem prover [NPW02]. By incorporating functionality for flexary applications and binders, the authors apply these ideas to OPENMATH and Content-MATHML, and have extended the ISABELLE's precedence-based elision algorithm [Pau05] with general flexible elision func-

tionality. The authors have also compared the infrastructure in OMDOC [Koh00, Koh06] and XSLT-based approaches in general. Programming XSLT templates is criticised as being tedious and error-prone as well as non-declarative, which is ill-suited as definitional mechanism and are tied to the presentation procedure of the XSLT programming language. Although authors only have to write XSLT in exceptionally complex cases, there are significant disadvantages with an XSLT-based rendering implementation. [Koh09] emphasises that the problem with the rendering approach for OMDOC “*was not using XSLT, but using XSLT to compile OMDOC presentations to XSLT and then using XSLT to collect notations and to bind these into large situation-specific XSLT files. This is flexible, but not ideal to debug, and not efficient*”. In particular, dynamic, context-sensitive selection of notation for different parts of a document can not be supported easily.

[Lib07] present a pattern-based encoding of mathematical notation converters from OPEN-MATH and Content-MATHML to Presentation-MATHML, which can be guided by user-dependent preferences. The reference implementation for this approach is based on XSLT and thus faces the same fundamental problem as the specification in OMDOC 1.2 [Koh06]. In XSLT, one cannot directly render a document using any convenient syntax (be it declarative or pattern-matching), but first needs to generate the respective XSLT stylesheets. Moreover, XSLT is considerably hard to debug [Lan09a].

In [KMM07b], the redefinition of documents towards a more dynamic and living view has been initiated. The narrative and content layer was explicated and the document model was extended by a third dimension, i.e., the presentation layer. An extensional markup of the notation context of a document was proposed, which facilitates users to explicitly select suitable notations for document fragments. These extensional collections of notations can be inherited, extended, reused, and shared among users. For the notation approach presented in the next sections, the proposals in [KLR07, KMM07b] have been reengineered and extended.

3.4. Requirements Specification

Acknowledgement: This section is based on interviews with researchers and instructors of mathematics and theoretical computer science from New Zealand, Canada, and Europe.

Informal discussions with an international group of instructors and researchers of mathematics and theoretical computer science have revealed valuable requirements for the adaptation of mathematical notations as well as other notation services. In the following we outline the core ideas from the discussions.

Some participants disagreed that notations are context-dependent and vary frequently. They are convinced that mathematical notations are *universal* and *standardised* within a specific mathematical community. Alternatives are hardly used and would not be accepted. The author believes that these arguments do not conflict with the intention of her work. With ‘specific mathematical community’, the participant referred to a mathematical *sub-community* (e.g., a particular area or field). Concepts for which notations are standardised in one mathematical field, might be denoted differently in another one. Adaptation can help to bridge these differences.

3.4.1. Support for Standard and New Notations

One of the participants emphasised that one should distinguish between two kinds of mathematical knowledge, *standard mathematical knowledge* and *new mathematics*. The former denotes already discovered, well-known mathematical concepts and the latter denotes all

3. Introduction

knowledge that still has to be discovered/constructed. This observation relates to Godfrey Hardy’s reflection on mathematics [Har92]. Hardy distinguishes *elementary* from *pure mathematics*, where the former includes all school, most universities, and, in particular, applied mathematics and their different cultures. The latter subsumes mathematical research, the innovation and creation of new mathematical understandings.

In discussion with a participant, the author tried to explore the challenge of constructing new notations. The participant sees the main challenge (a basic skill) of a mathematicians in finding good notations. Objects do not really differ from their notations: “*While finding good notations you also construct the object*”. He distinguishes two types of notations, *description of mathematical objects* (group A) and *values or functions* (group B), but emphasises that these two types are interdependent and shouldn’t be viewed separately.

Examples for group B are $\sqrt{x^1+1}$ or $\frac{1+\sqrt{5}}{2}$, e , π , and $\binom{n}{k}$. According to the interview partner, a notation belongs to group A, if the value of the whole notation is more than of the single pieces. Often, these pieces are simple objects that in combination create complex structures (see ‘Kolmogorov Complexity’ [LVon]). To illustrate group A, the participant provided the following examples from group theory.

- $\mathbb{Z}/_p\mathbb{Z} \wr \mathbb{Z}$ — a group, \wr is the wreath product
- $GL_n\mathbb{Z}$ — invertible $n \times n$ integer matrices
- $Aut(SL_2(\mathbb{Z}))$ — Aut is the Automorphism Group, SL_2 is a matrix with determinant 1, and \mathbb{Z} the set of integers

Group A highlights that some mathematical notations are already very close to content markup. They describe the object and provide insight in how the object is constructed. Explicating the logical structure of complex notations allows us to identify the (simpler) constituents of which these expressions are build up of. Adaptation can be provided for these group of notations, if alternative, user-preferred notations of the constituents exists.

3.4.2. Consistency and Quality of Notations

In further discussions, the consistent presentation of mathematical symbols and formulae in a document were underlined: “*A system should warn you, if you want to reuse an already used notation or letter for a different concept. This is a very critical aspect as we make use of a lot of symbols but only have a limited amount of notations/letters. Choosing good notation is essential for mathematical writings*”. The argument emphasises two aspects: the *consistent* use of notation and the use of *good* notations. To illustrate the difference between good and bad notations, the following examples were mentioned.

- The notations $f(x)$ and $f(1/x)$ is a general example: The function f could typically denote some function from analysis. The second notation is too complicated, the function should be redefined, e.g., as denoted by the first notation.
- The notations $J^A(e)$ and $\varphi_e^A(e)$ are examples from computability [Nie09]. $J^A(e)$ is the result of a computation of a fixed universal Turing machine with oracle A and input e . One of many ways to obtain a machine is to list all oracle Turing machines and evaluate the e^{th} machine at input e . The corresponding function is $\varphi_e^A(e)$. Both notations, $J^A(e)$ and $\varphi_e^A(e)$, have the same semantics. $J^A(e)$ is to be preferred because it doesn’t double e and because the selected notations should not remind the reader of the way a Turing machine is obtained traditionally.

- The notations $\varphi_e^A(x)[s]$ and $\varphi_{e,s}^{A,s}(x)$ mean that we run the e^{th} machine with oracle A on input x but only for s steps. The first notation is preferable because it also distinguishes e and s typographically.

Good and bad notations have also been discussed by [Wol100], who outline how the quality of mathematical notations has increased over time. The authors exemplify Diophantus' notation for polynomials as an example for a notation that is not a good notation. They argue that “*Diophantus' notation looks extremely hard to understand*” and assume that “*the main reason — apart from the fact that it's not very extensible — is that it doesn't provide the mathematical correspondence between different polynomials and does not highlight the things that we think are important*”.

3.4.3. Value for Education

Most participants agreed that adapting notations might actually be useful for students, which are not yet well-experienced in the area. Some doubted that replacing an author's notation would be useful and rather worried that this would impair the understandability of a text and destroy the author's intention. After all, authors spend much time to select intuitive notations that can be understood/accepted by the community.

One participant emphasised that mathematical features on notations and interactions are less interesting for mathematical research, they are more interesting for mathematical education, though not of interest to all universities. The Jacobs University Bremen has a very multicultural student body (71% of foreign students [Uni10]), but this is not the case for other (German) universities (10%-20% foreigners). The participant was wondering whether the notation management is interesting for the Open University Project [OU] or for the increasing number of universities, which publish their lecture materials on the web to advertise their courses¹. Adaptation of notation could potentially help these university to attract even more students; providing a better accessibility of material by taking multi-cultural aspects into account.

A colleague of the author approached a representative of the Open University for feedback on this matter. The instructor commented on the elision support, the hiding and display of brackets. According to him, such services only make sense for large and complex expressions, which are not very common in learning material. He emphasises that students need to be motivated to make use of computers for learning. Any service needs an immediate benefit in order to be used.

3.4.4. Understanding Notations

Some participants wonder if the problem of understanding mathematical notations is rather a problem of understanding the underlying concepts. They believed that the latter can be improved by supporting users with additional information such as the natural language terms for the notations, definitions, explanations, or examples. Ideally, this information should be filtered according to the user's background and skills. A reading environment should thus not only support the adjustment of notations but also the user-specific recommendation of supplementary materials.

¹For example, see the MIT online material on mathematics for computer science [MIT] or the CONNEXIONS repository [CNX].

3. Introduction

3.4.5. A List of Services

Based on the discussions and in several iterations with some participants, the below ‘wish list’ was derived. It outlines, which services could ease up the life of researchers as well as instructors and improve the learning experience of students. Most of the examples are based on the binomial coefficient symbol.

1. Alternative ways of displaying symbols:
 - Show all alternative notations: $\mathcal{C}(n, k)$, ${}_n\mathcal{C}_k$, \mathcal{C}_n^k , $\binom{n}{k}$,
 - Point out notational differences: “We write $\binom{n}{k}$, but you know it as \mathcal{C}_n^k ”,
 - Allow to change notations on the fly while reading a document,
 - Read notations out loud: ‘*n choose k*’ (an aural notation, in fact) – a service that was considered particularly helpful for foreigners to become acquainted with the technical vocabulary of a new language,
 - Provide a natural language term for the concept: *binomial coefficient*.
2. Explaining the structure of a formula:
 - Flexible display and hiding (‘elision’) of brackets: If the reader is not yet familiar with the precedences of new operators, allow for making the structure of a term more explicit by showing redundant brackets,
 - Folding of complex sub-terms: allow for collapsing terms whose full rendering takes up too much space; replace sub-terms that are hard to understand by instructive labels, e.g., by their scientific meaning: ‘potential energy’ or $W_{\text{pot}}(R)$ instead of $\frac{-e^2}{4\pi\epsilon_0 R/2}$.
3. Using additional knowledge from the definition of a symbol, if the reader does not understand the notation itself or wants to do further explorations for other reasons:
 - Interlink symbols and their definition (of different level of formality),
 - Provide an informal explanation: *The number of k-element subsets of an n-element set*,
 - Provide an even more informal explanation: *The number of ways that k things can be chosen from a set of n things*,
 - Generate a guided tour to explain a given (complex) formula, which provides all definitions and explanations of symbols in the formula.

Section 5 evaluates whether or not these services are supported by the proposed rendering workflow.

4. Pattern-based Rendering of Notations

To illustrate the adaptation of mathematical notations, the example in Figure 24 is used. The document includes several sections on fundamental mathematical concepts, such as the binomial coefficient, the imaginary unit, and the set of natural numbers. We want to adapt the document to the user, taking his language constraints and other preferences into account.

1.1 Binomial Coefficient

The binomial coefficient $\binom{n}{k}$ is the number of ways of choosing k objects from a collection of n distinct objects without regard to the order. . . .

2.3 Imaginary Unit

The imaginary unit i is defined solely by the property that its square is -1

3.2 Natural Numbers

The set of natural numbers is either the set of positive integers $\{1, 2, 3, \dots\}$, denoted by \mathbf{N}^+ , or the set of non-negative integers $\{0, 1, 2, \dots\}$, denoted by \mathbf{N}

Figure 24.: An example document

Listing 13 provides an extract of the OMDOC representation for the document (Section 2.1.5). Note that we use OMDOC only for illustration purpose, this approach can be applied to any XML document that uses OPENMATH/Content-MATHML to mark the functional structure of expressions.

Listing 13: OMDOC representation for Figure 24.

```
<omdoc xmlns="http://omdoc.org/ns" xmlns:m="http://www.w3.org/1998/Math/MathML"
  xmlns:om="http://www.openmath.org/OpenMath">
  <theory xml:id="combinat1">
    <symbol name="binomial" />
    <definition for="binomial">
5      The binomial coefficient is the number of ways of choosing  $k$  objects from a
      collection of  $n$  distinct objects without regard to the order. We denote it by
      <om:OMOBJ ic="language:en">
        <om:OMA>
10          <om:OMS cd="combinat1" name="binomial"/>
          <om:OMV name="n"/>
          <om:OMV name="k"/>
        </om:OMA>
      </om:OMOBJ>
    </definition>
15  </theory> ...
  <theory xml:id="complex1">
    <symbol name="imaginary" />
    <definition for="imaginary">
20      The imaginary unit
      <om:OMOBJ ic="area:math">
```

4. Pattern-based Rendering of Notations

```
    <om:OMS cd="complex1" name="imaginary"/>
  </om:OMOBJ> ...
</definition>
25 </theory> ...
  <theory xml:id="nat1">
    <symbol name="posInt" />
    <symbol name="nonnegInt" />
    <definition for="N">
30   The set of natural numbers is either the set of positive integers ...
      denoted by
      <om:OMOBJ>
        <om:OMS cd="nat1" name="posInt"/>
      </om:OMOBJ>,
35   or the set of non-negative integers ..., denoted by
      <om:OMOBJ>
        <om:OMS cd="nat1" name="nonnegInt"/>
      </om:OMOBJ>
    </definition>
40 </theory> ...
</omdoc>
```

The OMDOC document contains three content dictionaries `combinat1`, `complex1`, and `nat1`, which are represented by `theory` elements. The content dictionaries embed mathematical symbols (the binomial coefficient, the imaginary unit, and the set of positive/non-negative integers) and definitions. Note that the `nat1` theory introduces two symbols, one for the set of positive integers and one for the set of non-negative integers. This allows a machine to disambiguate how a user defines the set of natural numbers. The definitions define these symbols with a mixture of text and formal notations, where the latter are represented in OPENMATH. Each OPENMATH symbol (OMS) points to the respective symbol element with its name and `cd`¹: For example, `<om:OMS cd="combinat1" name="binomial" />` points to the symbol `binomial`, in the content dictionary `combinat1`.

The OMDOC representation can be easily converted into a presentation such as in Figure 24. However, the content objects in OPENMATH format can not be directly used for display. They have to be converted into a suitable presentation-oriented form, i.e., Presentation-MATHML. This process is challenging if the conversion should dynamically adapt to the user's notation practice. The remaining sections explain how the user-specific adaptation of the document can be supported.

4.1. Information Model

This work is based on the conversion workflow specified by [KLM⁺09, KMR08]. Accordingly, presentational characteristics of mathematical objects are encoded declaratively in **notation definitions**, which consist of **prototypes** (patterns that are matched against content representations of mathematical formulae) and **renderings** (that are used to construct the corresponding notations).

Listing 14: A notation definition for the binomial coefficient.

```
<notation xmlns="http://omdoc.org/ns" xmlns:m="http://www.w3.org/1998/Math/MathML"
  xmlns:om="http://www.openmath.org/OpenMath">
```

¹Note that we omit the `cdbase` attribute [BCC⁺04]


```

<prototype>
  <om:OMA>
    <om:OMS cd="combinat1" name="binomial" />
    <expr name="arg1"/>
    <expr name="arg2"/>
  </om:OMA>
</prototype>
<rendering ic="language:de,en">
  <m:mrow>
    <m:mo></m:mo>
    <m:mfrac linethickness="0">
      <render name="arg1"/>
      <render name="arg2"/>
    </m:mfrac>
    <m:mo></m:mo>
  </m:mrow>
</rendering>
<rendering ic="language:fr,ru">
  <m:mssubsup>
    <m:mi>C</m:mi>
    <render name="arg1"/>
    <render name="arg2"/>
  </m:mssubsup>
</rendering>
</notation>

```

Listing 14 presents the XML representation of a notation definition that can be used to generate a user-specific notations for the content-oriented representation of the binomial coefficient in Listing 13. The English/German notation $\binom{n}{k}$ and the French/Russian notation C_n^k are specified. The pattern in the `prototype` element matches with the content expression in Listing 13 (Line 9 to 13). By default the first `rendering` element is selected and applied to generate the Presentation-MATHML expression in Figure 21.

For the further discussion, it is assumed that we can draw on a huge corpus of mathematical documents in a content-oriented format, such as OMDOC, in which all expressions are represented in OPENMATH/Content-MATHML and notation preferences are encoded as notation definitions. The adaptation of mathematical symbols and formulae is implemented in three steps:

1. the **collection** of notation definitions,
2. the **matching** of the prototypes in these notation definitions with a mathematical object,
3. the **selection** of one appropriate, user-specific rendering from these notation definitions, which is applied to generate an appropriate notation.

The outcome of the first and last step depends on the *semantic context* of documents parts, the *narrative context* of the document, and his individual *user context*, which were introduced in Section 1.4. These can be prioritised by users and allow them to overwrite semantic or narrative constraints.

In the next sections, the motivation and pragmatics of the first and last step as well as the context prioritisation are discussed. [KLM⁺09, KMR08] provide details on the pattern matching and its purpose. The rendering algorithm and technical details are provided in Section 4.2 to 4.4.

4.1.1. Collection of Notation Definitions

In Section 3.4.1 two types of notations have been distinguished: standard and new notations. *Standard notations* present already discovered, well-known mathematical concepts. They are arranged in notation systems and depend on preferences and conventions of individuals, communities, and whole mathematical fields/areas. The choice of an appropriate standard notation depends on the context of the document (defined by its author, conference, or audience) and can be rather challenging for new members of a community.

The OPENMATH community [OM-] provides a number of content dictionaries [OMC] to define commonly known and standardised mathematical concepts. These include the OPENMATH representation of the expressions as well as alternative notations in Presentation-MATHML. If we assume that all documents are written in a content-oriented document format like OMDOC and that all mathematical expressions are represented in OPENMATH, these content dictionaries would also include notation definitions or reference a supplementary notation documents. Consequently, in the scope of an ideal world, where all documents are written in OMDOC and all notation preferences are encoded as notation definitions, users do not have to write standard notations but can reuse the notation definitions from content dictionaries – or any other marked up document. For example, a considerably large amount of standard notations is acquired during education: books and lecture notes introduce mathematical concepts with specific notations that depend on the background, area, and individual preferences of the instructor or author. Instead of rewriting notation definitions, users should be able to point to the marked-up documents that contain them, e.g., a book, lecture notes, a content dictionary, or a notation document. Given respective manipulation services and an integration in the user's working environment, authors could also be relieved from the burden of adapting their notations prior publication. By reusing notations from other documents, readers could configure notations at any later stage.

New notations have to be constructed by authors of new mathematical knowledge. For example, when writing a book on new mathematical concepts, authors can usually not draw on standardised notations but have to select new notations. New notations are challenging for both, authors and readers. Authors have to produce simple intuitive notations, while readers have to get acquainted to the new notations. Adaptation can not be supported if no alternative notations exist for the new mathematical expression. As readers have to provide their preferred notations, the costs of respective adaptation services increases.

The author doubts that machines are yet capable of relieving humans from the markup of notation preferences. Nevertheless, support can be provided for complex mathematical objects that have been constructed by combining simpler, well-known objects (i.e., Group A, as illustrated in Section 3.4.1). Markup helps to explicate the structure of such complex concepts and to identify the simpler constituents. Constituents that refer to standardised mathematical concepts for which alternative notations exists, can be adapted according to specific user preferences and conventions. Consequently, users should also be supported to apply notation definitions to research papers and new textbooks.

To support a *consistent rendering* of notations (as proposed in Section 3.4.2), notation definitions can be grouped into blocks of consistent notation systems (called notation documents). These can be used to avoid ambiguities and inconsistencies on the presentation level of mathematical expressions. For example, consider the alternative notations/definitions for the set of natural numbers in Section 3.1. To assure a consistent use, users can write two notation documents: one document with English notations, including \mathbf{N}^+ for positive integers and \mathbf{N} for the non-negative integers, and one document with notations for German number theory, including \mathbf{N} for positive integers and \mathbf{N}_0 for non-negative integers. Choosing only one of the two documents prevents ambiguities caused by the notation \mathbf{N} .

In additions, users need fine-grained ways to associate notation definitions with their mathematical formulae. Pointing to a whole document is too coarse-grained if authors want to make comparisons between two notations or change a formerly introduce notations in the course of their argument. For example, consider the following sentence:

“In Germany, we denote the binomial coefficient with $\binom{n}{k}$. In France it is displayed as C_n^k .”

The above observations have led to the six options for the collection of notations (henceforth called **extensional collection**, Section 4.4.1), which allow users to prioritise the semantic, narrative, and/or user context of the document.

- Collection from another **document** allows users to overwrite semantic and narrative constraints with notation definitions of another document,
- Collection from the **current document** prioritises the narrative context since only notation definitions that have been visited *before* a mathematical symbol or formulae are considered
- Collection from **content dictionaries** traces the inheritance structure of mathematical knowledge to find notation definitions and thus supports the semantic context,
- Fine-grained collection from other documents **based on references in the current document** supports the narrative context since only explicit pointers within the document are considered — semantic and user-specific constraints are ignored,
- Fine-grained collection from other documents **based on tags** supports users to associated notation definitions fine-grained with the document,
- **System defaults** can be used as fallback.

4.1.2. Context-Sensitive Selection

A fine-grained adaptation of documents based on the explicit selection of notation definitions and collections (notation documents, lecture notes, books, etc) is not always preferred. Sometimes users do not know where a specific notation definition can be found but can easily specify the properties of the output. Such properties can be provided in form of context parameters, i.e., dimension-value pairs, where the first component provides a context dimension (e.g., area or language) and the second component a specific context value (e.g., physics or German).

Consider a professor, who is invited to give a lecture at a foreign university. He experiences that the students struggle with his lecture. They can understand the basic ideas but when it comes to solving assignments and applying the concepts in the lecture, they struggle with the introduced notations as they have learnt others in prior courses. The professor would like to adapt his notations. He is not sure which notation definitions to select but can provide the nationality and language of his audience. We can also think of a student, who has finished his bachelor degree in mathematics and decides to start a master in physics. To prepare for his studies he wants to get acquainted to the new notation conventions. To get started, he would like to identify notations in his former lecturer notes that differ to the notation conventions in physics.

To conclude, in addition to the selection of specific notation definitions, users should be able to specify a set of context parameters that describe the preferred notations. These context parameters can be provided globally or fine-grained. Authors can inscribe such context attributions into their documents, while other users have to draw on stand-off specifications.

The previous observations have led to the following options for the collection of contextual information (henceforth called **intensional collection**, Section 4.4.2).

4. Pattern-based Rendering of Notations

- Selection based on the **global context** allows users to provide a global set of context parameters,
- Selection based on **cascading context files** allows users to specify fine-grained contextualisation analogously to cascading stylesheets,
- Selection based on **context parameters in the current document** (by attributes or metadata) prioritises the narrative or semantic context since only context parameters within the current document are considered.

4.2. Specification of Notation Definitions

Note that the author was involved in the specification of notation definitions [KLM⁺09, KMR08]. In the further course, the relevant aspects of notation definitions are briefly summarised. An extract of the grammar of notation definitions is presented in Table 2.

Notation declarations	ntn	$::=$	$\varphi^+ \vdash [(\lambda: \rho)^p]^+$
Patterns	φ	$::=$	
Symbols			$\sigma(n, n)$
Variables			$\nu(n)$
Applications			$@(\varphi[, \varphi]^+)$
Symbol/Variable/Object/List jokers			$\underline{s} \mid \underline{v} \mid \underline{o}[\varphi] \mid \underline{o} \mid \underline{l}(\varphi)$
Renderings	ρ	$::=$	
XML elements			$\langle S \rangle \rho^* \langle / \rangle$
XML attributes			$S = " \rho^* "$
Texts			S
Symbol or variable names			\underline{q}
Matched objects			\underline{q}^p
Matched lists			$\mathbf{for}(q, I, \rho^*)\{\rho^*\}$
Match contexts	M	$::=$	cf. [KLM ⁺ 09, KMR08]
Notation context	Π	$::=$	ntn^*
Adaptation context	Λ	$::=$	cp^*
Context annotation	λ	$::=$	cp^*
Context parameter	cp	$::=$	$(d = v)$
Context dimension	d	$::=$	$\sigma(n, n)$
Context value	v	$::=$	$\sigma(n, n)$
Precedences	p	$::=$	$-\infty I \infty$
Names	n, s, v, l, o	$::=$	C^+
Integers	I	$::=$	integer
Qualified joker names	q	$::=$	$l/q s v o l$
Weight	w	$::=$	real
Position	pos	$::=$	int
Strings	S	$::=$	C^*
Characters	C	$::=$	character except /

Table 2.: Extract of the grammar for notation definitions [KLM⁺09, KMR08]

The adaptation of mathematical notations is defined as the construction of a **variant notation** for a given mathematical content object ω . The new notation is generated (or *rendered*) by mapping ω to the patterns of a notation definition $ntn = \varphi_1, \dots, \varphi_r \vdash$

4.2. Specification of Notation Definitions

$(\lambda_1 : \rho_1)^{p_1}, \dots, (\lambda_s : \rho_s)^{p_s}$. If a matching pattern φ_i is found, one of the rendering elements ρ_i is used to generate the new notation. Which rendering is chosen, depends on the set of constraints (or context parameters) that guide the selection and reflect the user's notation practice. The integer values p_i give the output precedences of the renderings, which are used to dynamically determine the placement of brackets.

The **patterns** φ_i are formed from a formal grammar for a subset of OPENMATH objects extended with named jokers [KLM⁺09, KMR08]. For example, the `pattern` element in Listing 14, includes two variable jokers, `arg1` and `arg2` (represented with an `expr` element), which can be referred to in the renderings ρ_i . Note that mathematical content objects ω_i are patterns that do not contain jokers.

The **renderings** ρ_i are formed by a formal syntax for simplified XML extended with means to refer to the jokers used in the patterns [KLM⁺09, KMR08]. For example, the renderings in Listing 14 (specifically their `render` elements) refer to the two variable jokers `arg1` and `arg2` and initiate a recursion of the rendering algorithm.

A **notation context** Π contains all available notation definitions for the rendering, where Π_ω denotes the set of notation definitions that is available for rendering a content object ω .

A **context parameter** cp is represented as key-value pair ($d = v$), where d denotes a context dimension and v its context value. Context parameter describe document parts (sections, definitions, mathematical expressions, etc) or – in other words – specify the properties that a document part has to satisfy. They can be provided with inline or stand-off markup. Context parameters thus act as constraints for the adaptation of notations. For example, in Listing 14 the values of the `ic` attributes describe the `rendering` elements of the notation definition. The first `rendering` can be selected to produce a German/English notation for the binomial coefficient, while the second rendering can be used to denote the concept with its French notation. The `ic` attributes of the mathematical objects (represented with an `OMOBJ` element) in Listing 13 define the constraints for the adaptation: an English notation and a notation for math is preferred.

We call an object description **context annotation** (denoted by λ) and the adaptation constraints **adaptation context** (denoted by Λ). Both are represented as ordered set of context parameters (cp^*). The **effective adaptation context** of a document part depends on its position in the document. It is a concatenation of the context parameters, which have been associated with the part, and the context parameters of its parents in the XML document tree. The order of a context parameter cp_s in Λ denotes its **weight** w for the adaptation. It is computed by subtracting the position pos of cp_s from the total number of context parameters in Λ (or the size of the set Λ), i.e., $w(cp_s) = size(\Lambda) - pos(cp_s)$.

Context dimensions and **context values** correspond to mathematical symbols, which are defined in content dictionaries. For example, the content dimension $d = \sigma(cd, n)$ references the mathematical symbol n in the content dictionary cd . Here such content dictionaries are used as background ontology for context metadata [LK09]. They support users and developers of adaptation systems to create and document new dimension/value symbols and, thus, to introduce context parameters that can be used within the adaptation.

Listing 15 illustrates a content dictionary represented in OMDOC: the dimension `language`², `area`, and `difficulty` are defined. The dictionary does not only define, which values a specific dimension can take but can also define certain properties of the set of possible values. For example, the set of values for the `difficulty` dimension is a set $\{high, medium, low\}$, where $high > medium > low$.

²Note that as OMDOC integrates the Dublin Core scheme [Dub] (e.g., defining the language dimension), the content dictionary could only contain the missing dimension and value symbols. However, as we are targeting an approach that is *independent of a specific document format*, we define all relevant context dimension in a comprehensive document.

4. Pattern-based Rendering of Notations

Listing 15: A content dictionary for context dimension and context values

```

<omdoc ...>
  <theory xml:id="var.context">
    <symbol name="language" /><symbol name="en"/><symbol name="de"/>
    <definition for="language">
      An "language" dimension specifies the nationality or language area in which a notation
      definition is commonly used. It can have values such as
      <om:OMS name="de"> or <om:OMS name="en">.
    </definition>
    <symbol name="area" /><symbol name="math"/><symbol name="physics"/>
    <definition for="area">
      An "area of application" dimension specifies the particular environment (or scope) of a notation
      definition. It can have values such as
      <om:OMS name="math"> or <om:OMS name="physics">.
    </definition>
    <symbol name="difficulty" />
    <symbol name="high"/><symbol name="medium"/><symbol name="low"/>
    <definition for="difficulty">
      The "difficulty" dimension specifies the skill or prior knowledge, which is commonly required
      for understanding/using a notation definition. It can have one of the values in
      <om:OMOBJ>
        <om:OMA>
          <om:OMS cd="set1" name="set" />
          <om:OMS name="high"><om:OMS name="medium"><om:OMS name="low">
        </om:OMA>
      </om:OMOBJ>, where ...
    </definition> ...
  </theory>
</omdoc>

```

4.3. The Rendering Algorithm

Figure 25 illustrates the components of the rendering algorithm: the *notation collector*, the *context collector*, the *pattern matcher*, and the *rendering grabber*.

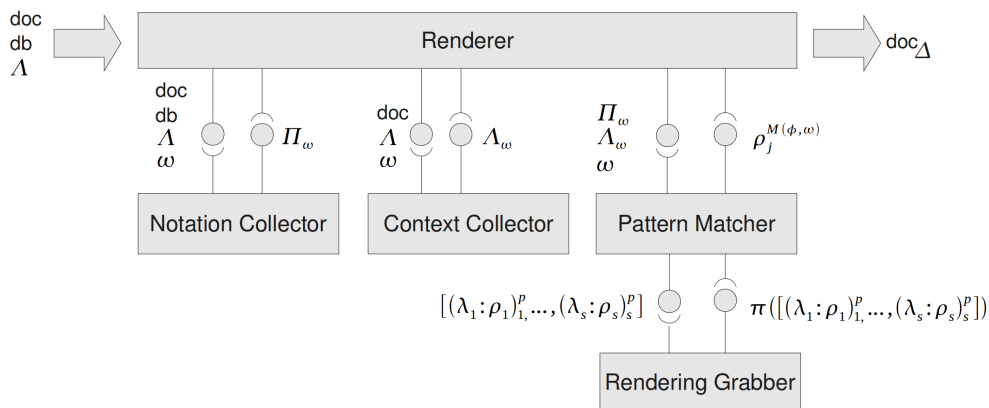


Figure 25.: The components of the rendering algorithm

The **renderer** takes as input a document doc , a document database db , and an adaptation context Λ . The intuition of db is that it provides all cross-references and imported documents for the adaptation. The algorithm outputs doc_{Δ} , where all OPENMATH objects have been replaced with appropriate notations. The input document doc is traversed from *left to right*³ and every OPENMATH object ω is substituted with a notation μ in Presentation-MATHML.

Listing 16: Generating a Presentation-MATHML notation μ for an OPENMATH object ω

```

1  $\Pi_{\omega}$  = construct notation context for  $(\omega, \Lambda, db, doc)$ 
2
3  $\Lambda_{\omega}$  = compute effective adaptation context for  $(\omega, \Lambda, doc)$ 
4
5  $\rho_j^{M(\varphi, \omega)}$  = select rendering element according to  $(\omega, \Lambda_{\omega}, \Pi_{\omega})$ 
6
7  $\mu$  = generate notation for  $\omega$  from  $\rho_j^{M(\varphi, \omega)}$ 

```

Listing 16 specifies the steps for the generation of notations. In the first step, the *notation collector* constructs the notation context Π_{ω} for ω , which contains all notation definitions according to object ω , the adaptation context Λ , the document database db , and the input document doc . The *context collector* computes the effective adaptation context Λ_{ω} for ω according to Λ and doc . Given Π_{ω} and Λ_{ω} as well as the object ω , the *pattern matcher* selects $\rho_j^{M(\varphi, \omega)}$, the rendering element ρ_j in context $M(\varphi, \omega)$. $\rho_j^{M(\varphi, \omega)}$ is used to generate the notation μ for ω .

4.3.1. The Notation Collector

The **notation collector** takes as input an object ω , its adaptation context Λ , the input document doc , and the database db . It returns the normalised notation context Π_{ω} , which contains all notation definitions that are explicitly selected for ω .

Listing 17: Constructing the notation context

```

1  $\Pi_{\omega}$  = collect all notation definitions for  $\omega$  from  $(doc, db)$ 
2 return normalise  $\Pi_{\omega}$ 

```

Listing 17 specifies the steps for the construction of the notation context. In a first step, all notation definitions for ω are collected from doc and db and form the notation context Π_{ω} . The collection is guided by the adaptation context Λ , which specifies which of the inline and stand-off markup options are considered for the collection (Section 4.4.1). Π_{ω} is normalised by grouping together renderings of the same patterns. If notation tags (Section 4.4.1) are provided, the elements in the renderings are filtered: only those remain, which are explicitly referenced by a tag. The normalised notation context is returned.

4.3.2. Context Collector

The **context collector** takes as input an object ω , the adaptation context Λ , and the input document doc . It returns the effective adaptation context for ω , which is computed from the inline and stand-off markup of context parameters. These markup options are discussed in Section 4.4.2.

³Note that we consider an XML document to be an ordered tree, following the XPATH data model specified in [CD99]. We use the term document and document tree interchangeably.

4. Pattern-based Rendering of Notations

4.3.3. Pattern matcher

The **pattern matcher** takes as input an object ω , its adaptation context Λ_ω , and the notation context Π_ω . It returns $\rho_j^{M(\varphi,\omega)}$. If the pattern matcher is invoked recursively to render a sub-object of ω , it takes an input precedence p , which is used for bracket placement, as an additional argument.

Listing 18: Constructing $\rho_j^{M(\varphi,\omega)}$

```
1  $\varphi_i$  = match  $\omega$  against the patterns in  $\Pi_\omega$ 
2  $L$  = get list of rendering tuples from  $ntn_i$ 
3  $L'$  = order  $L$  according to  $\Lambda_\omega$ 
4 return the first rendering  $\rho_j^{M(\varphi,\omega)}$  from  $L'$ 
```

Listing 18 specifies the computation by the pattern matcher. To construct $\rho_j^{M(\varphi,\omega)}$, ω is matched against the patterns in the notation definitions in Π_ω until a matching pattern φ is found. The notation definition, in which φ occurs, induces a list of elements of the form $(\lambda: \rho)^p$, where ρ denotes a rendering, λ its context annotation, and p its precedence. The *rendering grabber* is called to order the list according to Λ_ω . The first rendering is selected. $\rho_j^{M(\varphi,\omega)}$ is returned: the rendering of ρ_j in context $M(\varphi,\omega)$ as defined in [KLM⁺09, KMR08].

4.3.4. The Rendering Grabber

The **rendering grabber** takes as input an object ω , its adaptation context Λ_ω , and a list L of elements of the form $(\lambda: \rho)^p$. It returns the permutation of L (L'), in which the elements are ordered according to how well their context annotation λ matches Λ_ω , starting with the best matching one.

Listing 19: The matching of context annotation and adaptation context

```
1  $W$  = empty map
2  $L'$  =  $\emptyset$ 
3
4 forall  $(\lambda_i: \rho_i)^p$  in  $L$  do
5    $w(\lambda_i)$  = 0
6   forall  $cp_j$  in  $\lambda_i$  do
7     if  $cp_k$  in  $\Lambda_\omega$  and  $cp_k$  equals  $cp_j$  then
8        $w(cp_j)$  = order of  $cp_k$ 
9     fi
10    add  $w(cp_j)$  to  $w(\lambda_i)$ 
11  done
12  add key-value pair  $(\lambda_i: \rho_i)^p \rightarrow w(\lambda_i)$  to  $W$ 
13 done
14
15 while  $size(L') < size(L)$  do
16    $(\lambda_i: \rho_i)^p$  = get first key from  $W$  with max value
17   delete key-value pair from  $W$ 
18   append  $(\lambda_i: \rho_i)^p$  to  $L'$ 
19 done
20
21 return  $L'$ 
```

Listing 19 specifies the matching of context annotations with the adaptation context. In the first step, the map W is created, which maps the elements in L with their weight. To compute

a weight for an element $(\lambda_i: \rho_i)^p$ in L , the context parameters cp_j in λ_i are processed. All cp_j in λ_i that are equal to a context parameter cp_k in Λ_{ω_2} are weighted with the order of cp_k . The weight of an element $(\lambda_i: \rho)^p$ is computed by adding up the weights of the context parameters in λ_i . In a second step, the weight map W is used to create the permutation L' . All elements in W are added to L' as follows: the first key with the maximum value is selected, the key-value pair is deleted from W and is appended to L' .

4.4. Guiding the Rendering Algorithm

After briefly introducing the rendering algorithm, we will now discuss different options that allow the user to guide the processing of the notation collector and the rendering grabber.

4.4.1. Guiding the Notation Collector

We permit users to define the set of available notation definitions *fine-grained*, that is, for any mathematical object (and sub-object) in the input document. We call this **extensional selection** as user have to explicitly point to a rendering element, a notation definition, or a document with notation definitions (called a **notation container**). In the following, we discuss the collection of notations from various sources and the construction of Π_ω for a concrete mathematical object ω .

Options for Collecting Notation Definitions

The notation collector process two main steps: The *collection* of notation definitions and their *reorganisation*. The first step can be guided by several inline and stand-off markup options, which are provided by the adaptation context Λ . In particular, Λ specifies a totally ordered set \mathcal{S}_N of source names. Technically, the collection sources are represented as contextual key-value pairs, where the key denotes a source name and the value its priority. To allow machines to process source names (the context dimensions) and their priority values (the context values), these are defined in context dictionaries, such as illustrated in Listing 15.

Based on the hierarchy proposed in [NW03], we use the source names F , Doc , CD , EC , T , and SD explained below. The user can change their priorities by ordering them. The respective input sources are treated as follows.

- F denotes an **external notation document** from which notation definitions are collected. F can be used to overwrite the author's extensional context declarations.
- Doc denotes the **input document**. As an alternative to EC , Doc permits authors to embed notation definitions into the input document. Note that the XML document tree is traversed from left to right. All notation definitions that have been visited *before* the mathematical object ω are collected.
- CD denotes the **content dictionaries** defining symbols occurring in ω . These are searched in the order in which the symbols occur in ω . Content dictionaries may include or reference default notation definitions for their symbols.
- EC denotes the **extensional context**. It is used by authors to associate a list of renderings, notation definitions, or containers of notation definitions to any concept of the input document. The EC option has to be used in combination with either F (provided in an external file) or Doc (embedded in the input document). The effective extensional context is computed according to the position of ω in the input document.

4. Pattern-based Rendering of Notations

- T denotes **notation tags**, which provide the same functionality as the EC option without changing the input document. They allow readers full control of the eventually selected rendering elements as they explicitly point to the appropriate notations. T has to be used in combination with either F (provided in an external file) or Doc (embedded in the input document). Tags are collected in the order they occur in these documents.
- SD denotes the **system default** notation document, which typically occurs last in \mathcal{S}_N as a fallback if no other notation definitions are given.

In the second step the obtained notation context Π_ω is normalised: All occurrences of a pattern φ in notation definitions in Π_ω are merged into a single notation definition. All EC and T references to rendering elements are applied to filter and prioritise the $(\lambda: \rho)^p$ pairs.

Representation of Extensional Options

This section illustrates the machine-processable representation of the extensional options F , Doc , CD , EC , T , and SD .

Documents The F , Doc , and CD option reference container documents. Theoretically, the rendering algorithm can handle any kind of XML document, e.g., CNXML (Section 2.1.2), DOCBOOK (Section 2.1.3), and OMDOC (Section 2.1.5). However, all illustrations and the proof-of-concept implementation focusses on OMDOC documents.

System Default The system defaults are represented internally in a rendering system, such as the renderer for OMDOC (Section 5.2.1). Listing 20 illustrates a default notation definition for the rendering of variables. The pattern of the first notation definition matches with any OPENMATH OMV element (e.g., `<om:OMV name="n" />`), the variable `joker` `varname` captures the value of the `name` attribute of the OMV (e.g., `n`). The rendering generates an Presentation-MATHML `mi` element and implicitly embeds the captured name of the variable (e.g., `<m:mi>n</m:mi>`).

Listing 20: System defaults in the rendering prototype

```
<notation>
  <prototype>
    <element ns="http://www.openmath.org/OpenMath" name="OMV">
      <attribute name="name">
        <tvar name="varname" />
      </attribute>
    </element>
  </prototype>
  <rendering>
    <m:mi>
      <render name="varname" />
    </m:mi>
  </rendering>
</notation>
```

Extensional context The extensional context is represented with an `ec` attribute in the OMDOC namespace, which may be associated to any XML element. The value of the `ec` attribute is a whitespace-separated list of URIs of either rendering elements, notation definitions, or any other XML document. The latter is interpreted as a container, from which

notation definitions are collected. The `ec` attribute is empty by default. When computing the effective extensional context of an element, the values of the `ec` attributes of itself and all parents are concatenated, starting with the innermost. Listing 21 shows the OPENMATH representation for the imaginary unit. The `ec` attribute references the notation definition `img.ntn`.

Listing 21: Representation of the extensional context attribute

```
<OMOBJ>
  <OMS ec="#img.ntn" cd="complex1" name="imaginary"/>
</OMOBJ>
```

Notation tags Notation tags provide the same functionality as the extensional context but do not require to change the input document. They are represented with a `tag` element in the OMDOC namespace. The `type` attribute provides the type of the tag (e.g., `notation`). The value of the `xref` attribute is a whitespace-separated list of URIs of either rendering elements, notation definitions, or any other XML document. The value of the `object` attribute is a whitespace-separated list of URIs of references to mathematical objects, to which the rendering should be applied to. The optional `ic` attribute associates context parameters with the notation tag (Section 4.4.2) and the optional `owner` attribute relates a tag to the respective author or reader of a document. Its value is a URI reference to the user profile, e.g., identified by an OPENID (Section 10.3.6).

Listing 22: Two example notation tags

```
<notation ... xml:id="ntn123">
  <rendering xml:id="rend789"> ... </rendering>
  <rendering xml:id="rend456"> ... </rendering>
</notation>

<om:OMOBJ><om:OMS xml:id="obj455" name="imaginary" /></om:OMOBJ>
<om:OMOBJ><om:OMS xml:id="obj456" name="imaginary" /></om:OMOBJ>

<tag type="notation" xref="#rend456" ic="hasLanguage:German" object="#obj455 #obj456"
  owner="http://cmueller.myopenid.com"/>
```

Listing 22 illustrates a notation tag, which associates the rendering `rend456` to the objects `obj455` and `obj456` and is owned by a user `http://cmueller.myopenid.com`.

Example

The further illustration is based on Figure 26, which presents the structure tree of the document in Figure 24. Section 2.3. contains a notation definition for the imaginary unit, it includes one rendering element, which will render ω_2 as j . The dashed, red arrow represent an extensional references: the document root `document` references the notation document `en-math.doc`, which is interpreted as a container of notation definitions. `SD` is a collection of system defaults.

We call the notation collector to gather the notation definition for ω_2 . The adaptation context specifies the source list $\mathcal{S}_N = \{EC, SD\}$. For simplicity, context annotations and precedences are not displayed and a notation definition is denoted by $\varphi_2 \vdash j$, where φ_2 matches with ω_2 and j represents the $(\lambda: \rho)^p$ element, which, if applied, generates the notation j .

4. Pattern-based Rendering of Notations

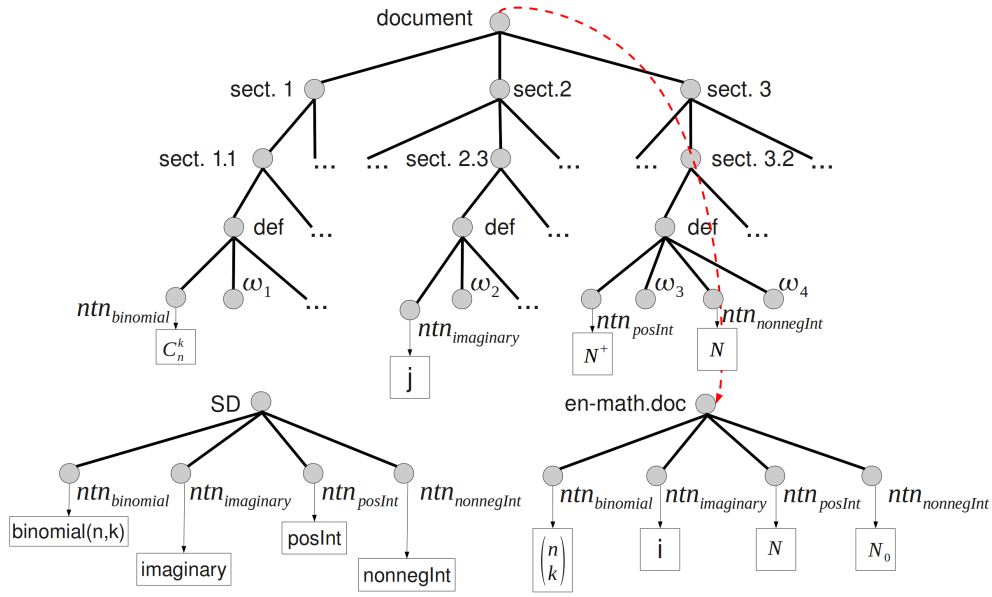


Figure 26.: Document tree with extensional options

```

1   $ec(\omega_2)$  = compute effective extensional context
2   $ec(\omega_2) = \{en - math.doc\}$ 
3
4   $\Pi_{\omega_2}$  = collect notation definitions for the EC option
5   $\Pi_{\omega_2} = \{\varphi_1 \vdash \binom{n}{k}, \varphi_2 \vdash i, \varphi_3 \vdash N, \varphi_4 \vdash N_0\}$ 
6
7   $\Pi_{\omega_2}$  = add notation definitions for SD
8   $\Pi_{\omega_2} = \{\varphi_1 \vdash \binom{n}{k}, \varphi_2 \vdash i, \varphi_3 \vdash N, \varphi_4 \vdash N_0, \varphi_1 \vdash binomial(\underline{n}, \underline{k}), \varphi_2 \vdash imaginary,$ 
9   $\varphi_3 \vdash posInt, \varphi_4 \vdash nonnegInt\}$ 
10
11  $\Pi'_{\omega_2}$  = normalise  $\Pi_{\omega_2}$ 
12  $\Pi'_{\omega_2} = \{\varphi_1 \vdash \binom{n}{k}, binomial(\underline{n}, \underline{k}), \varphi_2 \vdash i, imaginary, \varphi_3 \vdash N, posInt, \varphi_4 \vdash N_0, nonnegInt\}$ 

```

The listing above illustrates the construction of the normalised notation context Π'_{ω_2} . In a first step, the effective extensional context $ec(\omega_2)$ is computed based on the position of ω_2 in *doc*. The reference in $ec(\omega_2)$ are resolved and all notations definitions are collected – they form the notation context Π_{ω_2} . In the next step, the notation definitions for *SD* are collected and appended to Π_{ω_2} . Finally, Π_{ω_2} is normalised, yielding Π'_{ω_2} . The renderer receives the normalised notation context Π'_{ω_2} and calls the pattern matcher, which matches ω_2 against the patterns in Π'_{ω_2} , until the matching pattern φ_2 is found. As no further context parameters are given, the rendering grabber cannot reorder the induced list of rendering elements $\{i, imaginary\}$. Thus, the first rendering is selected and ω_2 is substituted with i . Note that we compute the same notation context for any ω_i in the example: $\Pi'_{\omega_2} = \Pi'_{\omega_1} = \Pi'_{\omega_3} = \Pi'_{\omega_4}$. Consequently, all remaining notations are generated based on the first rendering in the notation context above. ω_1 is substituted with $\binom{n}{k}$, ω_3 with N , and ω_4 with N_0 . The output document is displayed in Section 4.4.3. The notation definitions for the two variables n and k , which are also part of the system defaults *SD* (see Listing 20), have been omitted.

4.4. Guiding the Rendering Algorithm

To illustrate that different extensional options produce a varying output document, we now call the notation collector with the source list $\mathcal{S}_N = \{Doc, EC, SD\}$. For simplicity, we leave out the system defaults.

```

1  $\Pi_{\omega_2}$  = collect all notation definitions for Doc
2    $\Pi_{\omega_2} = \{ \varphi_1 \vdash \mathcal{C}_n^k, \varphi_2 \vdash j, \varphi_3 \vdash N^+, \varphi_4 \vdash N \}$ 
3
4  $\Pi_{\omega_2}$  = add notation definitions for EC
5    $\Pi_{\omega_2} = \{ \varphi_1 \vdash \mathcal{C}_n^k, \varphi_2 \vdash j, \varphi_3 \vdash N^+, \varphi_4 \vdash N, \varphi_1 \vdash (\frac{n}{k}), \varphi_2 \vdash i, \varphi_3 \vdash N, \varphi_4 \vdash N_0 \}$ 
6
7  $\Pi'_{\omega_2}$  = normalise  $\Pi_{\omega_2}$ 
8    $\Pi'_{\omega_2} = \{ \varphi_1 \vdash \mathcal{C}_n^k, (\frac{n}{k}), \varphi_2 \vdash j, i, \varphi_3 \vdash N^+, N, \varphi_4 \vdash N, N_0 \}$ 

```

The above listing illustrates the construction of Π'_{ω_2} for the *Doc* and *EC*. For the *Doc* option, all notation definitions before ω_2 in the input document are collected. The renderer receives the normalised notation context Π'_{ω_2} and calls the pattern matcher. As no further context parameters are given, the pattern matcher returns the first rendering in the induced list of rendering elements $\{j, i\}$, which is different than in the previous example: This time, ω_2 is substituted with j . Note that we compute the same notation context for any ω_i in the example and substitute ω_1 with \mathcal{C}_n^k , ω_3 with N^+ , and ω_4 with N . The output document is displayed in Section 4.4.3.

Discussion of Collection Strategies

In the following, we illustrate the advantages and drawbacks for collecting notations from the extensional options *F*, *Doc*, *CD*, *EC*, *T*, and *SD*.

- Authors can write documents which only include content objects and do not need to provide any notation definitions. Instead, they can reuse notation definitions from *CD* or simply rely on the defaults *SD*. The former allows to produce documents that conform to commonly agreed on notation conventions.
- The external document *F* permits authors to overwrite the default notation and to introduce individual notations that can be reused with any of their documents. However, *F* is applied globally to all notations in the document and does not support the rendering of different notations for the same mathematical expressions, e.g., in order to make comparisons such as $\mathcal{C}(n, k)$, ${}_n\mathcal{C}^k$, ${}^n\mathcal{C}_k$, ${}_n\mathcal{C}_k$, \mathcal{C}_n^k .
- Authors may embed notation definitions inside the structure of documents (*Doc*), which is more fine-grained than the *F* option but can cause redundancies and increase authoring/maintenance costs.
- Authors can use *EC* to include notation definitions and rendering elements fine-grained without embedding them multiple times into their documents. They can either embed them all in one section or outsource them to a separate document.
- *T* and *EC* select between alternative renderings inside one notation definition. All other options force users to modularise their notation definitions so that each only includes one rendering element.
- Users can overwrite the author's notation preferences fine-grained using the *T*, though, this can lead to inconsistencies, if only one but not all notations for a mathematical symbol are replaced.
- The *F* option can be used to collect notations for specific context (or notation system) and to assure that notations are *consistently* overwritten. In the example,

4. Pattern-based Rendering of Notations

`en-math.doc` denotes a document with English and math notations (including \mathbf{N}^+ for positive integers and \mathbf{N} for the non-negative integers). Similarly, we can create a document `de-number.doc` with notations for German number theory (including \mathbf{N} for positive integers and \mathbf{N}_0 for non-negative integers). Replacing `en-math.doc` with `de-number.doc` prevents ambiguities caused by the notation \mathbf{N} .

All of the above options have benefits and limitations. They allow authors and readers to explicitly relate any mathematical concept or formulae in a document with a specific notation definition (or rendering element). Alternatively, users can draw on context-sensitive selections. These allow users to implicitly guide the rendering by providing properties of the target document rather than pointing to a concrete output.

4.4.2. Guiding the Context Collection and Rendering Grabber

A fine-grained adaptation of documents based on the explicit selection of notation definitions and collections (notation documents, lecture notes, books, etc) is not always preferred. Sometimes users do not know where a specific notation definition can be found but can easily specify the properties of the output. Such properties can be provided in form of context parameters, i.e., dimension-value pairs, where the first component provides a context dimension (e.g., `area` or `language`) and the second component a specific context value (e.g., `physics` or `German`). The following sections discuss the collection of context parameters and illustrate how they can guide the ordering of rendering elements in the rendering grabber.

Options for Collecting Contextual Information

The ordering of rendering elements by the rendering grabber can be guided by several inline and stand-off markup options, which are provided by the adaptation context Λ : Λ specifies a totally ordered set \mathcal{S}_C of source names, which are represented as contextual key-value pairs. We allow the names *GC*, *CCF*, *IC*, and *MD*. When computing the effective adaptation context Λ_ω , the source names are resolved and the respective contextual information is collected and added as context parameters to Λ_ω . The respective input sources are treated as follows:

- *GC* denotes the **global context** which provides a global set of context parameters.
- *CCF* denotes **cascading context files**, which permit a fine-grained, stand-off contextualisation analogously to Cascading Style Sheets [BLLJ08].
- *IC* denotes the **intensional context**. It supports the inline association of contextual key-value pairs ($d = v$) with any element of the input document.
- *MD* denotes **metadata** in the input document, the strict inline markup alternative for the `ic` attribute.

Representation of Intensional Options

This section illustrates the machine-processable representation of the intensional options *GC*, *CCF*, *IC*, and *MD*.

```
language=de
area=math
```

Global Context The global context is provided in the required input format of the rendering system. For example, the renderer for OMDOC (Section 5.2.1) parses an ASCII representation of context parameters as illustrated to the right.

CCF Users can write a Cascading Context File, which associates context parameters to the input document using the *id* and *class* selectors as specified by Cascading Style Sheets [BLLJ08].

Listing 23 provides an OMDOC example. It includes *id* and *class* selectors, which context properties are specified in the CCF below. The CCF associates a context parameter (*area = math*) to all elements with *class=math* and the context parameter (*area = physics*) to all elements with *id=physics*. Note that the *class* and *id* attribute are not allowed by the OPENMATH format. In order to use the CCF approach, one would have to extend the specification of OPENMATH [BCC⁺04].

Listing 23: Parsing context parameters from a CCF

```
<omdoc>
  <omtext>
    <CMP>In mathematics the imaginary number is represented
    as <om:OMOBJ>
      <om:OMS class="math" cd="complex1" name="imaginary"/>
      </om:OMOBJ>, while electrical engineers will use <om:OMOBJ>
      <om:OMS id="physics" cd="complex1" name="imaginary"/></om:OMOBJ>.
    </CMP>
  </omtext>
</omdoc>
```

```
.math { area: math }
#physics { area:
  physics }
```

Intensional Context The intensional context is implemented as an *ic* attribute in the OMDOC namespace. The value of the *ic* attribute is a semicolon-separated list of elements in the form $d : v_1, \dots, v_i$, where v_1, \dots, v_i are multiple values for the context dimension d . The *ic* attribute in Listing 24 provides the context parameter (*language = en, de*).

Listing 24: Representation of the intensional context attribute

```
<OMOBJ>
  <OMA>
    <OMS ic="language:en,de" cd="combinat1" name="binomial"/>
    <OMV name="n"/>
    <OMV name="k"/>
  </OMA>
</OMOBJ>
```

Metadata The representation of metadata is actually a more structured alternative to the *ic* attribute: In OMDOC jargon, *ic* is the *pragmatic* markup for context parameters, while *metadata* is the *strict* markup form. One could argue that *metadata* is more coarse-grained than the less structured *ic* attribute as it can not be associated to every node in the input document. For some metadata schemes this is true as they only allow to describe coarse-grained components of a document and have a rather limited vocabulary. Other metadata formats are more flexible. In particular, an RDFa [RDF08] markup supports users to add metadata to almost any fragment of a document, without being limited to a particular metadata vocabulary [ABMP08].

By representing metadata as RDFa, the new metadata syntax for OMDOC [LK09] supports the integration of several metadata schemes with only little modifications of the document format. Only two elements are added to the document format: The *meta* element (for the markup of properties) and the *link* element (for the markup of properties and dependencies).

4. Pattern-based Rendering of Notations

Listing 25: Metadata Representation in OMDOC

```

<proof xml:id="proof1" for="#lemma1" xmlns:cc="http://omdoc.org/ctxt.omdoc?var.context?">
  <metadata>
    <meta property="cc:language">cc:de</meta>
    <meta property="cc:area">cc:math</meta>
    <meta property="cc:expertise">cc:intermediate</meta>
    <link rel="o:proves" href="#lemma1" />
    <link rel="cc:more-difficult-than" href="#proof2"/>
  </metadata>
  <om:OMOBJ><om:OMS cd="complex1" name="imaginary"/></om:OMOBJ>
</proof>

```

Listing 25 illustrates the metadata for a proof object. The namespace declaration introduced the prefix `cc`, which points to the content dictionary `http://omdoc.org/ctxt.omdoc?var.context?` in Listing 15. Due to this reference, the metadata markup is more structured, it provides the meaning for the context dimension and values in the respective namespace. The strict metadata syntax can be parsed into richer context parameter, e.g., $(cc:language = cc:de)$, $(cc:area = cc:math)$, and $(cc:expertise = cc:intermediate)$ for the proof in Listing 25. As the specification of metadata is a central topic of one of the author's colleagues [LK09], the further discussions focus on the pragmatic specification of context parameter and omit references to a symbol's content dictionary. Future work has to address how metadata of fine-grained components like `CMP` or `OMOBJ` elements can be supported.

Example

The further illustration is based on Figure 27. A global context is declared, which specifies the language and area for the whole document: The output should include notations for German and physics. The dashed, blue arrow represent an intensional reference: the definition for imaginary unit is associated with a context parameter ($area = math$), which is interpreted as request to render it according to mathematical conventions. The context parameter initiates the selection of an appropriate rendering in `en-math.doc`, which is also annotated with the context parameter ($area = math$).

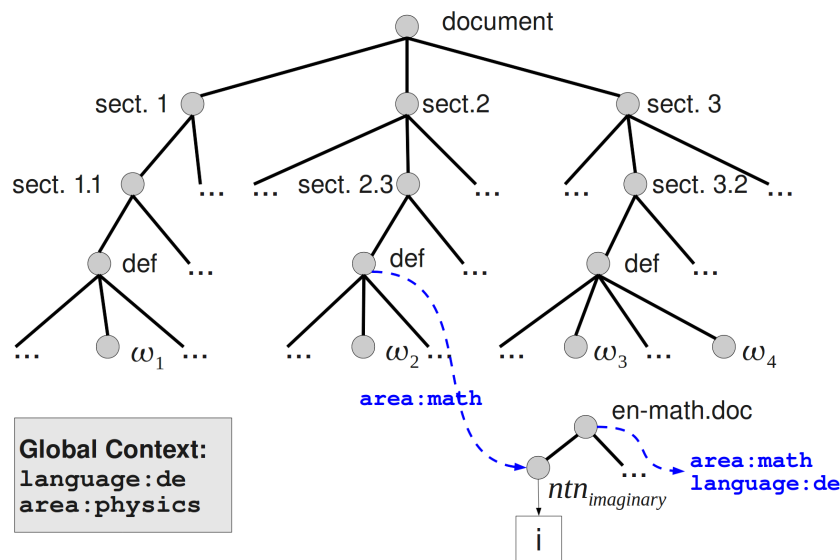


Figure 27.: Document tree with intensional options

4.4. Guiding the Rendering Algorithm

We apply the rendering algorithm with the input object ω_2 , $\mathcal{S}_C = (GC, IC)$, and a list of context annotations and rendering pairs $(\lambda: \rho)^p$ as computed by the pattern matcher. For simplicity, the precedences and SD are omitted and rendering elements are denoted with the resulting notations.

```

1  $\Pi'_{\omega_2}$  = collect notations according to  $\Lambda$ 
2  $\Pi'_{\omega_2} = \{ \varphi_1 \vdash C_{\underline{n}}^k, \binom{n}{k}, \varphi_2 \vdash j, i, \varphi_3 \vdash N^+, N, \varphi_4 \vdash N, N_0 \}$ 
3
4  $\Lambda_{\omega_2}$  = collect the context parameters from  $GC$ 
5  $\Lambda_{\omega_2} = \{(\text{language} = \text{de}), (\text{area} = \text{physics})\}$ 
6
7  $\Lambda_{\omega_2}$  = add context parameters from  $IC$ 
8  $\Lambda_{\omega_2} = \{(\text{language} = \text{de}), (\text{area} = \text{physics}), (\text{area} = \text{math})\}$ 
9
10  $n\text{tn}$  = match  $\omega_2$  with  $\Pi'_{\omega_2}$  until  $\varphi_2$  is found
11
12  $L$  = get list of context annotations/rendering pairs
13  $L = \{(\{(\text{area} = \text{physics})\}: j), (\{(\text{area} = \text{math})\}: i)\}$ 
14
15  $L'$  = create permutation of  $L$  according to  $\Lambda_{\omega_2}$ 
16  $L' = [(\{(\text{area} = \text{physics})\}: j), (\{(\text{area} = \text{math})\}: i)]$ 
17
18 substitutes  $\omega_2$  with  $j$ 

```

The above listing illustrates the rendering of ω_2 . In the first step, the effective adaptation context is computed by collecting the context parameters from GC and IC . In the next step, the notation collector returns the notation context Π'_{ω_2} according to $\mathcal{S}_N = (Doc, EC, SD)$. The pattern matcher matches ω_2 with Π'_{ω_2} until the matching pattern φ_2 is found. The notation definition induces a list L of context annotations and rendering pairs. The rendering grabber matches Λ_{ω_2} against the context annotations λ of the elements in L . It returns its permutation L' . The render substitutes ω_2 with j . The language parameters guides the presentation of all remaining concepts: ω_1 is substituted with the German notation $\binom{n}{k}$, ω_3 with the German notation N , and ω_4 with the German notation N_0 . The output document is displayed in Section 4.4.3.

Discussion of Context-Sensitive Selection Strategies

In the following, we illustrate the advantages and drawbacks for collecting contextual information from GC , CCF , IC , and MD .

- The declaration of a *global context* allows users to globally overwrite the adaptation context. It does not allow to change the adaptation context fine-grained.
- The *intensional context* can be associated to any element in the input document and thus supports authors to specify context parameters fine-grained.
- Considering *metadata* is analogous to IC but provides a more structured annotation of context parameter: it can associate the meaning of the context dimension and value by referencing the respective content dictionary they have been defined in.
- *Cascading Context Files* permit a granular context specification without changing the input document.

4.4.3. Summary of Example Documents

Given the option $\mathcal{S}_N = \{SD\}$, the below document is returned: the notations reflect the system defaults.

1.1 Binomial Coefficient
 The binomial coefficient $\mathit{binomial}(n, k)$ is the number of ways of ...

2.3 Imaginary Unit
 The imaginary unit $\mathit{imaginary}$ is defined by the property that its square is -1 ...

3.2 Natural Numbers
 The set of natural numbers is either the set of positive integers $\{1, 2, 3, \dots\}$, denoted by posInt , or the set of non-negative integers $\{0, 1, 2, \dots\}$, denoted by $\mathit{nonnegInt}$

The extensional options $\mathcal{S}_N = \{EC, SD\}$ results in the below output. The notation definitions in the document `math-en.doc` have been selected, which includes a collection of notational convention for mathematics and English.

1.1 Binomial Coefficient
 The binomial coefficient $\binom{n}{k}$ is the number of ways of choosing k objects from ...

2.3 Imaginary Unit
 The imaginary unit i is defined by the property that its square is -1 ...

3.2 Natural Numbers
 The set of natural numbers is either the set of positive integers $\{1, 2, 3, \dots\}$, denoted by \mathbf{N} , or the set of non-negative integers $\{0, 1, 2, \dots\}$, denoted by \mathbf{N}_0 ...

Given the extensional options $\mathcal{S}_N = \{Doc, EC, SD\}$, the below document is returned. The Doc reflects the author's notation preferences for specific notations. It initiates that the notation definitions within the input document have a higher priority than the ones in `math-en.doc`.

1.1 Binomial Coefficient
 The binomial coefficient \mathcal{C}_n^k is the number of ways of choosing k objects from ...

2.3 Imaginary Unit
 The imaginary unit j is defined by the property that its square is -1 ...

3.2 Natural Numbers
 The set of natural numbers is either the set of positive integers $\{1, 2, 3, \dots\}$, denoted by \mathbf{N}^+ , or the set of non-negative integers $\{0, 1, 2, \dots\}$, denoted by \mathbf{N} ...

Given the extensional options $\mathcal{S}_N = \{Doc, EC, SD\}$ and the intensional options $\mathcal{S}_C = \{GC\}$, where $GC = \{(\mathit{language} = \mathit{de}), (\mathit{area} = \mathit{physics})\}$, the below output is generated. The notation definitions are selected from the input document and `math-en.doc` depending on how well they match the global context GC .

1.1 Binomial Coefficient
 The binomial coefficient $\binom{n}{k}$ is the number of ways of choosing k objects from ...

2.3 Imaginary Unit
 The imaginary unit j is defined by the property that its square is -1 ...

3.2 Natural Numbers
 The set of natural numbers is either the set of positive integers $\{1, 2, 3, \dots\}$, denoted by \mathbf{N} , or the set of non-negative integers $\{0, 1, 2, \dots\}$, denoted by \mathbf{N}_0 ...

5. Summary & Evaluation of Notations

To illustrate the proposed adaptation services for narrative documents, mathematical documents have been chosen. This decision has required the author to take an essential aspect of mathematical text into account. Mathematics is a mixture of natural language text, symbols, and formulae. Symbols and formulae can be presented with different notations. These notations can complicate communication and acquisition processes since notations are context-dependent and can considerably vary among different communities and individuals. These variations can cause ambiguities and misunderstandings.

When planning the content and structure of a document, the mathematical notations in the document have to be adapted respectively. Such a support for notations is a major concern in proof assistance systems. Unfortunately, literal reuse from these systems will not work as their rendering workflows are commonly tied to their internal data structures. Related approaches on the conversion between the web-accessible formats OPENMATH and MATHML have not yet yielded a satisfying solution to the problem.

Consequently, though notations are an essential aspect for the adaptation of narrative documents, we were not able to fully control them. In this situation, a comprehensive framework was proposed that allows users to configure notations for mathematical documents regarding a semantic, narrative, and user context. To prioritise these contexts and to guide the adaptation, a combination of extensional and intensional options is provided with which users can obtain a context-dependent association of notations with mathematical objects. This gives users full control over the adaptation processes.

The proposed adaptation approach for mathematical notations is based on the notation system specified by [KLM⁺09, KMR08]. The novelty of this notation system is the introduction of jokers as well as the dynamic user-specific selection of notation definitions based on an extensible number of context parameters. Although the approach does not support all possible mathematical scenarios, it covers more than competitive approaches were able to provide. For example, the declarative notation definitions in [KLR07] are limited for the following examples: $\text{sin}(\text{square}(x)) \Rightarrow \text{sin}^2 x$ and $\text{polynomial}(x, 1, 0, -1, 2) \Rightarrow 2x^3 - x^2 + 1$ and $\text{forall}(\text{lambda}(x, f(x))) \Rightarrow \forall x. f(x)$ and $\text{log}(e, x) \Rightarrow \ln x$. These work better with pattern-matching, though $\text{polynomial}(x, 1, 0, -1, 2) \Rightarrow 2x^3 - x^2 + 1$ can also not be covered with the new notation system.

The following section discusses the services of the proposed notation framework and points out limitations as well as issues for future work.

5.1. Services & Limitations

The adaptation approach imposes additional markup efforts on the user. Having written their documents in an XML-based format where all symbols and formulae are represented in OPENMATH or Content-MATHML, authors also have to explicate their notation preferences in form of notation definitions in order to draw on the adaptation services. Such additional efforts will probably not pay off if a user decides to adapt his notations only ones. They

5. Summary & Evaluation of Notations

will rather pay off when maintaining a large collection of documents, which contents are heavily reused and written by numerous authors from different communities and with varying notation practices and preferences. For example, this was a central argument for projects like CONNEXIONS (Section 2.2.2) or ACTIVEMATH (Section 2.2.3) to get involved in the notation topic. In particular, if users want to create a new document (a textbook or lecture notes) from a large, multi-authored collection of documents, adaptation of notations becomes a central issues and helps to avoid notational inconsistencies.

Apart from having chosen mathematics as test tube, the author's initial motivation for addressing the adaptation of notations was based on collaborations with colleagues. In [KLR07, KMM07b] the discussion on notation were started in order to provide interactive services that allow readers to get involved with the rendered content. These included notational services such as flexible elision and folding that were addressed by the SWIM/JOBAD project (Section 2.2.4). A fundamental framework was needed on which these services could be based on and is now proposed by this work. In order to support any adaptation/interaction services, the herein proposed framework has to be integrated into an interactive environment like SWIM, CONNEXIONS, ACTIVEMATH, or *panta rhei* (Section 10). The following sections discuss the adaptation/interaction services introduced in Section 3.4 and outline the required functionality of such an environment.

5.1.1. Alternative Ways of Displaying Symbols

The first group of services of the wish list in Section 3.4.5 refers to alternative ways for displaying symbols, including the presentation of alternative notations, the explanation of differences between notations and the user's preferences, the change of notations while browsing a document, the reading of notations to the user, and the support with natural language terms.

The visualisation of *alternative notations* for a concept requires an encoding of all notation preferences as notation definitions. These are applied to the content representation of the concept (in OPENMATH) and generate all alternative notations (in Presentation-MATHML). According to the design of the user interface, the notations can be displayed as tooltip or popup drawing on technologies such as AJAX¹ and JAVASCRIPT [GLR09].

To *point out notational differences* between the displayed notations and the user's background/preferences, information on the user has to be gathered and maintained (Section 10.3). Given such user information, simple templates can be provided, which are instantiated whenever the user's background/preferences differ with the displayed notation. For example, the template "We write $\langle display \rangle$, but you know it as $\langle user \rangle$ " includes two variables `display` and `user`. It can be instantiated with the displayed notation $\binom{n}{k}$ and the user's preferences C_n^k . The display of the generated text depends on the user interface. For example, it can be shown whenever the user's mouse hovers over the notation $\binom{n}{k}$.

Changing notation on the fly while reading an online document requires a storage of the user's configuration (Section 10.1). The new notation preferences should be applied onto the current document or all documents in the system depending on the user's preferences.

The *reading of notations* to the user requires an integration with a screen reader. Given such a component, the new notation framework (and respective notations definitions) can be used to generated canonical MATHML. This output can be more easily accessed by existing tools [AM06]. In collaboration with experts in the field of aural mathematical support, future work could also address the specification of a new aural framework for mathematics.

¹See [Gar05] for an introduction and the w3c Working Draft on the *XMLHttpRequest Object* [vK09], the major component behind AJAX technology.

The *display of natural language terms* for mathematical concepts can be supported by providing respective notation definitions. Analogously to the display of an alternative notations, the notation definitions can encode a natural language term.

The *panta rhei* system (Section 10.1) supports users in configuring their documents interactively by setting a selection of extensional and intensional notation options. It demonstrates the change of notation preferences on the fly. All remaining display-oriented services (display of alternatives, pointing out differences, reading notations, and display of natural language) are supported by the notation system but have not yet been implemented and tested in an environment.

5.1.2. Explaining the Structure of a Formula

The two structure-oriented services of the wish list in Section 3.4.5, i.e., the flexible display of brackets and the folding of sub-terms, have been specified and implemented in the JOBAD project [KGLZ09, GLR09]. Both services required an extension of notation definitions and additional parameters that guide the rendering process. For example, the rendering of brackets is supported by encoding the precedence level of concepts and expressions in the notation definitions. Depending on the environment and the user's preferences, all brackets or only relevant brackets can be displayed.

One more general, user-adaptive practice that this approach can not offer are *abbreviation*. If a formula is too large or complex to be digested in one go, mathematicians often help their audience by abbreviating parts, which are explained in isolation or expanded when the general picture has been grasped. Abbreviation generation shares many aspects with elisions, but is less structured and more dependent on abilities of the reader. This issue is addressed in the JOBAD project [GLR09].

5.1.3. Using Additional Knowledge

The last group of services of the wish list in Section 3.4.5 addresses the accessing of additional knowledge, beyond a symbol and its notations. They require a markup on higher levels of mathematical knowledge, i.e., the statement and theory level (Section 2.1.5).

The interlinking of symbols and definition has been addressed by the SWiM project, which integrates JOBAD to support a lookup of definition stored in the TNTBASE repository (Section 2.2.4).

The selection of additional information such as informal explanations or examples is addressed by the author. As requested by participants of the requirements study in Section 3.4, this information should be filtered according to the user's background and skills. A reading environment should thus not only support the adjustment of notations but also the selection of user-specific supplementary text paragraphs. The required extension of the notation system are discussed in Part III.

The generation of a guided tour is not discussed in this work. Other research projects implement the first steps for such services. For example, ACTIVE MATH (Section 2.2.3) allows users to insert a list of concepts, which are then explained in a user-specific course document. Future work could address the extension of the content planning approach in Part III for the structuring of guided tours.

5.1.4. Open Research Questions

Not all issues from the requirements specification in Section 3.4 have been addressed by this work. This section outlines issues for further research.

5. Summary & Evaluation of Notations

Consistent Rendering of Notations The notation framework supports the consistent use of notations in documents, which are aggregated from a multi-author document collection. Instead of having to cope with several individual notations, the aggregator can specify his own preferences and thus assure that notations are rendered consistently throughout the document. However, problems arise if the same notation is used for two different symbols.

To support a consistent rendering of such notations, notation definitions can be grouped into blocks of consistent notation systems (called notation documents). These can be used to avoid ambiguities and inconsistencies on the presentation level of mathematical expressions. For example, consider the alternative notations/definitions for the set of natural numbers in Section 3.1: the set of positive integers and the set of non-negative integers can both be denoted by \mathbf{N} , which can lead to ambiguities.

This level of support for consistent notations is not sufficient. It is up to the user to verify that only appropriate notations are collected and inconsistencies in the rendered document are avoided. The notation system has no means to relate or compare the generated notations, e.g., to prevent notations that might conflict with ones that have already been used in the document. The required functionality is specified with the content planning approach in Part III. Future work could extend the notation framework to assure that two mathematical objects are not presented with the same Presentation-MATHML expression.

Modelling Notation Preferences Now that notations have been reified, the design of a flexible management process for notations can be addressed. Notation preferences of authors, aggregators, and readers can be captured to support adaptive notation services (Section 10.3).

Empirical Evaluation An empirical study should evaluate whether adaptation of notation is really beneficial. Some participant of the requirements survey expressed concerns whether the replacement of an author's notations would not destroy his intentions and impair the understandability of his writing. After all, authors spend much time to select appropriate notations that can be easily understood by readers and are accepted by the community.

Future work should also observe how authors can be supported to lock their notations. Currently, adaptation of notations can be prevented if authors include a Presentation-MATHML expressions rather than their content-oriented correspondents. As soon as OPEN-MATH/Content-MATHML are provided, authors can merely make recommendations by using the inline markup options of the rendering workflow. These can be overwritten by readers at their own risks.

Authoring of Notation Definitions This work focused on the representation of notations, the rendering algorithm, and potential services/values for mathematics. An essential part of the presented infrastructure, i.e., the authoring of notation definitions, has not been addressed. This includes research questions such as how to resolve conflicts during collaborative authoring and how to take changes of notation definitions into account. Details on respective specifications and tool support are provided by the SWIM project (Section 2.2.4). The *locutor* project [Mül10] addresses change management and consistency issues.

Generalisation Future work can verify whether a generalisation of the notation framework can be applied to other (even non-technical) disciplines. In general, a symbol can be any object, picture, written word, sound, or particular mark that represents something else by association, resemblance, or convention. Possibly, the notation framework can be extended to support the adaptation of arbitrary conceptualisations.

The user-specific rendering might also be applicable to other rendering processes. Future work could observe whether a generalised approach can support the conversion of documents for various output devices (a smart phone, a tabloid, laptop, etc) or a specific web browser.

5.2. Proof of Concept

5.2.1. The Rendering Prototype

Acknowledgement: *The implementation of the notation algorithm is collaborative work with Dimitar Mišev and Normen Müller. Normen Müller specified and implemented the initial notation framework, in particular, the pattern matching algorithm and command-line client. The framework was extended by the author with a notation collector and rendering grabber. Dimitar Mišev refactored the initial library into JOMDOC and added various extensions.*

To evaluate this approach, the open-source Java Library for OMDOC (JOMDOC [JOM08a]) has been implemented, which provides an API, a command-line frontend, and a graphical user interface, via which OMDOC documents can be parsed, validated, manipulated, and serialised. JOMDOC implements the proposed rendering algorithm independent from the OMDOC format². In particular, it provides

- the context-sensitive conversion of mathematical formulae in arbitrary XML documents by collecting and applying notation definitions, context parameters, and tags,
- the rendering of complete OMDOC documents using XSLT transformations,
- the tracking of all renderings that are applied during the conversion and the preservation of this information in the output document: Attached to the output expression, `tag` elements and `ec` attributes support to capture, which rendering elements have been applied during the conversion. This is particularly useful for interactive features as well as implicit user modelling techniques (Section 10.3).

The JOMDOC library has been successfully integrated into diverse systems, such as the semantic wiki SWIM (Section 2.2.4), the *panta rhei* system (Section 10.1), and lately in the TNTBASE web-interface [ZK09], allowing these systems to display OMDOC documents as adaptable and interactive mathematical documents. The proposed revisions for OMDOC will be integrated into a subsequent versions of the format.

5.2.2. Education Case Study

Acknowledgement: *The evaluation in this section is collaborative work with Michael Kohlhase and Christoph Lange. Michael Kohlhase is the head developer of $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ and an enthusiastic follower of semantic markup. All slides have been marked up by him and are extensively used in his lecture. With the help of Bruce Miller he implemented the conversion from $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ to OMDOC, including the generation of notation definitions from the $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ source.*

In addition to the proof-of-concept prototype, a coverage evaluation was conducted based on the lecture materials for the GENCS lecture. With a total of approx. 1000 OMDOC pages, the GENCS corpus provides a sufficiently large and diverse source to test the coverage of the conversion workflow.

As common in mathematics and computer science, the lecture material is authored in $\mathcal{L}\mathcal{T}\mathcal{E}\mathcal{X}$ rather than an XML-based format. In particular, a semantic extension for $\mathcal{L}\mathcal{T}\mathcal{E}\mathcal{X}$ referred to as $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ is used, which allows to enrich $\mathcal{L}\mathcal{T}\mathcal{E}\mathcal{X}$ with semantic annotations and to convert the slides into OMDOC+OPENMATH, drawing on the $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}\mathcal{M}\mathcal{L}$ conversion process [Mil].

²Note that JOMDOC does not implement the complete conversion workflow, in particular, symbol, variable, and list jokers are not yet supported. However, the respective extension are provided in the MMT system [MMT].

5. Summary & Evaluation of Notations

Currently 0% elisions and 0% pattern matching is used in the authored slides as the developers of \LaTeX have not yet found a way to nicely annotate respective expressions. Only mixfix notations with flexary operators are supported in \LaTeX , from which pattern-based notation definitions can be easily generated via \LaTeXML . Based on the new notation system, the GENCS instructor was able to generate 2206 notation definitions for the GENCS slides and have tremendously improved the Presentation-MATHML output for the embedded notations. The lecturer is particularly happy that no brackets are missing and all brackets are consistently either left out or left in, a task that he has so far not achieved with other conversion workflows.

Note that the GENCS notes do not yet use all of the extensional and intensional option proposed by this work. To the best of the author's knowledge only the *Doc* option is used, i.e., notation definitions are embedded in the slides and applied to display formulae and symbols. The evaluation of the remaining options with the lecture notes remains future work.

Apart from the GENCS case study, all notation definitions in the OPENMATH standard content dictionaries, which cover the K-14 fragment of mathematics, have been reformulated.

5.3. Chapter Summary

An important aspect of mathematical texts are symbols, formulae, and their notations. Notations can complicate communications and acquisition processes since they are context-dependent and can vary among different communities and individuals. This work specified a notation framework that allows users to guide the rendering of symbols and formulae. Consequently, readers can adapt documents post-publications according to their individual preferences or specific conventions of their mathematical field. Technically, the approach encodes presentational characteristics of symbols and formulae declaratively in notation definitions, which consists of prototypes (patterns that are matched against content representation trees) and renderings (that are used to construct the corresponding notation). An elaborated mechanism supports the collection of these notation definitions from various sources and the selection of appropriate renderings according to the user's notation preferences.

With the notation framework, the author has implemented the foundation for interactive, user-specific notation services, such as the display of alternative notations, the explanation of differences between the notations in a document and the user's preferences/background, the reading of notations to the users, the display of natural language terms, definitions, and examples as well as the change of notations on-the-fly while reading a document. Some of these services are already provided by systems such as *panta rhei*, SWIM, and JOBAD.

However, several issues remain for future work. These include the authoring of notation definitions, the consistent rendering of notations, the modelling of notation preferences for adaptive services, as well as the empirical evaluation of the acceptance by users and the impact on the author's intentions. The author believes that the acceptance of the proposed framework and the usefulness of its services can only be achieved in close collaboration with researchers, instructors, and students from mathematics as well as practitioners from industry that aim at applying mathematics for their products. In particular novices might be motivated to exploit some of the described services as they are often challenged in understanding mathematical notations.

Nevertheless, the adaptation of notations can not be seen independently from the adaptation of the whole document, the *content planning* of documents. For example, consider that a student enters his preferred language or area of application. This request should not solely change the notations in the document but should also retrieve alternative paragraphs that best satisfy the user context. In the next section, the rendering algorithm is extended to support the substitution as well as ordering of document parts.

Part III.

Content Planning for Mathematical Documents

6. Introduction

Acknowledgement: First ideas on this chapter are collaborative work with Michael Kohlhase, Achim Mahnke, and Normen Müller and have been published in [KMM07a, KMM07b].

The rise of modern web technologies has tremendously increased the creation of web documents, such as news, advertisements, product descriptions, user manuals, learning materials, software documentations, and scientific publications. Nowadays users no longer face a problem of creating and exchanging these documents but rather struggle to find the appropriate resource in time and to acquire the essential knowledge conveyed in this resource. Adapting these resources to the user context, e.g., skills, backgrounds, and preferences of users, can improve the understandability of web documents.

Adaptation can be provided on all document layers: the content, structure, and presentation layer. The presentation layer, with a focus on the adjustment of notations, was discussed in Part II. The adaptation on the content and structure layer of documents (called content planning) is discussed in this chapter. A focus is placed on the *substitution* of a document's content with alternative, user-specific document parts and the *reordering* of document parts. The author expects that a generalised form of the notation framework from Part II can be reused for the content planning.

Several applications in industry and research address the content planning of documents. Unfortunately, literal reuse from these systems will not work as all of them focus on topic-oriented materials: Cross-references and transitions are omitted and reduce the coherence of the personalised materials. The adaptation of *narrative documents* remains challenging. They can not be easily modularised into self-contained parts that can be arbitrarily combined and arranged according to user-specific constraints.

In this situation, the author proposes an advanced approach for the content planning of narrative documents, which applies the topic-oriented principles of modularisations to the narrative world: Documents are modularised into infoms, for which various transitional texts and cross-references are preserved. These can be traversed to arbitrarily combine and arrange infoms. A focused is placed on the modification of documents rather than their assembly from document parts. Starting of with the original document a new, **variant document** is constructed, in which parts of the original document are rearranged and/or substituted with **alternative parts**. This content planning process can be guided by the the semantic context of document parts, the narrative context of the document, and the user context.

6.1. Representation of Variants/Alternatives

In contrast to the representation of mathematical symbols, formulae, and their notations, there is no unique, standardised representation of variants for markup languages. In the following we introduced the most important approaches.

6. Introduction

6.1.1. Introduction

The borderline between the definitions for ‘variants’ and ‘alternatives’ is not always clear, often both terms are used analogously. An **alternative** is often defined as decision between two or more options or the opportunity to select between two or more things. These choices are alternative objects that can vary in type but target at the same goal. For example, the HTML `alt` attribute is used to specify a text as replacement for an image in a website, whenever the image can not be displayed. **Variants** are sometimes defined as something a little different from others of the same type or something differing from a norm or standard. A variant can be an event that departs from expectations or a differing, deviant, abnormal execution. In the scope of the work, variants provide a notion of two equivalent document parts, which are not identical but differ in certain properties. They become alternatives if they match the semantic, narrative, or user context. Document parts should only be substituted with appropriate alternatives.

The next sections introduce the representation of variants in markup languages and the state of the art on content planning.

6.1.2. Conditional Markup

The first variant documents have been created by technical writers, who realised that components of a document could be reused in another document. As a first approach writers copied and pasted the common paragraphs from the original document into the new (variant) document. Soon they realised that such redundancies could cause inconsistencies in a document repository that were hard to trace.

For variant documents that only differ in few paragraphs, a much better and more maintainable solution was introduced by markup technologies: Only one file is maintained, in which all alternative paragraphs are annotated with conditions that specify when to select one of the alternatives. This technique is often called *conditionalised texts* or *conditional markup* and is supported by markup languages, such as DITA (Section 2.1.1), OPENMATH (Section 3.2), and OMDOC (Section 2.1.5).

DITA provides a markup of alternative sentences in topics. Respective alternatives are annotated with conditions, which specify when the alternative should be included. DITA only supports a number of built-in attributes to specify conditions: `type`, `audience`, `platform`, `product`, `importance`, `revision`, and `status`. To define what conditions apply, rules are provided in separated `ditaval` files (Section 2.1.1).

Listing 26 illustrates the markup in DITA, which can be used in a introductory lecture in mathematics as well as theoretic computer science. The introductory text (“*We will now look at an example.*”) can be reused in both lectures. It is a non-variant or static part. The example is configurable depending on the actual audience. It is a dynamic or variant part. Two alternative examples are provided. One illustrates the proof by induction based on binary trees (a computer science topic) and the other one on natural numbers (a mathematical topic). The conditions are represented by an `audience` attribute with value `compscience` or `math`, respectively. Unfortunately, the condition metadata for describing conditional texts is predefined and can not be extended by users, only `type`, `audience`, `platform`, `product`, `importance`, `revision`, `status` are supported.

Listing 26: Two alternative audiences.

We will now look at an example.

```
<ph audience="compscience">Let us proof that all binary trees ...</ph>  
<ph audience="math">... prove that ... holds for all natural numbers ...</ph>
```

6.1. Representation of Variants/Alternatives

OMDOC supports the markup of conditional text within its statements (e.g., examples, definitions, proofs, etc.) but is limited to the representation of language variants and formal variants. The selection between the variants is currently defined by specifying parameters during the XSLT-based conversion (Section 2.1.5).

Listing 27 illustrate the markup of a multilingual document in OMDOC. The alternative texts are grouped in a `proof` element. The conditions are specified by the `xml:lang` attribute.

Listing 27: Multilingual document.

```
<proof>
  <CMP>Let us proof that ...</CMP>
  <CMP xml:lang="de"> Gegeben ist ...</CMP>
  <CMP xml:lang="fr"> Étant donné ...</CMP>
</proof>
```

The markup of formal variants in OMDOC is based on the specification in OPENMATH. Listing 28 illustrates an extract of an OPENMATH content dictionary, which combines an informal and formal definition for the `equivalent` symbol. The informal, natural language text is embedded in a `CMP` element, while its formalisation $A \equiv B \equiv ((A \implies B) \wedge (B \implies A))$ is contained in the `FMP` element.

Listing 28: OPENMATH CD "logic1" available at <http://www.openmath.org/cd/logic1.occ>

```
<CD xmlns="http://www.openmath.org/OpenMathCD">
  <CDDefinition>
    <Name> equivalent </Name>
    <Description>
      This symbol is used to show that two boolean expressions are logically equivalent, i.e.,
      they have the same boolean value for any inputs.
    </Description>
    <CMP>
      The condition (A is equivalent to B) is equivalent to the condition
      (A implies B and B implies A).
    </CMP>
    <FMP><OMOBJ>
      <OMA>
        <OMS cd="logic1" name="equivalent"/>
        <OMA>
          <OMS cd="logic1" name="equivalent"/>
          <OMV name="A"/>
          <OMV name="B"/>
        </OMA>
        <OMA>
          <OMS cd="logic1" name="and"/>
          <OMA>
            <OMS cd="logic1" name="implies"/><OMV name="A"/><OMV name="B"/>
          </OMA>
          <OMA>
            <OMS cd="logic1" name="implies"/><OMV name="B"/><OMV name="A"/>
          </OMA>
        </OMA>
      </OMOBJ></FMP>
    </CDDefinition>
  </CD>
```

6. Introduction

In the addition to the inline markup of conditional texts, some markup languages support a stand-off markup. For example, DITA provides a grouping of references to alternative topics. Listing 29 illustrates the `linkpool` element, which groups two references (represented with a `link` element) with common type `concept` (they are both concept topics) and common target audience `students` (they are both useful for students). Unfortunately, these groupings can only be used to relate topics, more fine-grained or more coarse-grained document parts can not be associated.

Listing 29: Grouping of variant topics

```
<linkpool type="concept" audience="students">
  <link href="set.dita"/>
  <link href="integers.dita"/>
</linkpool>
```

Current conditional markup approaches are limited to a predefined set of annotations, such as language or formality. They can also not be applied to document parts of arbitrary granularity, e.g., to specify variant definitions, proofs, or exercises. Moreover, none of the markup approaches specifies a workflow that allows users to conveniently guide the selection among the variants.

6.1.3. Markup of Variant Relations

Mathematical markup formats, such as OMDOC, explicate the semantic context of document parts. They mark the inheritance structure of mathematical knowledge, categorise document parts, and mark their dependencies. The latter are sometimes defined in specific ontologies. For example, the OMDOC ontology specifies categories of texts, such as `proof`, `example`, or `definition`, and relations, such as `proves`, `exemplifies`, or `defines` (Figure 9).

Various metadata schemes exist that allow to explicate further properties and relations of document parts. For example, the ACTIVEMATH system (Section 2.2.3) has extended OMDOC with a didactic metadata layer, which is formalised in the ontology of instructional objects [UII08]. Among other this extension allows authors to associate learning objects with alternatives. The latter is represented by the relation `isVariantOf`. Similarly, the WELSA system (Section 2.2.6) uses an extension of the ACTIVEMATH instructional ontology and Dublin Core [Dub] to express the analogy of document parts with the `isAnalogous` relation.

The specification of variant relation in the above markup approaches is limited as properties of the relation (symmetry, transitivity, reflexivity, etc) are not formalised. Respective formalisations would support systems to make inferences in order to enlarge the variant search space as well as to improve the accuracy of the selection (Section 7.2).

6.1.4. Transclusion Mechanism

Instead of mixing variants into a single document, such as proposed by a conditional markup approach, XML technology support the transclusion of document parts: Rather than copying information units into the documents, they are reused from other documents by placing content references in the documents, e.g., using the XML inclusion operator `XINCLUDE` [MOV06] or the OMDOC `ref`-operator (Section 2.1.5). The transclusion mechanism presupposes a modularisation of document into addressable, reusable units.

Based on a fine-grained modularisation of documents, the transclusion mechanism supports variations of documents, e.g., by modifying the arrangement of content references or substituting these pointers. For example, transclusion is used by the ACTIVEMATH course

generator to produce dynamic, user-specific course structures with content references to the most appropriate learning objects (Section 2.2.3).

Although the transclusion mechanism can be applied to document parts of arbitrary size in various files, the content references have to point to *one* specific document part and do not support the selection between *alternative* document content. One solution is to extend the DITA stand-off markup for conditional texts as illustrated in Listing 29. Transclusion could be extended to select one of the `link` elements according to the users preferences (e.g., specified in a `ditaval` file) and to embed the referenced document part.

6.1.5. Dynamic Items

Dynamic items are an extension of the transclusion mechanism and have been specified by the ACTIVEMATH group (Section 2.2.3). Instead of pointing to a specific document part, they encode queries to the ACTIVEMATH course generator. Dynamic items are inserted during the course generation and are not expanded until the learner wants to see the respective document parts. They thus ensure that the course adapts to the adequate model of the learner.

```
<dynamic-item servicename="ExerciseGen" queryname="Train" type="dynamicTask">
  <ref xref="def_diff" />
  <queryparam property="difficulty" value="very_easy"/>
  <queryparam property="competency" value="solve" />
</dynamic-item>
```

The previous listing illustrates the XML representation of a dynamic item (as embedded in the OMDOC pages in ACTIVEMATH), which encloses the task `t=(trainWithSingleExercise def_diff very_easy solve)`. The task and its parameters are encoded in the `queryname` attribute and the sub-elements `ref` and `queryparam`. The former references the concept, which should be trained, and the latter specifies the conditions for the selection of an appropriate exercise.

Dynamic items can also be used by authors. For example, if a lecturer wants to create a document that does not fit in any of the predefined scenarios of the course generator, he/she can manually assemble a document and insert dynamic items, which are automatically filled by the system based on the reader's competencies.

6.1.6. Arrangement of Document Parts

While the substitution of document parts requires a notion of variants, the arrangement is based on the properties of document parts and their dependencies. It has primarily been addressed by topic-oriented approaches. In WELSA (Section 2.2.6), annotations are used to order document parts, where document parts are preferred if they match with the user's learning style best. In ACTIVEMATH (Section 2.2.3), the arrangement of document parts is implemented as hierarchical network planner, where task templates and processing of didactic prerequisites guide the assembly, while the user's educational level and competencies are matched with the annotations of document parts to select the appropriate learning objects. However, neither approach specifies a workflow that allows users to conveniently guide the content planning. The adaptation routines are hidden from the users and solely draw on a predefined set of competency annotations.

6.2. State of the Art

This work defines content planning of documents as the process of substituting and reordering document parts. Starting with the original document a new, variant document is constructed, in which parts of the original document have been rearranged and/or replaced with alternative parts.

Related approaches use document templates, which are filled with user-specific content, and call this process document assembly, document automation, or document modelling (Section 6.2.1). Others focus on the construction of documents from a collection of learning objects and refer to this process as document planning or document generation (Section 6.2.3). All of these approaches generate user-specific documents by selecting the most appropriate resources from a collection of variants and/or by arranging document parts according to the user. Henceforth, the term ‘content planning’ summarises all of these approaches.

6.2.1. Template-based Document Generation

A plethora of application [exa, Hota, IMD, Doc, Thu, MEK] implement a template-based document generation that allows to adapt a document for different communication channels, customer groups, or an individual customer. The templates are optimised for the specific business use case, primarily in the legal, finances, services, and risk management industries. Consequently, templates hard-code predefined scenarios that are relevant to a company or business sector. Typically, respective systems provide a complete workflow that guides the selection of an appropriate document template as well as the assembly of concrete instances by filling the templates with file or case data, while automatically resolving conditional texts, variable texts, and options. The user data is usually collected from forms, e.g., web-based or offline questionnaires.

The document assembly software EXARI [exa] supports the template-based assembly for business documents that are usually done in volume, including contracts, loan agreements, insurance documents, license agreements, finance documents, non-disclosure agreements, service agreements, proposals, and tenders. These templates can be authored in WORD [Cor] using simple text markers to define the automated logic of the document. These markers (also called semantic tags) include

- variables in square brackets: “*This is a contract dated [Contract Date] between [Customer Name] and [Supplier Name]*”,
- optional content: “*[OPT *CustmerUSA* We also offer free delivery to our US customers on purchases over \$500]*”,
- alternative content (or conditional texts): “*[ALT *Warranty-Supplier* Supplier warrants that it will perform its obligations in accordance with the highest professional standards ...] [ALT *Warranty-Mutual* Each party warrants that it will perform its obligations in a professional manner ...]*”.

To gather the variable data for the template, user questionnaires are provided. *Dynamic sub-forms* allow the system to adapt to the user’s answers. A personalisation already takes place in interaction with the interviewee. In addition, answers are recommended based on the entries by other users: a ‘learning mode’ keeps track of which answers are most commonly chosen by other users, taking into account their previous choices. Templates and the data from the questionnaires are used to produced tailored documents. For example, a user entry can trigger that either the `Warranty-Supplier` or `Warranty-Mutual` alternative is included in the document.

HOTDOCS [Hota] is a document assembly software package that helps users to embed variables and simple scripting instructions into legal documents. The result is a pair of files, the template and a matched HOTDOCS component file. When a template is assembled, end users are presented with dynamic, hierarchical interviews, where their entries are stored in XML answer files. A customized document is produced, the answer files can be reused when assembling other templates. HOTDOCS scripting (also called document modelling) ranges from simple variable substitutions to complex systems involving logic, repetitions, insertions, or recursions [Hotb].

Serial letters are documents sent to multiple recipients, which are commonly generated from templates. Address, title, and parts of the content are often personalized and allow the appearance of a personal letter. Text processing systems that support serial letters, e.g., MS Office [Cor], are based on a *database* and a *document template*. The database includes variable data (names, addresses, titles), which are integrated in the template. In addition, the database can provide information to adapt the letter for a particular audience by filtering the list of recipients and by conditionalising the content for better individualisation.

6.2.2. Literate Programming Tools

Conditional markup of formal variants is based on the *literate programming* [Knu92] paradigm. The main idea of literate programming is to regard a program as a communication to human beings rather than as a set of instructions to a computer. It can be interpreted as an alternative approach for maintaining variants in one documents. Here, two variants, the *program code* and its *documentation*, are maintained in the same source document, from which two separate documents, the code and its documentation, can be generated.

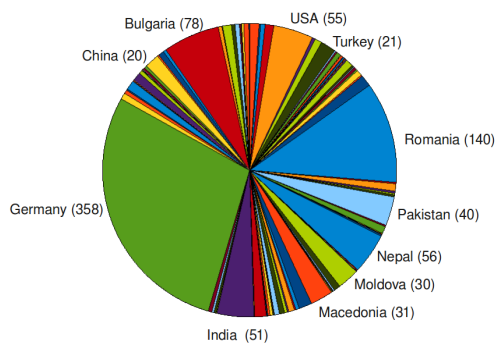
The first programming environment that supported literate programming was WEB [Chi] (and its successor CWEB [KL]), which used Pascal (and later C) as programming language and \TeX as documentation generator. These environment allowed programmers to chop up their programs into chunks, mix them with documentation, and distribute them across a document in arbitrary order [Knu]. Unfortunately, literate programming environments are hardly used in real software engineering. Haskell [Hut07] is one of the few languages that provides native features to support literate programming.

Nevertheless, the concept has been partially adopted by other programming and markup languages: Documentation generators such as DOXYGEN [dox] or JAVADOC [jav] provide a fixed and restricted format for documenting program code. Document markup formats, such as OPENMATH, OMDOC, or MATHLANG, make use of the metaphor of literate programming and allow authors to combine informal content (informal proofs or documentations) and formal mathematical content (formal proofs or code). Technically, these formats apply conditional markup (Listing 28) and transclusion.

6.2.3. Adaptive eLearning Systems

In Section 2.2 several eLearning systems have been analysed. ACTIVEMATH and WELSA exploit didactic metadata for their adaptations services, mathematical relations, properties, and structures are neglected. MATHDOX and PLAT Ω exploit the mathematical structure and dependencies of content but do not consider didactic aspects. CONNEXIONS stores user preferences (e.g., in form of lenses) but does not consider the semantic context of its content. SWiM exploits the semantic context but does not process information on users. None of the system exploits the narrative context of documents for their services. From all content-oriented systems (i.e., systems, which draw on markup techniques), only ACTIVEMATH and WELSA provide user-specific content planning services. They are thus compared to the proposed content planning approach in Section 8 and 9.

6.3. Requirements Specification



With the ever-increasing globalisation of higher education, educational institutions have to cope with culturally induced differences in prerequisite knowledge and learning practices. This is especially pronounced at Jacobs University Bremen with its international student body: 1254 students from 97 countries on March 8th, 2010 [Uni10] (see figure to the left for the distribution).

Surprisingly, this also affects subjects like mathematics and computer science that are often considered culture-independent.

Even though most of the students are well prepared and possess good mathematical knowledge, a needs assessment study [AAS08] revealed mathematical discrepancies. Students of the one-year, introductory course on computer science (GENCS) reported that they had problems to get acquainted with the professor's notation system. Some had the feeling that the pace and difficulty of the course was inappropriate and seemed to be determined by the best students. Some felt embarrassed to ask questions, while others did not face any problems and stated that they were able to balance out based on their previous education. While students of several nationalities struggled with the course, Romanian and Bulgarian students were very confident with their mathematical skills and Indian students were very satisfied with their programming skills. Most students rated these discrepancies as problematic and believed that they can be associated with different educational and cultural backgrounds.

Lecturers from the Open Universitat Oberta de Catalunya [UOC] and the the Open University [OU] reported that they are facing similar problems: In a distance learning setting it is extremely difficult to balance the difficulty of a lecture (or an exam) for the rather anonymous group of students. They believed that a more flexible handling of their lecture material would help them to improve the learning experience of their students. For example, they suggested a tool that would allow them to adapt exams to specific competency levels and backgrounds.

The above observations have encouraged the author to base the illustration of the proposed content planning services on an educational example, in particular, the context-sensitive generation of exams and the ordering of lecture notes according to the semantic, narrative, and user context.

6.3.1. An Exam Generator

This section introduces how the content planning can support the generation of an exam. Let us consider an example. A tutor has to create the final exam for the GENCS lecture 2009 (called `exam2009.omdoc`). The exam should have the same structure as the exam from 2000 (called `exam2000.omdoc`), should omit problems that were used in the exam from 1999 (called `exam1999.omdoc`), should be more difficult than the exam from 2000, and should contain the exercise 'what is a function' problem. Figure 28 illustrates the `exam2000.omdoc`, which contains three sections, where each section contains one exercise: a graph exercise 'number of friends' (called `friends.ex`), a tree exercise 'spanning tree' (called `spanning_tree.ex`), and a function exercise 'what is a function' (called `fun.ex`).

The tutor uses the ACTIVE MATH course generator to generate the exam. He selects the scenario `exam`, provides the three topics `graphs`, `trees`, and `function`, and selects the

6.3. Requirements Specification

time frame for the exam (60 min). ACTIVE MATH generates an exam, which consists of a sequence of randomly arranged interactive exercises on graphs, trees, and functions. As the interactive exercises are automatically verified by ACTIVE MATH, the tutor can even save time in grading the exam. Students can use the system to solve an individualised version of the exam. This requires the students to build up their user models in ACTIVE MATH and does not assure that all students receive the same exam. Instead, the exam adapts to their learner model.

Exam 2000
Written by: Christine Müller

1. Graphs

EXERCISE (ANALYSIS OF PROBLEMS):
Suppose we draw a graph in which each node in the graph is a person and each edge in the graph is a connection between two people who are friends.

In graph the degree of a node is the number of edges attached to the node. Within the graph represent the connections between friends, then the degree is the same as the number of friends.

For each person in the graph above, determine their number of friends.

1. Arin
2. Denny
3. Chis
4. Dorothea
5. Fran
6. Erich
7. Geoff
8. Hanswulf

2. Trees

EXERCISE (CONSTRUCTING TREES):
Can the spanning tree and draw a spanning tree for the below graph.

3. Functions

EXERCISE (WHAT IS A FUNCTION?):
What is a function?

Figure 28.: Exam 2000

Exam 2009
Written by: Christine Müller

1. Graphs

EXERCISE (ALGORITHM FOR SHORTEST PATH):
Write an algorithm to find the shortest path from one node in a graph to all the other nodes. Assume there exists a path from that node to all the other nodes in the graph.

2. Trees

EXERCISE (ALGORITHMS FOR TREE TRAVERSAL):
Write two algorithms that traverses a binary tree (a) in pre-order and (b) in level-order. Compare the two traversal strategies.

3. Functions

EXERCISE (WHAT IS A FUNCTION?):
What is a function?

Figure 29.: Exam 2009

The tutor decides to use the herein described content planning approach. In order to initiate the generation of a new exam, he performs the following steps:

1. Select an exam as reference (or template) for the exams generation. He uses `exam2000.omdoc`.
2. Specify which parts of the exams should be adaptable and which should remain static. The tutor makes the ‘what is a function’ problem static as he wants to include it in every exam.
3. Specify a collection of exercises from which the substitution can select the most appropriate variants from.
4. Specify constraints according to which the new exam is generated. The tutor uses two constraints:
 - (1) Omit all problems in `exam1999.omdoc`,
 - (2) The exam should be harder than `exam2000.omdoc`.
5. Start the generation process.

Figure 29 provides the output of the exam generator. The exam includes two new exercises — the graph problem ‘algorithm for shortest path’ and the tree problem ‘algorithm for tree traversal’, which satisfy the tutor’s constraints: they are not contained in `exam1999.omdoc`

6. Introduction

and are more difficult than `exam2000.omdoc`. The ‘what is the function’ problem is also included. Section 8 illustrates *how* this output is computed and explains why the graph and tree problem were chosen.

Note that this work does not aim at competing with intelligent tutoring system but rather uses the exam generation example for illustration purposes. In contrast to the ACTIVEMATH system, which focusses on interactive exercises and automatic verification, it solely handles static exercises (or static document parts in general). Consequently, the user can not draw on automatic verifications of the exercises. Optimisations according to a given time frame are also not supported.

6.3.2. Ordering Lecture Notes

This section introduces how the content planning can support the ordering of lectures notes. Let us consider an example. To prepare for an exam, a student wishes to adapt the GENCS lecture notes 2009. She does not want to allow any substitutions as she is afraid to miss out any of the material. Instead, she hopes to gain better intuitions on the lecture by ordering the notes according to different criteria, such as the mathematical inheritance structure of the document’s parts as well as specific content preferences (she likes to see images and examples first before studying definitions and algorithms).

The student first tries the WELSA system as she knows that it only recommends learning object without omitting any content. Soon she realises that WELSA can not adapt coarse-grained parts of the document but solely provides an ordering of the leaves of a document structures, i.e., of learning objects. It can also not order document parts according to their dependencies but solely considers specific properties, such as the instructional role, media type, level of abstractness, or formality. Moreover, she is not in control of the adaptation result as WELSA solely considers her user model and can thus not explore different ordering strategies.

1.3.2. Algorithms

DEFINITION:

An algorithm is a collection of formalised rules that can be understood and executed, and that lead to a particular endpoint of results.

EXAMPLE: (Pseudocode for Kruskal's algorithm)

```

1 function Kruskal(G)
2   for each vertex v in G do
3     Define an elementary cluster C(v) = {v}.
4   Initialize a priority queue Q to contain all edges in G, using the weights as keys.
5   Define a tree T = ∅. //T will ultimately contain the edges of the MST
6   //n is total number of vertices
7   while T has fewer than n-1 edges do
8     //edge u,v is the minimum weighted route from/to v
9     (u,v) = Q.removeMin()
10    //prevent cycles in T: add u,v only if T does not already contain a path between u and v.
11    //Note that the cluster containing more than one vertex only if an edge containing a pair of
12    //the vertices has been added to the tree.
13    let C(u) be the cluster containing u, and let C(v) be the cluster containing v.
14    if C(u) ≠ C(v) then
15      Add edge (u,v) to T.
16      Merge C(u) and C(v) into one cluster, that is, union C(v) and C(u).
17  return Tree T

```

EXAMPLE: (Kruskal's algorithm, a graph algorithm for spanning trees)

Randomly add a pair to the tree if it won't create a cycle. Repeat until a spanning tree has been created

EXAMPLE: (Visual Example of Kruskal's algorithm, a graph algorithm for spanning trees)

1.3.3. A spanning tree

DEFINITION:

A spanning tree T of a connected, undirected graph G is a tree composed of all the vertices and some (or perhaps all) of the edges of G . That is, every vertex lies in the tree, but no cycles (or loops) are formed.

DEFINITION:

A spanning tree of a connected graph G can also be defined as a maximal set of edges of G that contains no cycle, or as minimal set of edges that connect all vertices.

EXAMPLE: (Visual Example of a Spanning Tree)

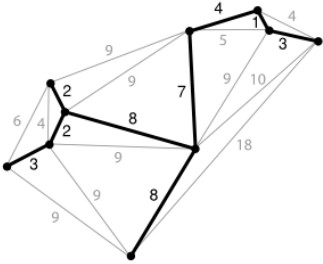
Figure 30.: The algorithm and spanning tree sections of GENCS 2009

The student decides to use the herein described content planning approach. Due to the limited amount of space only the two sections in Figure 30 are considered for the further illustrations — the *algorithm* and *spanning tree* section. The slides were created based on the constraints of the student’s professor, who prefers definitions and algorithms. Consequently, the two sections first provide the definitions, then the examples with algorithms, and finally examples with text and images. To initiate the reordering of the lecture notes, the students performs the following steps:

1. Select the document. She uses the GENCS notes 2009.
2. Specify which parts of the document should be adaptable and which should remain static. The student is very careful and does not want to miss out on any content. She just wants those notes to be reordered, which the lecturer marked as sortable.
3. Specify constraints according to which the document is reordered. The student has two constraints:
 - a) Visual parts and examples should be placed before definitions and algorithms.
 - b) The parts of the document should be ordered according to the mathematical dependencies: the *algorithm* and *spanning_tree* theory are not arranged according to their mathematical dependencies. The student wants them to be switched.
4. Start the reordering process.

1.3.2. A spanning tree

EXAMPLE: (Visual Example of a Spanning Tree)



DEFINITION:

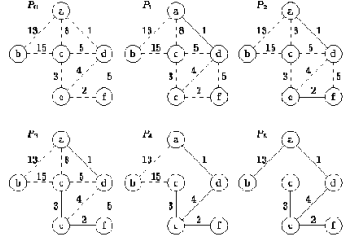
A spanning tree T of a connected, undirected graph G is a tree composed of all the vertices and some (or perhaps all) of the edges of G . That is, every vertex lies in the tree, but no cycles (or loops) are formed.

DEFINITION:

A spanning tree of a connected graph G can also be defined as a maximal set of edges of G that contains no cycle, or as minimal set of edges that connect all vertices.

1.3.3. Algorithms

EXAMPLE: (Visual Example of Kruskal’s algorithm, a graph algorithm for spanning trees)



EXAMPLE: (Kruskal’s algorithm, a graph algorithm for spanning trees)

Randomly add a pair to the tree if it won’t create a cycle. Repeat until a spanning tree has been created

EXAMPLE: (Pseudocode for Kruskal’s algorithm)

```

1 function Kruskal(G)
2   For each vertex v in G do
3     Define an elementary cluster C(v) = {v}.
4   Initialize a priority queue Q to contain all edges in G, using the weights as keys.
5   Define a tree T = ∅ // T will ultimately contain the edges of the MST
6   // n is total number of vertices
7   while T has fewer than n-1 edges do
8     // edge u,v is the minimum weighted route from/to v
9     (u,v) = Q.removeMin()
10    // prevent cycles in T, add u,v only if T does not already contain a path between u and v.
11    // Note that the cluster contains more than one vertex only if an edge containing a pair of
12    // the vertices has been added to the tree.
13    Let C(v) be the cluster containing v, and let C(u) be the cluster containing u.
14    If C(v) ≠ C(u) then
15      Add edge (u,v) to T.
16      Merge C(u) and C(u) into one cluster, that is, union C(v) and C(u).
17   return tree T
    
```

DEFINITION:

An algorithm is a collection of formalised rules that can be understood and executed, and that lead to a particular endpoint of results.

Figure 31.: The adapted lecture notes 2009

Figure 31 illustrates an extract of the reordered course. The section algorithm and spanning tree have been switched. Moreover, the text paragraphs in both sections have been rearranged: First, examples with images are shown, then text examples, then algorithms (or examples with pseudo-code), and finally the definitions. Section 9 illustrates *how* the lecture material can be reordered.

6. *Introduction*

7. The Content Planning Approach

This work proposes an approach for the content planning of narrative, mathematical documents, which assures the coherence of the adaptation result by preserving their connectedness via transitions and cross-references. The most important prerequisites for such a service is a representation, which unlocks and structures the content of these documents so that they are comprehensible to a computer system. XML-based markup languages, in particular, the OMDOC format, are exploited to represent the semantic context of mathematical documents. Unfortunately, OMDOC does not yet mark the transitional texts in a document. The coherence of adaptation results is at risk, if transitional texts and cross-references are not reflected by mathematical dependencies between the document parts (Section 1.3).

To overcome the limitation of (mathematical) document formats, this work applies the topic-oriented principles of modularisation and reuse to the narrative world. In Section 1.4.2 we have learned that a topic-oriented approach can not be directly applied to narrative documents. These can not be sufficiently decomposed into (narratively) self-contained units. Instead, the adaptation of narrative documents requires a new adaptation model: Narrative documents are modularised into *infoms*, for which all transitional phrases and cross-references are explicitly marked (see (1) in Figure 32). Infoms and their dependencies are modelled as dependency graph (see (2) in Figure 32). Two dependency types are distinguished: semantic dependencies and narrative dependencies; in Figure 32 the former are presented with solid arrows between the infoms and the latter with dashed ones. These can overlap and oppose each other. Narrative dependencies are *dynamised* by enriching infoms with alternative transitions and cross-references (see (3) in Figure 32). The narrative dependencies between document parts are thus no longer static but rather depend on the combination and arrangement of document parts in the user-specific document.

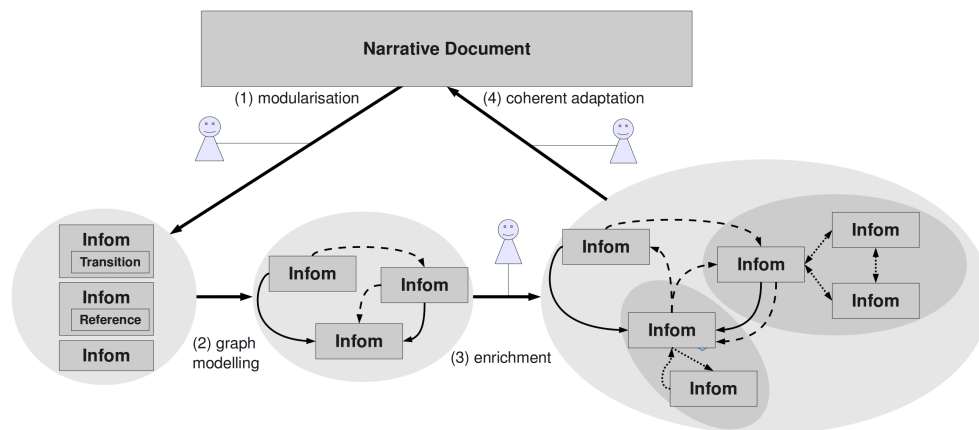


Figure 32.: Content planning of narrative documents

The dependencies graph is enriched with another relation type, called variant relations. In Figure 32 these are presented with dotted arrows between the infoms. A **variant relation** associates two document parts, which are equivalent and can be substituted with each

7. The Content Planning Approach

other. While narrative dependencies allow an adaptation engine to vary the arrangement of document parts, variant relation support the variation of document content by substituting a document part with an equivalent but user-specific one (i.e., a variant). To guide the substitution and arrangement of the modularised document parts into narrative documents, users can prioritise the semantic, narrative, and user context (see (4) in Figure 32).

The following section describes the *modularisation* of narrative (mathematical) documents. It uses the GENCS lecture notes of an introductory computer science course at Jacobs University for illustration purposes. The lecture was enriched with alternative transitions and cross-references, thus, allowing adaptation engines to generate user-specific documents with varying arrangements. In order to support documents with varying content, Section 7.2 introduces the specification and markup of *variants* and *variant relations*. Section 7.3 introduces the *terminology* for the substitution algorithm and reordering algorithm.

7.1. Modularisation of Mathematical Documents

Mathematical markup language, such as OMDOC (Section 2.1.5), support the modularisation of mathematical documents into dependency graphs, where nodes correspond to addressable document parts and edges denote their *semantic dependencies*.

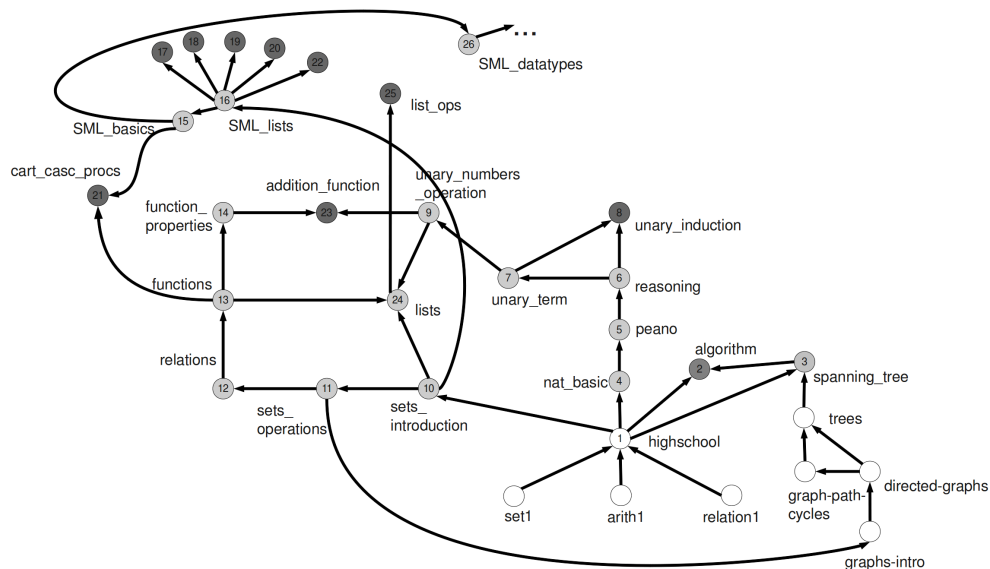


Figure 33.: Document parts and their dependencies in the GENCS material

Figure 33 illustrates the dependency graph of an extract of the GENCS course. The numbered nodes correspond to theories of the lecture notes, the unnumbered nodes are either prerequisites from previous education or subsequent theories that are covered in the advanced GENCS lecture. The latter include the `trees`, `graph-path-cycle`, `directed-graphs`, and `graphs-intro` theories. The theory `highschool` (1) summarises all prerequisites of the course and depends on several fundamental theories. These fundamental theory nodes have an *in-degree* of zero and are thus independent from other nodes. They are considered as the entry points into the lecture graph. Since these fundamentals are not covered in the course, the `highschool` theory is an artificial start

7.1. Modularisation of Mathematical Documents

node for the course. An alternative start node is `SML_basic` (15). All nodes that have an *out-degree* of zero are not required by other nodes in the course and are possible end-points for the course. In the small GENCS example, these are the nodes `algorithm` (2), `unary_induction` (8), `addition_function` (23), and `list_ops` (25), and the SML theories (17, 18, 19, 20, 21, 22). All other theories either depend on other theories or provide the prerequisites for another theory.

To produce a document-centered display, the dependency graph has to be linearised into a sequence of nodes (called a **document path**, short **path**). Ideally, the path is constructed so that from each node on the path there is an edge to the succeeding node, i.e., all nodes on the path are connected via dependencies. The main goal of the content planning approach is to *traverse* the dependency graph of a document along the dependencies between document parts to find a user-specific path. During the traversal, dependencies are used as **transitions** between document parts. From now on the term ‘dependency’ and ‘transition’ are used analogously. We recapitulate the terminology introduced in Section 1.3, respectively: A **transition** associates a supporting document part (where the edge is outgoing) with a dependant document part (where the edge is incoming). The **supporter** is required in order to disambiguate and understand the information conveyed in the **dependant**. When arranging document parts in a document, the supporter has to be placed before the dependant.

We have learned that narrative documents (in contrast to topic-oriented documents) include numerous transitional phrases that guide the reader and produce a coherent flow through the document. Consequently, in addition to the **semantic transitions**, we can consider **narrative transitions**, which bridge two document parts that do not necessarily relate via a semantic dependency. Such transitions reflect the narrative practice of the author and are visualised with transitional words and phrases.

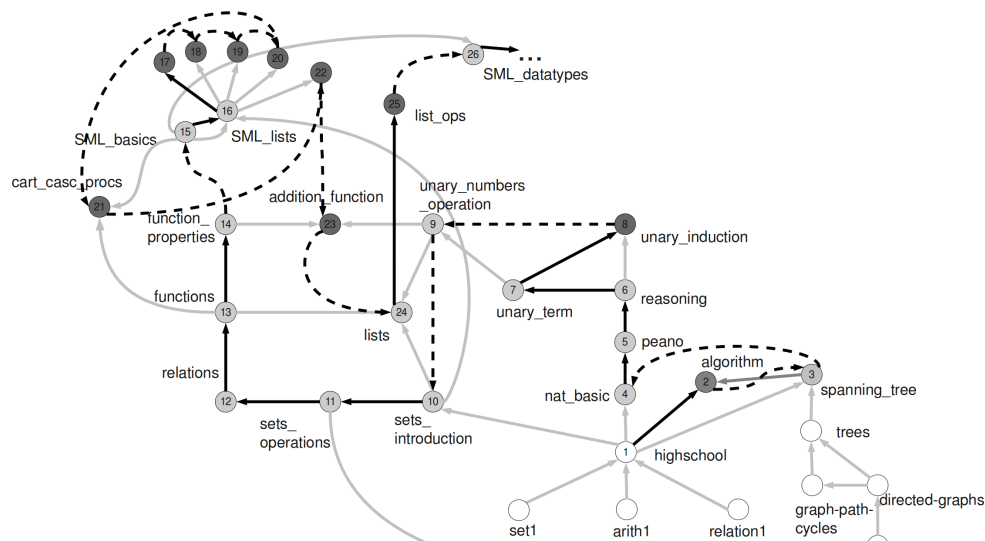


Figure 34.: GENCS graph with semantic and narrative transitions

Figure 34 illustrates the GENCS graph (as specified by the instructor) with semantic and narrative transitions. The solid arrows denote semantic transitions. The dashed arrows mark narrative transitions. The path of the GENCS lecture notes through the graph is highlighted. It starts with the `highschool` theory (1). Then the `algorithm` theory (2) is visited, which includes several examples that illustrate the construction of an algorithm for comput-

7. The Content Planning Approach

ing a spanning tree. The theory thus depends on the `spanning_tree` theory (3), which is briefly introduced afterwards. One could argue that it is a better practice to first introduce all (semantically) required concepts before visiting the next theory in the graph. In this case, the lecturer uses examples with old and new concepts as narrative transitions between two theories for didactic reasons. For all other theories in the course, the semantic prerequisites are introduced beforehand. For example, `addition_function` (23) requires the theories `unary_numbers_operation` (9) and `function_properties` (14). The lecturer chooses to first cover all theories on the path to the former theory (4, 5, 6, 7, 8, 9), then adds a narrative transition to the theory `sets_introduction` (10), and continues to introduce all theories on the path to `function_properties` (10, 11, 12, 13, 14). Although all semantic prerequisites are now provided to cover `addition_function` (23), the lecture first introduces basic SML concepts (15–22). Another narrative transitions leads back to the `addition_function` (23). The path continues with `lists` (24) and `list_ops` (25) before covering more advanced SML concepts (26,...).

The path in Figure 34 includes 26 transitions, of which 15 are narrative and 13 are semantic. 2 narrative transitions overlap with semantic transitions and 11 narrative transitions are associated with transitional texts, which are summarised below.

- 1 to 2 (semantic), 2 to 3 (narrative): no text
- 3 to 4 (narrative): “*We have seen in the last section that we will use mathematical models for objects and data structures throughout computer science. As a consequence, we will need to learn some math before we can proceed. Let’s start with the math! Discrete math for the moment.*” The lecturer first uses examples (the algorithms on spanning trees) to underline his intention of covering a lot of math in his course. He uses a transition to emphasise this aspect for the students: it summarises the previous sections and introduces the next one.
- 4 to 5 (semantic), 5 to 6 (semantic), 6 to 7 (semantic), 7 to 8 (semantic), and 8 to 9 (narrative): no text
- 9 to 10 (narrative): “*On our way to understand functions. We need to understand sets first.*”
- 10 to 11 (semantic), 11 to 12 (semantic), 12 to 13 (semantic), 13 to 14 (semantic): no text
- 14 to 15 (narrative): “*Enough theory, let us start computing with functions. We will use Standard ML for now*”. This transition reflects the the lecturer’s intention. He chose a detour over the basic SML concepts before going back to the ‘addition function’ (23). Maybe the lecturer wants to attract the attention of the computer science students, which are more interested in programming concepts. Alternatively, he might want to prevent to overload the students with mathematical fundamentals.
- 15 to 16 (semantic, narrative): “*What’s next? More SML language constructs and general theory of functional programming.*”
- 16 to 17 (semantic, narrative): “*In fact, the phenomena of recursion and inductive types are inextricably linked, we will explore this in more detail below.*”
- 17 to 18 (narrative): no text
- 17, 18 to 19 (narrative): “*Defining functions by cases and recursion is a very important programming mechanism in SML language. At the moment we have only seen it for the built-in type of lists. In the future we will see that it can also be used for user-defined data types. We start out with another one of SML languages basic types: strings.*”
- 19 to 20 (narrative): “*The next feature of SML is slightly disconcerting at first, but is an essential trait of functional programming languages: ...*”

7.1. Modularisation of Mathematical Documents

- 20 to 21 (narrative), and 21 to 22 (narrative): no text
- 22 to 23 (narrative): “Now that we know some SML. What does this all have to do with functions?”
- 22 to 23 (narrative): “Let us now go back to look at concrete functions on the unary natural numbers. We want to convince ourselves that addition is a (binary) function. Of course we will do this by constructing a proof that only uses the axioms pertinent to the unary natural numbers: the Peano Axioms.”
- 23 to 24 (narrative): “As we have identified the necessary conditions for proving function-hood, we can now generalize the situation, where we can obtain functions via defining equations: we need inductively defined sets, i.e., sets with Peano-like axioms.”
- 24 to 25 (semantic): no text
- 25 to 26 (narrative): “Now, we have seen that inductively defined sets are a basis for computation, we will turn to the programming language to see them at work.”
- 26 to ... (semantic): no text

After observing the transitional phrases, it can be concluded that they are an important mean to improve the coherence of the document. They summarise sections in the course, provide reasons why certain concepts are covered and how concepts relate, and can be combined to build up more complex structures. Omitting these texts would severely reduce the quality of the course material. Transitional texts can only be left out if the materials neatly intertwine, e.g., as in the sections on natural numbers and induction (4, 5, 6, 7, 8, 9) or on basic SML language concepts (14, 15, 16, 17, 18, 20, 21, 22).

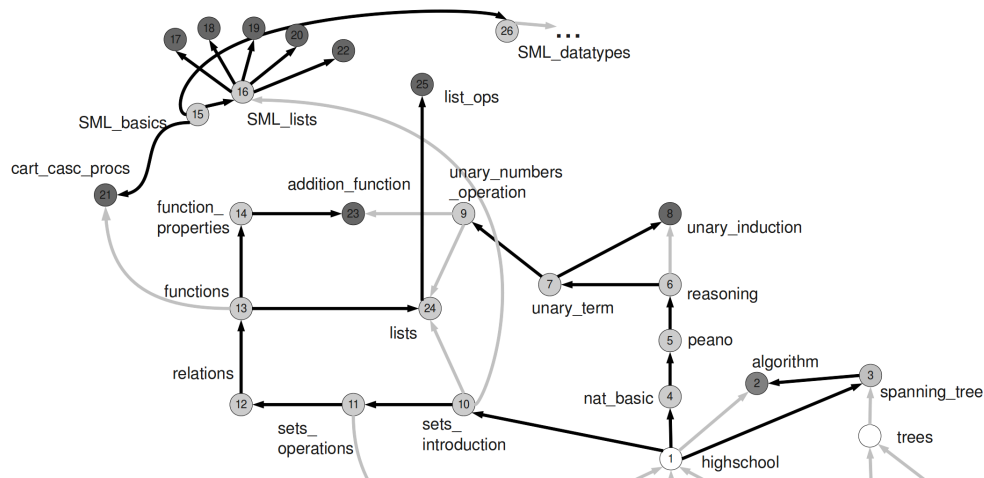


Figure 35.: GENCS graph traversal

Figure 35 illustrates an alternative traversal of the GENCS graph along solely semantic transitions, narrative transitions are omitted. The graph is traversed based on the algorithm in Section 9.3.2 and the path 1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 24, 25, 14, 23, 15, 16, 22, 20, 19, 18, 17, 26, ..., 21 is constructed.

The alternative sequence does not produce a better flow through the course material. As we have seen above, the material consists of a lot of informal texts, which reflect narrative transitions. Although the material is ordered with respect to the marked up dependencies, the

7. The Content Planning Approach

resulting flow of material is of very poor quality. Many transitional phrases are now inappropriate and the flow of the course is destroyed. Consequently, solely considering semantic transitions to reorder a document is not sufficient.

7.1.1. Representation of Discourse Structures

Apart from transitions, the structure layer of documents has to be considered. Usually documents like the GENCS lecture notes are not a sequence of document parts but are *structured*, i.e., the content is grouped into chapters, sections, and subsections. This structure is henceforth called the **discourse structure** of a document. The constituents of the discourse structure, such as chapters, sections, and subsections, are called **discourse containers**.

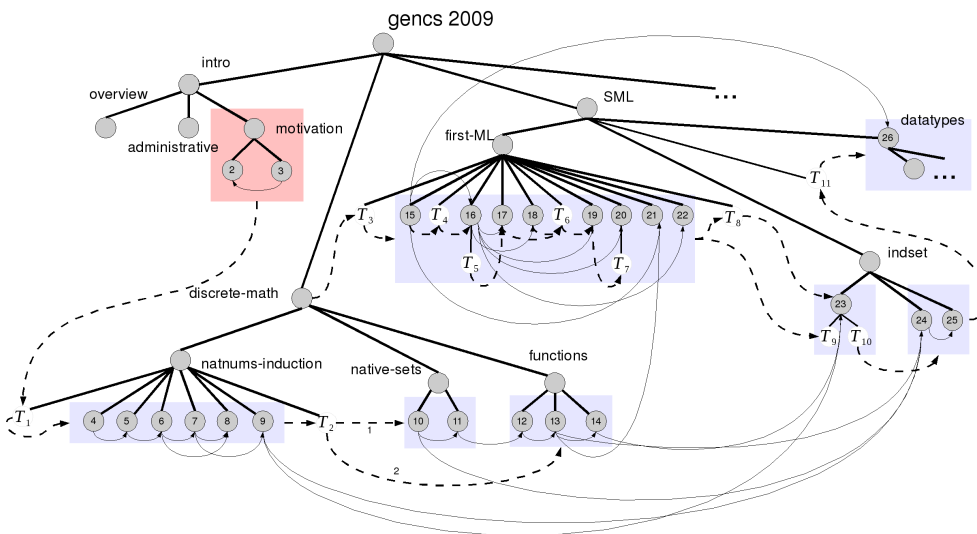


Figure 36.: The discourse structure of the GENCS notes

Figure 36 illustrates the discourse structures of the GENCS example. The solid arrows represent the semantic transitions, the dashed arrows represent narrative transitions. The example document includes three sections, i.e., `intro`, `discrete-math`, and `SML`. The `discrete-math` section includes the subsections `natnums-induction`, `native-sets`, and `functions`, which each embed the document parts in Figure 33.

Listing 30 illustrates the representation of discourse containers in OMDOC (Section 2.1.5). The `omgroup` element with an optional attribute `type` explicates a document structures. The `type` attribute can take values such as `sequence` for a succession of paragraphs, `itemize` for unordered lists, `enumeration` for ordered lists, or `sectioning` for chapters, sections, and subsections. Alternatively, the `tgroup` element can be used to structure OMDOC documents [Koh06, chapter 15.6]. Each element in OMDOC can carry an `xml:id` attribute [MVW05], which supports the unique addressing of the element.

Listing 30: Representation of discourse containers in OMDOC

```
<omdoc ...>
  <omgroup>
    <omgroup type="enumeration">...</omgroup>
    <omtext>...</omtext>
  </omgroup>
</omdoc>
```

```

<theory xml:id="t1">
  <tgroup>...</tgroup>
  <tgroup>...</tgroup>
</theory>
<omgroup type="slide">...</omgroup>
</omgroup>
</omdoc>

```

Given a markup and addressing scheme for discourse containers, they can also be included in the dependency graph. The nodes of a dependency graph can thus be of any granularity: from chapters, sections, and subsections, to mathematical theories, to paragraphs and single phrases. Moreover, transitions can not only be associated between nodes of the same type but also between different types of document parts, e.g., a section and a theory. Often document parts are highly semantically interconnected. Transitions can thus also link sets and sequences of document parts.

In Figure 36, all nodes with label T denote narrative transitions and connect discourse containers and their constituents. For example, T_1 connects the discourse container *motivation*, which includes the theory set 2, 3, with the set of theories $\{4, 5, 6, 7, 8, 9\}$ in the *natnums-induction* section. The second transition T_2 in *natnums-induction* connects the set $\{4, 5, 6, 7, 8, 9\}$ with a sequence of discourse containers, which correspond to the sections *native-sets* and *function*. The *native-sets* section has to be discovered next and the *function* section immediately afterwards.

7.1.2. Deriving Transitions from Semantic Markup

In order to consider semantic or narrative transitions between document parts, they need to be made processable. A respective markup should not replace existing markup for narrative and semantic relations. In the following, such markup for the OMDOC format is illustrated.

Narrative prerequisites and consequence are not represented by current markup formats (Section 2.1). Some XML elements, such as links or citations markings, could be used to infer them. However, they can not be associated with words and phrases and can not express more complex transitions such as identified above. For example, some transitions presume two previous sections (from 17, 18 to 19), while others define a specific sequence of sections (from sequence 4, 5, 6, 7, 8, 9 to section 10, 11 and then to section 12, 13, 14). Current markup formats do also not support a markup of alternative transitions.

Semantic transitions are a special case of narrative transitions. They solely represent a dependency between two document parts and are already thoroughly marked by semantic document formats such as OMDOC. In OMDOC, the prerequisites of theories are indicated by theory morphisms, which are represented by an `imports` element. The prerequisites of statements, such as examples, definitions, or proofs, are referenced by a `for` attribute. The type of relation is specified in the OMDOC ontology (Section 2.1.5) and can be mapped to a dependency. For example, an `example` element carries a `for` attribute to associate it with a `definition` element. The relation between both statements is of type `illustrates` and denotes that the definition is required to understand the example: the definition acts as supporter and the example as dependant.

In order to map relations to their dependencies, we can build on the MATHLANG approach (Section 2.1.4). Accordingly, we first extract the *relation graph* from the document, where nodes represents structural units (the MATHLANG term for ‘document part’) and edges represent their relations. For each kind of relation in the graph, we computes its dependency. These are used to generate a *dependency graph*, where nodes correspond to the nodes in the relation graph and edges denote the computed dependencies.

7. The Content Planning Approach

In the following we illustrate the modelling of a dependency graph for the example document in Section 1.3. Figure 37 presents a relation graph. Table 3 presents the relations and the corresponding semantic dependencies (or semantic transitions). These are used to generate the dependency graph in Figure 38: the nodes in the dependency graph correspond to the nodes in the relation graph. The edges represent the semantic transitions from one node to another one and correspond to the relations in the relation graph.

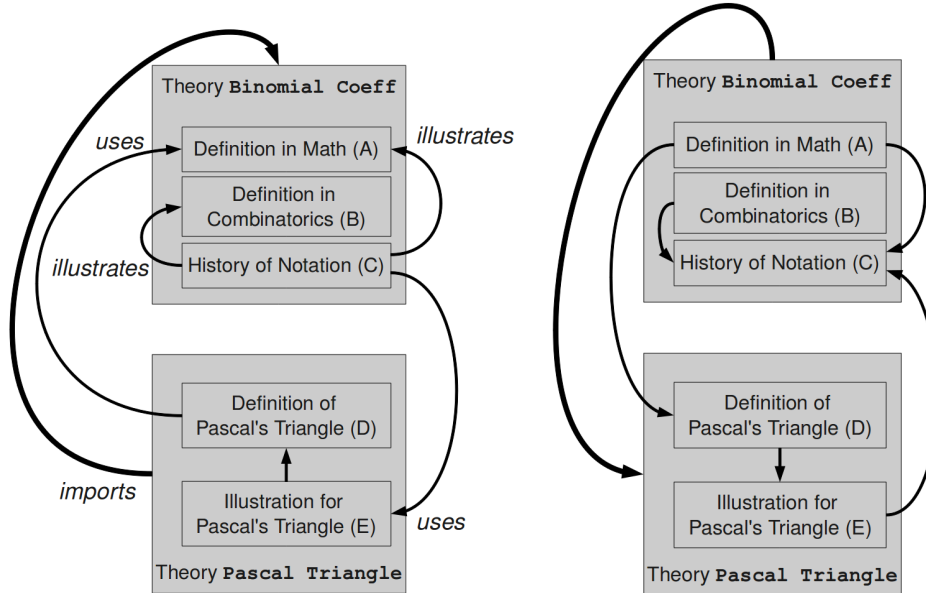


Figure 37.: Relation graph

Figure 38.: Dependency graph

Relation	Meaning	Dependency/Transition
A uses B	A uses a part of B , e.g., a statement of a symbol or figure.	$B \rightarrow A$
A illustrates B	A exemplifies parts of B .	$B \rightarrow A$
A imports B	A imports symbols defined in B .	$B \rightarrow A$

Table 3.: Semantic relations and dependencies between document parts

7.1.3. Representation of Transitions and Walks

To integrate transitions into XML-based document formats, the following markup is proposed¹: A `transition` element is introduced, which has to be associated with an infom. Each transition element carries a `type` attribute, denoting whether it is a narrative or semantic transition and has two children: a `prerequisites` element and a `consequences` element. Both children may contain a number of child element with label `prerequisite` and `consequence`, respectively, which reference an infom with a `xref` attribute. The `prerequisites` element and `consequences` elements carry an optional `order` attribute to denote whether they represent a set or sequence of references. To guide the selection among transition elements, they can be associated with context parameters, represented with the `ic` attribute or `metadata` element.

¹The transitions and their constituents are defined in Section 7.3

In addition to the markup of transitions, all transitional texts have to be explicitly marked. Transitional text can be represented by any kind of infom, e.g., an `omtext` element but also `definition`, `example`, or `omgroup` elements. These infoms are called transition infoms. To associate a transition infom with a `transition`, the `transition` element has to carry a unique `xml:id`. The `for` attribute is used to indicate that an infom is associated with a transition. The value of the `for` attributes is a white-space separated list of references, thus, elements can be associated with more than one transition. Thanks to the markup of transitions and transitional texts, transitional phrases and cross-references can be separated from the reusable parts of a document: The transitional texts of an infom are referred to as its **variable part** and the remaining content as its **reusable part**. By removing the variable parts from the infom it can be turned into a **narratively self-contained** information unit. In order to fit it in the narrative flow of a document, one transition can be selected and its transitional texts can be preserved, while all others are hidden. By default, all infoms are displayed in the document. An infom is hidden if its `display` property is marked as `false`. The `display` property can be represented with an `ic` attribute or `metadata` element.

Listing 31: XML representation of transitions and transitional texts

```
<omdoc ...>
  <theory xml:id="algorithm">
    <definition for="alg"><CMP>An algorithm is ...</CMP></definition>
    <omtext type="transition" for="#n1">
      <CMP>We have used the term spanning tree without defining it. Let us do that now.</CMP>
    </omtext>
    <transition type="narrative" xml:id="n1">
      <prerequisites />
      <consequences>
        <consequence xref="#spanning_tree" />
      </consequences>
    </transition>
    <transition type="semantic">
      <prerequisites>
        <prerequisite xref="#spanning_tree" />
      </prerequisites>
      <consequences />
    </transition>
  </theory>
  <theory xml:id="spanning_tree">
    <omtext type="transition" for="#n1">
      <CMP>Having seen some algorithms on spanning tree, we will define the concept.</CMP>
    </omtext>
  </theory>
</omdoc>
```

Listing 31 illustrates the proposed markup. The document includes two theories, `algorithm` and `spanning_tree`. The `algorithm` theory contains two `transition` elements. The first transition is `narrative`. It has no `prerequisites`, thus, no other document part has to be placed before the `algorithm` theory. The `consequences` element requires that the theory `spanning_tree` is the next part of the document. The `omtext` elements in both theories are associated with the transition `n1`, the value of their `xref` attribute references the transition. Alternative narrative transitions can be added into the `transitions` environment, indicating alternative flows through the document. The second transition is `semantic` and specifies that the `spanning_tree` theory is visited before

7. The Content Planning Approach

the `algorithm` theory.

As alternative to the transition markup that has to be associated with a specific element, users can use a markup for semantic and narrative walks. For this, a `walks` element is introduced, which groups a number of `walk` elements. Each `walk` element carries a `type` attribute, denoting whether it is a semantic or narrative walk, and includes a sequence of `step` elements. A `step` element references other infoms, which are part of the walk, with its `xref` attribute. The `xml:id` of the walk element can be referenced by the transitional texts in a document, indicating that they are associated with the walk. To guide the selection between walks, `walk` elements can be described by a `ic` attribute or `metadata` element.

Listing 32: XML representation of narrative and semantic walks

```
<omdoc ...>
  <theory xml:id="algorithm">
    <definition for="alg"><CMP>An algorithm is ...</CMP></definition>
    <omtext type="transition" for="#n1">
      <CMP>We have used the term spanning tree without defining it. Let us do that now.</CMP>
    <omtext>
  </theory>
  <theory xml:id="spanning_tree">
    <omtext type="transition" for="#n1">
      <CMP>Having seen some algorithms on spanning tree, we will define the concept.</CMP>
    <omtext>
  </theory>
  <walks>
    <walk type="narrative" xml:id="n1" ic="audience:cs">
      <step xref="#algorithm" />
      <step xref="#spanning_tree" />
    </walk>
    <walk type="semantic" xml:id="s1" ic="audience:math">
      <step xref="#spanning_tree" />
      <step xref="#algorithm" />
    </walk>
  </walks>
</omdoc>
```

Listing 32 illustrates the proposed markup. The document includes two walks, a narrative walk `n1` and a semantic walk `s1`. The former first visits the `algorithm` theory and then the `spanning_tree` theory, The latter traverses the reversed order. Both walks are described with an `ic` attribute, which indicates that the former is preferred by the `cs` audience and the latter by the `math` audience. If the latter is chosen, the `omtext` elements associated with `n1` have to be hidden to assure a coherent document.

7.2. Variants & Variant Relations

The modularisation of documents into infoms and transitions is a precondition for any content planning service. A markup of transition rules is primarily needed for the reordering of document parts (Section 9). For the substitution, these rules are processed to identify if a document part can be removed from a document: supporters for infoms in the document cannot be removed. They are also required to analyse whether a document part can be inserted into a document: dependants can not be inserted if their supporters are not part of the document. In addition, a *notion of variants* assures that only appropriated document parts are considered for the substitution.

7.2.1. Introducing Variants & Variant Relations

Variants are defined as *equivalent* infoms that *differ* in (a few) properties and relations. They become alternatives if they match a given semantic, narrative, or user context. Infoms should only be substituted with appropriate alternatives.

A **difference relation** expresses the *dissimilarity* between two infoms with respect to the *context annotations* of the infoms (i.e., the set of all context parameters that describe the infom, Section 4.2). Difference relations can be transitive, symmetric, or ordered.

Table 4 illustrates difference relations between two infoms n , m and the context dimension d_n , d_m in which they differ or in which they need to have certain properties. For example, two translations differ in their language dimension, while a formalisation from n to m induces that n is formal (i.e., it has a formality context parameter) and m is informal (i.e., it has a language context parameter). The difference in the context dimension also needs to hold for transitive difference relations, e.g., if ‘ n translates m ’ and ‘ m translates w ’, then we can infer that ‘ n translates w ’ if both infoms differ in their language dimensions.

Relation name	d_n	d_m	Relation properties
translates	language	language	symmetric, transitive
formalises	formality	language	-
verbalises	language	formality	-
shorter_than	length	length	totally ordered
longer_than	length	length	totally ordered
smaller_than	size	size	totally ordered
larger_than	size	size	totally ordered
generalise	abstractness	abstractness	totally ordered
specialise	abstractness	abstractness	totally ordered
easier_than	difficulty	difficulty	totally ordered
more_difficult_than	difficulty	difficulty	totally ordered
different_language	language	language	symmetric, transitive
different_device	device	device	symmetric, transitive
different_layout	layout	layout	symmetric, transitive
different_area	area	area	symmetric, transitive
different_audience	audience	audience	symmetric, transitive

Table 4.: Examples for difference relations

An **equivalence relation** expresses the *interchangeability* between two infoms, which are different in some properties and relations but essentially similar in others. It is assumed that two infoms are equivalent (thus interchangeable) if their differences are not noticeable or ignorable in a specific adaptation context. We can distinguish several kinds of equivalence relations, e.g., property equivalence, goal equivalence, substance equivalence, and partial substance equivalence. They express different notions of interchangeability.

Two infoms are **property equivalent** if they are equal regarding specific properties, their content (substance) and intentions can vary. Examples for property equivalence are ‘same language’, ‘same layout’, ‘same area’, or ‘same audience’.

Two infoms are **goal equivalent** if they share the same *intentions*, their content and properties can vary. Goal equivalence can be inferred from ontological relations.

For example, the two proofs p and p' in Figure 39 are goal equivalent because they *prove* the same theorem t . The three examples e , e' , and e'' are goal equivalent because they *illustrate* the same or an *equivalent* definition: e' and e'' illustrate the same definition d' , while e

7. The Content Planning Approach

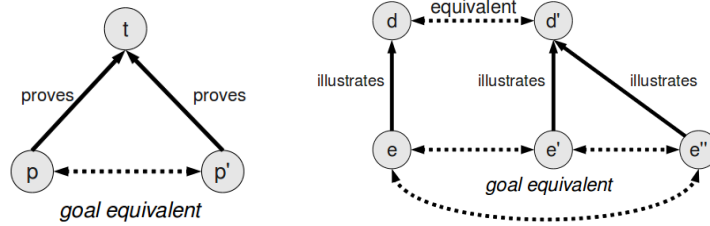


Figure 39.: Examples for goal equivalence

and e' as well as e and e'' are goal equivalent because they illustrate one of the two equivalent definitions d and d' .

Two infoms are **substance equivalent** if they have identical content or represent the same information value. Two infoms are **partially substance equivalent** if they have a ‘high’ proportion of identical (or equivalent) components.

For simplicity, we assume that the measure for the level of partial equivalence between two infoms is based on the number of shared children. Of course, using the number of shared children to express different levels of equivalence is far too simple and is only used for illustration. Instead, the coverage, size, or (even better) the *semantics* of the shared components should be considered.

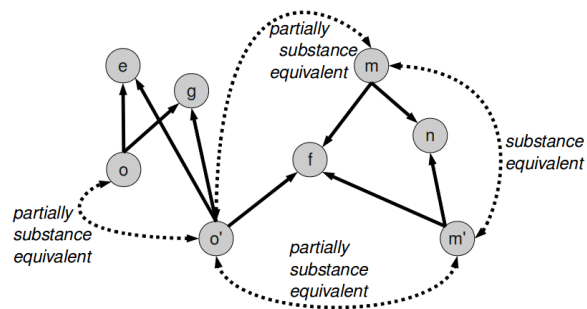


Figure 40.: Substance equivalence

Figure 40 illustrates the substance equivalence between the infoms m , m' , o , and o' . The infoms m and m' share two children (f and n) and are substance equivalent. The infoms o and o' share two children (g and e) and are partial substance equivalent with level 2. The infom o' shares one child with m and m' , o' and m as well as o' and m' are partial substance equivalent with level 1.

A **variant relation** is a difference relation extended by an equivalence relation (see Table 5). As equivalence relations are symmetric, an *inverse* relation can be identified for any non-symmetric variant relation. The definition of variants can now be refined: Two infoms are variants, if they relate in a variant relation.

Table 5 provides examples for variant relations. Some difference relations imply the equivalence of two infoms. For example, two infoms are translations if they differ in language and if they are substance equivalent. An infom n formalises an infom m if they are substance equivalent and n is formal and m is informal. Other difference relation become variant relation if an equivalence can be identified. For example, the difference relation `different_areas` between two infoms is a variant relation, if the two infoms are goal equivalent.

7.2.2. Representation of Variant Relations

The annotation of variant relations assures that only appropriated document parts are considered for the substitution and that the coherence of the document is preserved. In addition to the representation of variant relations, properties of these relations (relation type, symmetry, transitivity, or reflexivity) as well as inverse relations can be formalised. These support

Variants	Variant Relation	Equivalence	Inverse
translations	translates	substance equivalent	
formality variants	formalises	substance equivalent	verbalises
length variants	shorter_than	partially equivalent	longer_than
size variants	smaller_than	partially equivalent	larger_than
abstractness variants	generalises	partially equivalent	specialises
resolution variants	smaller_res_than	partially equivalent	higher_res_than
difficulty variants	more_difficult_than	goal equivalent	easier_than
device variants	different_device	goal equivalent	
layout variants	different_layout	goal equivalent	
area variants	different_area	goal equivalent	
audience variants	different_audience	goal equivalent	

Table 5.: Examples for variant relations

content planners to make inferences in order to enlarge the variant search space as well as to improve the accuracy of the selection. In order to be considered in a content planning workflow, variant relations and their properties have to be formalised and annotated in a machine-processable form.

Annotation of Variant Relations

Variant relations are a special kind of context parameter and can thus be annotated with the markup options of the notation framework (Section 4.4.1 and 4.4.2), in particular, the *T*, *MD*, *CCF*, and *IC*.

Listing 33: Tagging of variant relations in OMDOC

```
<tag type="varrel" xref="proof1" ic="more_difficult_than:proof2"
  owner="http://cmueller.myopenid.com"/>
```

Listing 34 illustrates the markup of variant relations using `tag` elements with type `varrel`. The tag references the object to be described (`proof1`) and specifies the variant relation in the `ic` attribute of the tag: `proof1` is more difficult than `proof2`. The `owner` attribute is used to associate the tag with an individual users. This can be helpful if the categorisation of variants is not distinct but rather depends on individual perspectives².

Listing 34: Metadata markup of variant relations in OMDOC

```
<proof xml:id="proof1" for="#lemma1" xmlns:cc="http://omdoc.org/ctxt.omdoc?var.context?">
  <metadata>
    <link rel="o:proves" href="#lemma1" />
    <link rel="cc:more_difficult_than" href="#proof2"/>
  </metadata>...
</proof>
<proof xml:id="proof2" for="#lemma1">
  <metadata>
    <link rel="o:proves" href="#lemma2" />
  </metadata> ...
</proof>
```

²Tags can be used to annotate any property/relation and, thus, support a collaborative markup approach (Section 8.4.3).

7. The Content Planning Approach

Listing 34 illustrates the inline markup of variant relations using the new metadata syntax for OMDOC. The namespace declaration introduces the prefix `cc`, which points to the content dictionary in Listing 38. The content dictionary defines the `more_difficult_relation` (and all other required relations and concepts). It allows to describe that the proof `proof1` is more difficult than proof `proof2`, though both prove the same lemma (`lemma1`).

Listing 35: Grouping of variant document parts

```
<variants ic="language:en">
  <proof xml:id="proof1" for="#lemma1">
    </proof>
  <proof xml:id="proof2" for="#lemma1">
    </proof>
  <ref xref="#proof3" />
</variants>
```

Listing 35 illustrates a pragmatic markup of variants. A `variants` element is used to group alternative document parts or references to document parts. The `ic` attribute defines the intensional constraints for the selection of one of these variant parts.

Listing 36: Grouping of variant paragraphs within statements

```
<proof>
  <variants ic="language:en">
    <CMP ic="language:en">Let us proof that ...</CMP>
    <CMP ic="language:de">Gegeben ist ...</CMP>
    <CMP ic="language:fr">Étant donné ...</CMP>
    <ref xref="#cmp.4" />
  </variants>
</proof>
```

The `variants` element can also be used within OMDOC statements to group paragraphs, sentences, and single words. In Listing 36 the `variants` element groups a number of alternative `CMP` elements of a proof. Listing 37 illustrates how the `variants` element can be used to group phrases.

Listing 37: Grouping of variant phrases within statements

```
<definition>
  The binomial coefficient  $C_n^k$  is the number of ways of choosing  $k$  objects from a collection of  $n$ 
  distinct objects without regard to the order. Alternative notations include the
  <variants>
    <phrase>French notation  $C_n^k$ </phrase>
    <phrase>English notation  $\binom{n}{k}$ .</phrase>
    <ref xref="#binom.ru" />
  </variants>
</definition>
```

Formalisation of Variant Relations

A machine-representable formalisation of variant relations allows users to guide the substitution processes declaratively. Widely used approaches are content dictionaries, ontologies, and rule systems.

Listing 38: A content dictionary for variant relations

```

<omdoc ...>
  <theory xml:id="var.context">
    <symbol name="more_difficult_than" />
    <definition for="more_difficult_than">
      <CMP>
        The more_difficult_than relation is a variant relation that expresses the goal equivalence
        between two infoms, which differ in the dimension "difficulty". The more_difficult_than
        relation is a total order. It can have infom references as value to denote the infom which is
        less difficult than the infom it describes.
      </CMP>
      <FMP>
        If  $X$  is totally ordered under  $>$ , then the following statements hold for all  $a, b$  and  $c$  in  $X$ :
        If  $a > b$  and  $b > a$  then  $a = b$  (antisymmetry);
        If  $a > b$  and  $b > c$  then  $a > c$  (transitivity);
         $a > b$  or  $b > a$  (totality). ...
      </FMP>
    </definition>
    <symbol name="goal-equivalence" /> ...
  </theory>
</omdoc>

```

Listing 38 illustrates the formalisation of variant relation in content dictionaries. The definition element in the content dictionary for the `more_difficult_than` relation embraces an `CMP` and an `FMP` element. While the former includes an informal description of the relation, the latter provides a formalisation of the relation's properties. For simplicity, a mixture of text and formulae is used in the `FMP` element to express that the relation is totally ordered. A representation in `OPENMATH` supports the automatic processing of these properties and the declaratively guiding of the content planning processes. Without such formalisations, machines can at least distinguish different relations by resolving the symbol pointer `cc:more_difficult_than` and by selecting among hard-coded routines that interpret these relations. Moreover, the informal specification in the `OPENMATH` content dictionaries suits as documentation between developers of respective adaptation services and can be incrementally formalised by them.

Rule representation languages include the rule markup language `RULEML` [Rul], `OWL Description Logic (DL)` [SWM04], the declarative language `CycL` language [Cyc], the `GrGen` rule syntax [grg], but also formalisation in a pseudo-code languages (similar to `Prolog` [Wie09]) such as illustrated in Listing 39.

Listing 39: Rule system for difficulty and equivalence relations

```

1 n > m if
2   nested(n), nested(m),
3   n ~ m,
4   children(n) forall _ > childOf(m) .

6 n > m if
7   nested(n), nested(m),
8   n ~ m,
9   c1 = childOf(m),
10  children(n) forall _ > c1 or not (_ < c1),
11  child(n) > child(m) .

12 n > m if

```

7. The Content Planning Approach

```
14  n ~ m,  
    effort(n) greater-than effort(m) .  
16  
17  n > m if  
18  n ~ m,  
    isHole(n) .  
20  
21  n ~ m if  
22  nested(n), nested(m),  
    cl = childOf(n),  
24  children(n) U children(m) forall cl ~ _ .  
  
26  n ~ m if  
    nested(n), nested(m),  
28  sameSize(children(n), children(m)),  
    children(n) forall _ ~ childOf(m),  
30  children(m) forall _ ~ childOf(n) .  
  
32  n ~ m if  
    sameType(n, m),  
34  sameTopic(n, m) .  
  
36  n ~ m if  
    isHole(n),  
38  sameTopic(n, m) .
```

The rule system in Listing 39 defines the `more_difficult_than` variant relation, denoted by $>$, and its equivalence relation, denoted by \sim . The intuition of the illustrated equivalence is that two atomic infoms are equivalent if they have the same type (e.g., `exercise`) and the same topic (e.g., `graphs`). An atomic infom n is more difficult than another atomic infom m , if they are equivalent and if the effort for n is greater than the effort for m . For example, for exercises the *effort* refers to the average time frame needed to solve an exercise.

The machine-processable annotation and formalisation of variant relations require specific parsers in order to be processed. Here we can draw on a plethora of tools, such as the KREX-TOR [Lan09b] for the parsing of RDFa annotations in OMDoc as well as *locutor* [loc] for the processing of rule systems, e.g., in GrGen syntax [grg]. In the further course, we focus on the internal data structures and the routines for processing the respective variant relations and their properties.

7.3. Specification for the Content Planning

Figure 41 shows the grammar for the content planning. It is based on the grammar of notation definitions as introduced in Section 4.2 and as defined by [KLM⁺09, KMR08].

An **infom** ι is an addressable document part of arbitrary size. Document parts that include other document parts are represented as **nested infoms**: $\iota = (\langle \iota_1, \dots, \iota_i \rangle, \{t_1, \dots, t_j\}, \lambda)$. Document parts that are solely plain text are represented as **atomic infoms**: $\iota = (\langle C_1, \dots, C_i \rangle, \{t_1, \dots, t_j\}, \lambda)$. An infom is represented as tuple: The first component is the content of an infom. It is represented as a sequence of infoms (for nested infoms) or a sequence of characters (for atomic infoms). The infoms ι_1, \dots, ι_i are also called the children of ι . The second component is a *set of transitions* of the infom. The third component is the *con-*

7.3. Specification for the Content Planning

Variation context	Π	::=	ι^*
Infom Graph	\mathcal{G}	::=	$(\mathcal{V}, \mathcal{E})$
Infomset	\mathcal{V}	::=	ι^*
Edgeset	\mathcal{E}	::=	e^*
Edge	e	::=	$e_l(\iota_i, \iota_j)$
Edge Label	l	::=	$(\#, \tau)$
Infom	ι	::=	$(\iota^+ C^+, t^*, \lambda)$
Hole	\square	::=	ι
Walk	\mathcal{W}	::=	$\vec{\iota}^*$
Abstraction context	Σ	::=	$\#^*$
Transition	t	::=	$(\#, \tau, r, \lambda, \vec{\iota}^*)$
Transition type	τ	::=	$\text{sem} \mid \text{nar}$
Transition rule	r	::=	$pre \rightarrow con$
Prerequisite	pre	::=	$pred$
Consequence	con	::=	$pred$
Predicate	$pred$::=	$\#^* \epsilon$
Transition infom	$\vec{\iota}$::=	ι
Adaptation context	Λ	::=	cp^*
Context annotation	λ	::=	cp^*
Context parameter	cp	::=	$(d = v)$
Context dimension	d	::=	s
Context value	v	::=	$s \#$
Unique identifier	$\#$::=	C^+
Symbol	s	::=	$\sigma(n, n)$
Names	n	::=	C^+
Weight	w	::=	real
Position	pos	::=	integer
Character	C	::=	any character

Figure 41.: Grammar for the content planning

text annotation of the infom. The context annotation includes a context parameter cp , which specifies whether the infom is displayed, i.e., $cp = (\text{display} = \text{true})$, or hidden in the document, i.e., $cp = (\text{display} = \text{false})$. Infoms are referenced with a *unique identifier*. A **unique identifier** $\# = \langle C_1, \dots, C_i \rangle$ is represented as sequence of characters, which are permitted according to the `xml:id` reference scheme [MVW05].

A **transition** $t = (\#, \tau, r, \lambda, \{\vec{\iota}_1, \dots, \vec{\iota}_i\})$ is a tuple, where the first component is the *unique identifier* of the transition, the second component is the *transition type*, the third component is a *transition rule*, the fourth component is the *context annotation* of the transition, and the fifth component is the set of *transition infoms*. A **transition type** τ can have the value `sem`, to denote a semantic transition, and `nar`, to denote a narrative transition. A **transition rule** $r = pre \rightarrow con$ is a rule, where the *prerequisite* of the rule (pre) specifies which infoms in a document have to be included before the infom and the *consequence* of the rule (con) defines the subsequent infoms in the document. The prerequisite and consequence of a rule are predicates. A **predicate** $pred$ is an *infom reference*, a *set* of infom references, a *sequence* of infom references, or *empty* (ϵ). Each transition is associated with an infom. Infoms can have multiple transitions, which produce alternative connections with other infoms. The context annotation can be used to select an appropriate transition between two infoms. A **transition infom** $\vec{\iota}$ is an infom, which represents a transitional text that is associated with a transition.

A **hole** $\square = (\langle \iota_1, \dots, \iota_i \rangle, \emptyset, \lambda)$ is an infom *without substance*. It is represented as tuple,

7. The Content Planning Approach

where the first component is a sequence of infoms, the second component is an empty set, and the third component is a context annotation. The elements ι_1, \dots, ι_i can be considered to replace (or *fill*) the hole and are called **variant** infoms. The context annotation of a hole can define the required properties and relations of the preferred variant.

A **document** doc is an ordered tree, where nodes correspond to the infoms in the document and edges denote their parent-child relations³. The document tree is traversed from left to right. Documents that contain variants are called **variant containers**. Documents with holes are called **abstract documents**, denoted by $d\Box c$.

\mathcal{L} is a lookup function. It takes as input a document doc and an identifier $\#$ of a transition and returns the set of transition infoms $\{\vec{\nu}_1, \dots, \vec{\nu}_i\}$ of this transition from doc .

The **abstraction context** $\Sigma = \{\#_1, \dots, \#_i\}$ is represented as set of infom references, which identifies all *replaceable* or *sortable* infoms. Replaceable infoms are substituted with variants, sortable infoms are reordered, that is, the order of their constituents is changed.

A **variant context** $\Pi = \langle \iota_1, \dots, \iota_i \rangle$ is a sequence of infoms. The substitution algorithm interprets the elements in Π as variants that can be used to replace a hole, where the first element in Π is the preferred one. For the reordering, the variant context includes the constituents of an infom, which is sortable, and specifies their order.

Context Dimension	Context Value
language	de, en, fr, ...
formality	FOL, HOL, ...
length	abstract < summary < fulltext < ...
size	1 < 2 < 3 < ...
abstractness	1 < 2 < 3 < ...
resolution	1 < 2 < 3 < ...
difficulty	easy < medium < hard
target device	PDA, desktop, laptop, ...
layout	display, text, figure, ...
area of application	mathematics, physics, ...
more_difficult_than	<i>graph.omdoc#spanningtree</i>
formalises	<i>graph.omdoc#informal-definition</i>
translates	<i>graph.omdoc#informal-definition</i>

Table 6.: Context dimensions and their values

A **context parameters** is a key-value pair ($d = v$), where d denotes a context dimension and v its context value. In order to specify relations, the previously defined context values v are extended (Section 4.2).

Context values are *symbols* or *infom references*. With this extension, a context parameter can define properties, e.g., (`language = en`) and relations, e.g., (`more_difficult_than = exam2000.omdoc`) (see Table 6).

The **adaptation context** Λ represents the ordered set of context parameter (or constraints) that define the adapted, user-specific document. The order of a context parameter cp_s in Λ denotes its priority P for the adaptation, which is computed by subtracting the position pos of cp_s from the total number of context parameters in Λ , i.e., $P(cp_s) = size(\Lambda) - pos(cp_s)$.

A **context annotations** λ of an infom is represented as ordered set of context parameters that describe the infom (see definition in Section 4.2).

³Recall that we consider an XML document to be an ordered tree, following the XPATH data model specified in [CD99]. We use the term document and document tree interchangeably.

7.3. Specification for the Content Planning

An **infom graph** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a simple, directed, labelled multigraph, where \mathcal{V} is a set and \mathcal{E} is a set of ordered pairs of elements from \mathcal{V} . The elements of \mathcal{V} are called nodes and correspond to infoms. The elements of \mathcal{E} are called edges and represent semantic and narrative transitions between the infoms. An infom graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ is called the **subgraph** of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$, short $\mathcal{G}' \sqsubseteq \mathcal{G}$.

An **empty node** represents a reference to an infom that is not contained in the infom graph.

An **edge label** $l = (\#, \tau)$ provides a unique identifier of a transition $\#$ and the transition type τ of the transition, i.e., `sem` or `nar`. Labels with $\tau = \text{sem}$ are called **semantic labels**, labels with $\tau = \text{nar}$ are called **narrative labels**.

An edge $e_l(u, v)$ is **incoming** for v and **outgoing** for u . For a node $v \in \mathcal{V}$, we denote the set of incoming edges by $\mathcal{E}_i(v)$, the set of outgoing edges by $\mathcal{E}_o(v)$. The number of elements in $\mathcal{E}_i(v)$ is called the **in-degree** of v , denoted by $deg_i(v)$, the number of elements in $\mathcal{E}_o(v)$ is called the **out-degree** of v , denoted by $deg_o(v)$. A node v with $deg_i(v) = 0$ is **self-contained**.

An edge e is called **semantic edge** if it has a semantic label. A semantic edge $e_l(v, u)$ connects v and u semantically, i.e., it represents a semantic transition from v to u . The semantic edges in \mathcal{E} and the semantically connected nodes in \mathcal{V} form a acyclic directed graph. For a node $v \in \mathcal{V}$, we denote the set of incoming semantic edges by $\mathcal{E}_{si}(v)$, the set of outgoing semantic edges by $\mathcal{E}_{so}(v)$. The number of elements in $\mathcal{E}_{si}(v)$ is called the **semantic in-degree** of v , denoted by $deg_{si}(v)$. A node v with $deg_{si}(v) = 0$ is **semantically self-contained**. A node v with $deg_{si}(v) > 0$ is a dependant and requires other infoms. The set of its semantic supporters is denoted by $\mathcal{V}_{ss}(v)$. The size of $\mathcal{V}_{ss}(v)$ is equal to $deg_{si}(v)$. The number of elements in $\mathcal{E}_{so}(v)$ is called the **semantic out-degree** of v , denoted by $deg_{so}(v)$. A node v with $deg_{so}(v) > 0$ is a supporter and is required by other infoms. The set of its semantic dependants is denoted by $\mathcal{V}_{sd}(v)$. The size of $\mathcal{V}_{sd}(v)$ is equal to $deg_{so}(v)$.

A **semantic walk** \mathcal{W}^s in \mathcal{G} is a sequence $\langle v_1, \dots, v_k \rangle$ of nodes of \mathcal{G} , such that \mathcal{G} contains edges $e = e_l(v_i, v_{i+1})$ for all $i = 1, \dots, k$ with a semantic label. The length of the walk $\mathcal{W}^s = \langle v_1, \dots, v_k \rangle$ is k . A semantic walk is called a **semantic path**, if all nodes v_1, \dots, v_k are distinct. A semantic walk that includes empty nodes is not **complete**.

A node, which is *on a semantic walk*, has been **visited**. For $v \in \mathcal{V}$, we denote the set of all visited semantic supporters of v by $\mathcal{V}_{vss}(v)$, the set of its visited semantic dependant by $\mathcal{V}_{vsd}(v)$. The number of elements in $\mathcal{V}_{vss}(v)$ is denoted by $deg_{vss}(v)$, the number of elements in $\mathcal{V}_{vsd}(v)$ is denoted by $deg_{vsd}(v)$. The difference $deg_{si}(v) - deg_{vss}(v)$ is called the **semantic traversal in-degree**, denoted by $deg_{sti}(v)$, the difference $deg_{so}(v) - deg_{vsd}(v)$ is called the **semantic traversal out-degree**, denoted by $deg_{tso}(v)$.

An edge e is called **narrative edge** if it has a narrative label.

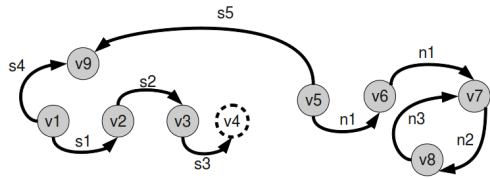
A **narrative walk** \mathcal{W}^l in \mathcal{G} is a sequence $\langle v_1, \dots, v_k \rangle$ of nodes of \mathcal{G} , such that \mathcal{G} contains edges $e_l(v_i, v_{i+1})$ for all $i = 1, \dots, k$ with narrative label $l = (\#, \tau)$ with equal transition reference $\#$. A narrative walk is called **narrative path**, if all nodes v_1, \dots, v_k are distinct. A narrative walk that includes empty nodes is not **complete**. A narrative walk $\mathcal{W}^l = \langle v_1, \dots, v_k \rangle$ connects nodes v_1, \dots, v_k narratively. All nodes support the narrative walk. If one of the nodes is removed or changed, all other nodes on the walk are **affected**.

\mathcal{N} denotes the set of all narrative walks \mathcal{W}_i^l in \mathcal{G} . $\mathcal{N}(v)$ denotes all narrative walks \mathcal{W}_i^l in \mathcal{G} , which include node v . The nodes in $\mathcal{N}(v)$ (apart from v) are the **narrative dependants** and **narrative supporters** of v . They form the set of narrative supporters denoted by $\mathcal{V}_{ns}(v)$ as well as the set of narrative dependants $\mathcal{V}_{nd}(v)$. The number of elements in $\mathcal{V}_{ns}(v)$ is equal to the number of elements in $\mathcal{V}_{nd}(v)$. It is called the **narrative degree** and is denoted by $deg_n(v)$.

A **traversal walk** P in \mathcal{G} is a sequence $\langle v_1, \dots, v_k \rangle$ through \mathcal{G} , where all non-empty nodes of the graph are (at least once) on the walk. A traversal walk is called **traversal path**, if

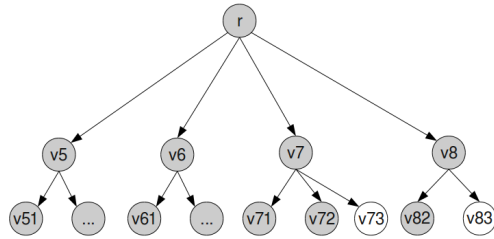
7. The Content Planning Approach

all v_1, \dots, v_k are distinct. The traversal can construct the sequence, while processing only the edges with semantic labels (called **semantic traversal**), only the edges with narrative labels (called **narrative traversal**) or edges with any label (called **hybrid traversal**). The semantic/narrative label of an edge act as condition for the traversal.



The semantic path $\langle v_1, v_2, v_3, v_4 \rangle$ is not complete: it includes an empty node (v_4). v_4 represents the reference that connects v_3 via the semantic edge s_3 .

The figure to the right illustrates a document tree, which includes the nodes v_5, v_6, v_7, v_8 as well as their children. Nodes v_{71}, v_{72} , and v_{73} as well as nodes v_{82} and v_{83} represent alternative transition infoms. Only those transition infoms are displayed, which guide the narrative transitions n_1 and n_2 in the graph above ($v_{51}, v_{61}, v_{71}, v_{72}$, and v_{82}). v_{73} and v_{83} are hidden as they support transition n_3 .

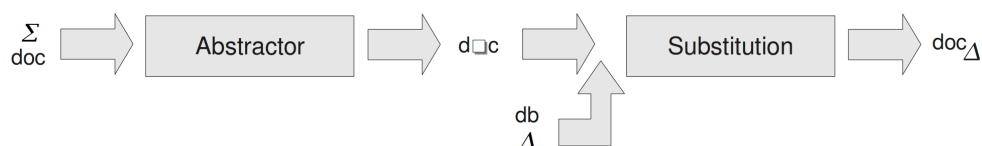


8. Substitution of Document Parts

In Part II we have solely looked at the phrase-level of mathematical text. However, adaptation of notations can not be seen independently from the adaptation of the whole document (Section 1.4). When entering constraints for the adaptation of notations, alternative document parts (exercises, definitions, examples, etc) can be retrieved that fit better. This section extends the notation framework for the substitution of document parts according to the semantic, narrative, and user context.

8.1. Information Model

The substitution of document parts is implemented as a two-stage process — the *abstraction* and the *substitution* of document parts. The term ‘**abstract document**’ is used to summarise the terminology of the previously discussed state of the art (Section 6.2). Accordingly, templates, literate programming documents, and documents with conditional texts, variables, options, or dynamic items are interpreted as abstract documents. They are configurable, e.g., by combining or referencing alternative content, and require a processing system in order to generate a concrete, user-specific instance.



In the first stage, the input document (*doc*) is **abstracted**: certain parts of the document are made *adaptable*. All other parts are *static* and can not be changed during the adaptation. The abstraction can be guided by an abstraction context (Σ). The intuition behind the abstraction context is to allow users to guide the abstraction by pointing to all infoms that should be made configurable and by omitting all others that should remain unchanged.

The document produced by the first step is an *abstract document* ($d\Box c$). If an infom in a document is made adaptable, it is actually replaced with a specific environment, a **hole**. The intuition of holes is the following: The content planning approach is specified as extension of the notation framework in which a content-oriented object, the OPENMATH representation of a mathematical expression, is substituted with a presentation-oriented object, the notation in Presentation-MATHML. The content-oriented object specifies the meaning of the mathematical symbol or formulae but can not be displayed to the user. Analogously, a term was needed to denote a representation of document parts, which can not be displayed to the user and which has to be substituted with a displayable, user-specific document part. The term ‘hole’ was used in analogy to the term ‘*black hole*’ in astronomy, a region of space from which nothing can escape and which absorbs any other particle. Analogously, a hole can contain any kind of document part. All document parts inside the hole are called **variants**. Users can skip the abstraction and provide a manually written abstract document (a document that contains *holes*) and use the adaptor to *complete* the document.

In the second stage, the adaptable infoms in the abstract document ($d\Box c$) are *substituted* with more appropriate infoms (retrieved from the database *db* and *doc*) according to the adap-

8. Substitution of Document Parts

tation context Λ . The adapted document doc'' is returned. This is done in three steps: alternative infoms (the *variants*) are collected, a user-specific variant is selected, and the hole is substituted with the user-specific variant.

These steps are similar to the processes of the notation framework. Its functionality is thus reused in a generalised form. The extension *collects variants* (instead of notation definitions) and *selects an appropriate variant* (instead of a rendering element). The respective hole (instead of the OPENMATH expression) is substituted with the selected variant (instead of the generated Presentation-MATHML expression).

If all holes have been replaced with user-specific infoms, a *consistency check* is applied onto the adapted document. The consistency check identifies whether all narrative walks between the infoms in the document are complete. This is particularly critical as these walks are usually visualised by transitional phrases and cross-references. If an infom on a narrative walk is missing, these visual markers can reduce the coherence of the document. The affected transition infoms are thus hidden from the document.

The next sections observe whether the extensional and intensional options of the notation framework can be reused. For the further discussion, it is assumed that we can draw on a huge corpus of mathematical documents in a content-oriented format, such as OMDOC, in which all document parts (exercises, examples, definitions, proof, etc.) can be uniquely addressed and their properties, transitions, and variant relations are marked. Furthermore, we require that the variant relations (and their properties, such as transitivity or partial order) are formerly defined and can be processed by the adaptation engine.

8.1.1. Extensional Collection of Variants

Variants are usually included in documents, such as textbooks, lecture notes, or manuals. For example, consider an instructor, who maintains a collection of exercises to reuse his exercises in exams and quizzes. These are grouped by topics and stored in a file system. Below a possible file structure is displayed.

```
gens/problems/function.omdoc
gens/problems/graph.omdoc
gens/problems/tree.omdoc
```

The directory `gens` includes a folder `problem` with three files `function.omdoc` (a collection of problems on functions), `graph.omdoc` (a collection of graph exercises), and `tree.omdoc` (a collection of tree exercises). When creating a new exam or quiz, the tutor should be able to reuse his exercise collection.

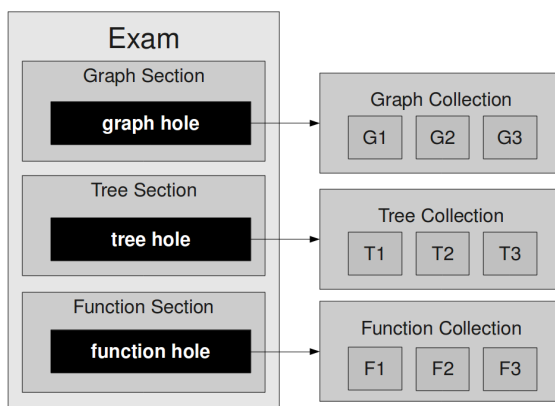


Figure 42.: Associating holes with exercises

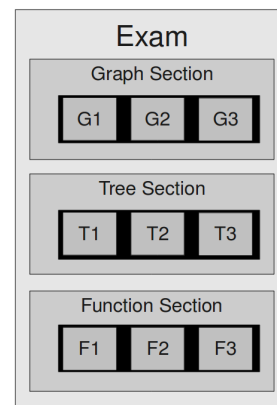


Figure 43.: Holes with exercises

To associate a collection of alternative document parts with the input document, the *extensional options* of the notation framework can be used (see Section 8.3.1). However, these options have to be handled with care! In the notation framework, only appropriate elements, the notation definitions, are collected. The pattern matching prevents that inappropriate notation definitions are applied. For the substitution arbitrary document parts (exercises, examples, proofs, etc.) can be collected. Pattern matching does not apply. Instead, authors have to assure that only applicable document parts are collected.

For example, Figure 42 illustrates how the instructor can use the extensional option *EC* or *T* to associated holes with appropriate exercise collections: the graph collection is associated with the graph hole, the tree collection with the tree hole, and the function collection with the function hole. This assures that only relevant exercises are placed in the respective hole.

Alternatively, the instructor can write an abstract documents and include arbitrary many variants into the holes of the document. The abstract exam in Figure 43 has three holes, which each include a number of exercises. These have been copied from the exercise collection: the graph collection was copied to the graph hole, the tree collection to the tree hole, and the function collection to the function hole. During the substitution each holes is substituted (or ‘filled’) with one of the variants it embeds.

Both approaches sound rather tedious. They prevent the tutor from simply pointing to a document that contains variants and to leave all tedious associations to the system. An intensional specification of context parameters can support such services, in particular, if variant relations are included. Variant relations assure that only relevant document parts are considered. For example, the tutor can add a context parameter to the graph hole, which defines that the selected variant should be more difficult than another exercise (see Listing 46).

8.1.2. Intensional Selection of Variants

Context parameters allow to take specific *properties* (e.g., the type or topic of the preferred document part) and *relations* (e.g., variant relations) into account. They can filter the number of collected documents parts and increase the accuracy of the output document. The substitution reuses the options of the notation framework to collect the context specifications.

Context parameters should include variant relations to assure that only appropriate document parts are considered. In addition, dependencies between document parts should be taken into account, which denote their prerequisites. They allow the system to filter variants according to whether all their prerequisite are included in the previously traversed part of a document or not and increase the coherence of the document.

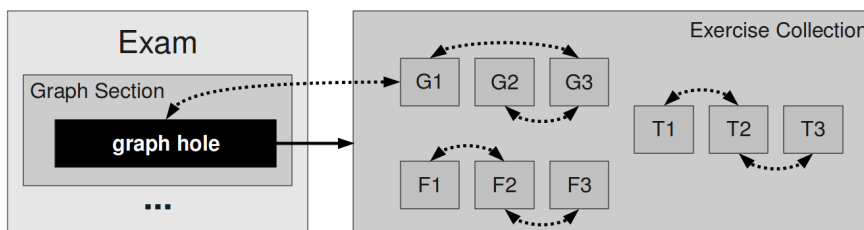
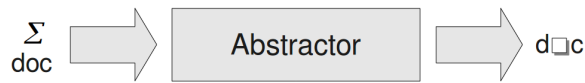


Figure 44.: Intensional selection of variants

The document in Figure 44 includes a hole, which is associated with a collection of exercises. These include graph, tree, and function problems. However, the hole is part of the graph section and should only be replaced with graph problems. A context parameter of the hole specifies that it should be `more_difficult_than` the graph exercises G1. This relations assures that only appropriate exercises, i.e., G2 or G3, are considered for the substitution.

8.2. The Abtractor

The abtractor takes as input a document doc and an abstraction context Σ . The abstraction context is represented as set of infom references $\#_i$, which identifies all *replaceable* infoms in the document that can be substituted with more appropriate variants. The abtractor returns the abstracted document $d\Box c$, in which all replaceable infoms have been substituted with holes.



8.2.1. Representation of Holes

For the representation of holes, a `variants` element (in the OMDOC namespace) is introduced. The `variants` elements can be used to manually configure an abstract document and to skip the abstraction. Each `variants` element can be associate with an adaptation context, using the options in Section 8.3. Optionally, `variants` elements can contain potential candidates for the substitution. For example, the `variants` element in Listing 40 includes two alternative examples.

Listing 40: XML representation of holes

```

<variants>
  <metadata><link rel="for" href="#def1" /></metadata>
  <example ic="language:en" for="#def1">...</example>
  <example ic="language:de" for="#def1">...</example>
</variants>
  
```

8.2.2. Guiding the Abstraction

The abstraction context is a form of *stand-off markup*. It allows users to guide the abstraction by pointing to the replaceable paragraphs in a document without modifying it. Alternatively to the abstraction context, users can use *inline markup* to guide the abstraction. For this, all elements in the input document doc can be annotated with a context parameter (`replaceable = true|false`) to mark whether they are replaceable or not. The current approach uses the `ic` attribute and `metadata` element in the OMDOC namespace.

Listing 41: XML representation of the abstraction property

```

<definition ic="replaceable:true">
  <CMP>
    An algorithm is ...
  </CMP>
</definition>
<example>
  <metadata>
    <dc:title>Kruskal algorithm, a graph algorithm for spanning trees</dc:title>
    <meta property="cc:replaceable">true</meta>
  </metadata>
  ...
</example>
  
```

In Listing 41 both markup alternatives are illustrated. They initialise the replacement of the `example` and `definition` elements with `variants` elements.

8.2.3. The Abstraction Algorithm

For the abstraction, the tree-representation of the input document doc is processed and references to all elements marked by the previously introduced inline markup are added to the abstraction context. Afterwards, the input document tree is traversed from left to right and every infom ι , which is referenced by the abstraction context Σ_ι , is abstracted. The abstract document $d \sqsupset c$ is returned.

The abstraction supports a strict and a relaxed mode. The **strict mode** prevents that semantic or narrative supporter are removed from a document. Removing these supporters is problematic since they are required by other infoms in the document. Infoms that do not act as supporter, can be removed since they do not foster the understanding of other infom. The **relaxed mode** permits the removal of supporters, though, inconsistencies can arise.

Listing 42: Strict abstraction of infoms

```

1 if  $deg_{so}(\iota) == 0$  then
2   if  $deg_n(\iota) == 0$  then
3      $\square = \text{new hole}$ 
4     add  $\lambda$  of  $\iota$  to  $\square$ 
5     substitute  $\iota$  with  $\square$ 
6     add  $\iota$  to  $\square$ 
7   fi
8 fi

```

Listing 42 specifies the *strict* abstraction of an infom. In a preprocessing step, the input document is parsed into an infom graph $\mathcal{G}(doc)$. The abtractor first verifies, whether the infom ι is a supporter for any other infom in the document doc : If the semantic out-degree $deg_{so}(\iota)$ and the narrative degree $deg_n(\iota)$ is zero, there are no semantic or narrative dependants for ι in the document. If the infom is not a supporter for any infom in doc , it can be removed from the document as it is not required to understand other dependent infoms. The abstraction proceeds: A hole is created. The context annotation of the infom ι (i.e., all properties and relations) are added to the hole. The ι is substituted with the hole. It is *preserved* in the hole, i.e., added as child of the hole. The relaxed abstraction mode solely computes the steps from Line 3 to 6 and, thus, permits that supporters are removed.

<pre> <omdoc xml:id="exam_template.omdoc" ...> <theory xml:id="exam_template.t"> <tgroup layout="section"> <exercise xml:id="friends.ex"> </exercise> </tgroup> <tgroup layout="section"> <exercise xml:id="spanningtree.ex"> </exercise> </tgroup> <tgroup xml:id="sec_fun" layout="section"> <exercise xml:id="whatis.ex"> </exercise> </tgroup> </theory> </omdoc> </pre>	<pre> <omdoc xml:id="exam_template.omdoc" ...> <theory xml:id="exam_template.t"> <tgroup layout="section"> <variants> <exercise xml:id="friends.ex"> </exercise></variants> </tgroup> <tgroup layout="section"> <variants> <exercise xml:id="spanningtree.ex"> </exercise></variants> </tgroup> <tgroup layout="section"> <variants> <exercise xml:id="whatis.ex"> </exercise></variants> </tgroup></theory></omdoc> </pre>
--	---

Figure 45.: Abstraction of exam2000.omdoc

8. Substitution of Document Parts

Example Figure 45 illustrates the Exam 2000 from our exam generation use case (Section 6.3.1) before the abstraction (left hand side) and after the abstraction (right hand side). The original document contains three sections, which each embed one `exercise` element. The abstraction context $\Sigma = \{ // exercises \}$ is given and defines that all exercises are replaceable. In the abstract document, all exercises are embedded in a `variants` element, which denotes that they can be replaced with a more appropriate one.

8.3. The Substitution Algorithm

The substitution algorithm is a generalisation of the rendering algorithms in Section 4. The notation collector is replaced by the general *infom collector* and the rendering grabber is substituted with the *variant grabber*. Figure 46 illustrates all components of the *substitution*.

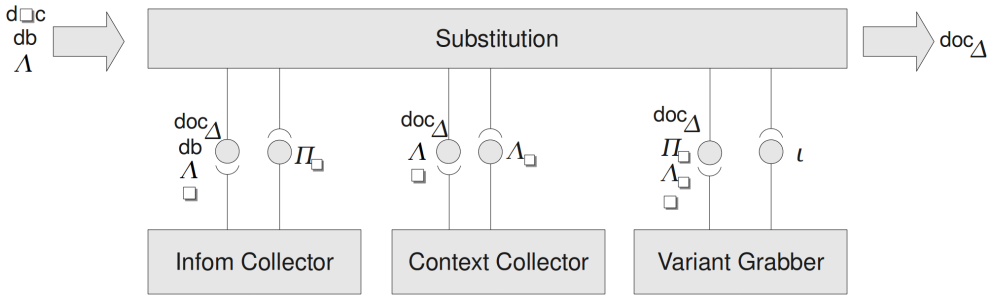


Figure 46.: Components of the substitution algorithm

The **substitution** takes as input an abstract document $d\square c$, a document database db , and an adaptation context Λ . The intuition of db is that it provides all cross-referenced and imported documents for the adaptation. The substitution returns the user-specific document doc_Δ , which adapts to the semantic context, the narrative context, and the user context as specified by the adaptation context Λ .

Listing 43: Producing doc_Δ

```

1   $doc_\Delta = d\square c$ 
2
3  forall holes  $\square_i$  in  $d\square c$  do
4     $doc_\Delta = \text{substitute hole with infom for } (\square_i, doc_\Delta, db, \Lambda)$ 
5  done
6
7   $doc_\Delta = \text{remove all obsolete transition infoms in } doc_\Delta$ 

```

Listing 43 specifies how the user-specific document doc_Δ is constructed from the abstract document $d\square c$. The document doc_Δ denotes the *current adaptation result*. It is initialised with the abstract document $d\square c$ and incrementally modified (line 3-4): The document tree doc_Δ is traversed from left to right and every hole \square is substituted with an appropriate user-specific infom. For the substitution the subroutine `substitute hole with infom` in Listing 44 is called iteratively. In a final step, a *consistency check* is applied to doc_Δ , which hides all obsolete transition infoms (Listing 45).

Listing 44: Substitute hole with infom for $(\square, doc_\Delta, db, \Lambda)$

```

1  $\Pi_\square$  = construct the variant context for  $(\square, \Lambda, db, doc_\Delta)$ .
2
3  $\Lambda_\square$  = compute the effective adaptation context for  $(\square, \Lambda, doc_\Delta)$ 
4
5  $\iota$  = select infom for  $(\square, doc_\Delta, \Lambda_\square, \Pi_\square)$ 
6
7  $doc_\Delta$  = substitute  $\square$  in  $doc_\Delta$  with  $\iota$ 

```

Listing 44 specifies the steps for selecting a user-specific infom. The *infom collector* constructs the variant context Π_\square for the hole, which contains all infoms that can potentially replace the hole in doc_Δ according to Λ . The *context collector* computes the effective adaptation context Λ_\square . Given Π_\square and Λ_\square as well as the hole and the current adaptation result doc_Δ , the *variant grabber* sorts and filters the variant context and returns the most appropriate infom ι (Listing 47). Finally, the hole is substituted with the selected infom and the adaptation result doc_Δ is updated respectively.

The **consistency check** assures that the adaptation results remains coherent, i.e., it verifies whether all narrative supporters are part of the document doc_Δ . If a narrative supporter is missing, the consistency check changes doc_Δ . The removal of a narrative supporter from the original document doc is extremely critical since their connections with other infoms (its dependants) is visualised by transitional phrases and cross-reference. If the narrative supporter is removed, all dependent infoms are affected and the coherence of the document is no longer guaranteed. The consistency check restores the coherence of the document by hiding all affected *transition infoms* that support the narrative transitions between the narrative supporter and its dependants.

Listing 45: Consistency check for doc_Δ

```

1  $\mathcal{G}(doc_\Delta)$  = parse  $doc_\Delta$ 
2
3 forall narrative walks  $\mathcal{W}_i^{(\#,nar)}$  in  $\mathcal{G}(doc_\Delta)$  do
4   if  $\mathcal{W}_i^{(\#,nar)}$  not complete
5      $doc_\Delta$  = hide all transition infoms in  $\mathcal{L}(doc_\Delta, \#)$ 
6   fi
7 done

```

Listing 44 specifies the consistency check, which is applied onto doc_Δ . In a first step, the document doc_Δ is parsed into an infom graph $\mathcal{G}(doc_\Delta)$. All narrative walks $\mathcal{W}_i^{(\#,nar)}$ in $\mathcal{G}(doc_\Delta)$ are retrieved and verified. If a narrative walk is not *complete*, it includes empty nodes. These missing nodes make the respective narrative transitions between the nodes on the walk obsolete. Consequently, all transition infoms that are associated with the transition with unique identifier $\#$ have to be hidden (their display status is set to *false*). These transition infoms are retrieved by calling the lookup function $\mathcal{L}(doc_\Delta, \#)$.

Example The next sections show how the exam use case (Section 6.3.1) is supported. In particular, it is illustrated how the *graph hole* (the first `variants` element with `xml:id="graph_hole"`) in the abstract exam in Listing 46 is substituted with a concrete exercise. For this purpose, the previously constructed abstract exam from Figure 45 is enriched with exercises from other exams and problem collections. One of the exercises in the graph hole is a tree problem (`spanningtree.ex`) and should not be selected.

8. Substitution of Document Parts

The semantic and narrative transitions are marked for our example. They are parsed into the internal data structure for context parameters¹. Thanks to the transition markup all transitional phrases and cross-references are separated from the reusable parts of the document parts². For example, the content of the `centergraph.ex` is divided into a *reusable part* asking the students to define the shortest distance in a graph and a *variable part*, which requires the figure in the exercise `shortestpath.ex` in order to be completed. The variable part depends on the exercise `shortestpath.ex`. It should not be inserted if the supporter (`shortestpath.ex`) is not placed before the exercise.

Listing 46: Abstract exam for the exam use case

```
<omdoc xml:id="exam_template.omdoc" ... xmlns:cc="http://omdoc.org/ctxt.omdoc?var.context?">
  <theory xml:id="exam_template.t">
    <tgroup layout="section">
      <metadata><dc:title>Graphs</dc:title></metadata>
      <variants xml:id="graph.hole">
        <exercise xml:id="friends.ex">
          <metadata>
            <meta property="cc:language">cc:en</meta>
            <link rel="cc:not_contained_in" href="#exam1999.omdoc"/>
          </metadata>
          ...
        </exercise>
        <exercise xml:id="centergraph.ex" >
          <metadata>
            <link rel="cc:more_difficult_than" href="#friends.ex"/>
            <link rel="cc:not_contained_in" href="#exam1999.omdoc"/>
          </metadata>
          <CMP>
            <p>Define shortest distance in a graph.</p>
            <p type="transition" for="#n1">For each node in the graph above, calculate the sum
              of the shortest distances to all the other nodes in the graph.</p>
          </CMP>
          <transition type="narrative" xml:id="n1">
            <prerequisites>
              <prerequisite xref="#shortestpath.ex" />
            </prerequisites>
            <consequences></consequences>
          </transition>
        </exercise>
        <exercise xml:id="shortestpath.ex">
          <metadata>
            <meta property="cc:language">cc:en</meta>
            <meta property="cc:area">cc:cs</meta>
            <link rel="cc:more_difficult_than" href="#friends.ex"/>
            <link rel="cc:not_contained_in" href="#exam1999.omdoc"/>
          </metadata>
          <CMP><p>Suppose we draw a graph in which each node in the graph is a person and each
            edge in the graph is a connection between two people who are friends.</p>
```

¹Note that we focus on the pragmatic specification of context parameter and omit the reference to a symbol's content dictionary.

²Note that OMDOC does not yet support the `type` and `for` attribute for non-semantic markup elements, such as `p`. An alternative is the representation of type and relation with the `ic` attribute. Possibly, future work can specify a more elegant, fine-grained markup solution for the association of transitions and transition infoms.

```

<omlet data="#fimg" action="display" show="embed" />
<private xml:id="fimg">
  <data format="image/jpeg" href="exams/img/friends.jpg" />
</private>
<p>What is the shortest path distance between two nodes in a graph?</p></CMP>
</exercise>
<exercise xml:id="spalgo.ex">
  <metadata>
    <link rel="cc:more_difficult_than" href="#shortestpath.ex"/>
    <link rel="cc:more_difficult_than" href="#centergraph.ex"/>
    <link rel="cc:not_contained_in" href="#exam1999.omdoc"/>
  </metadata>
</exercise>
<exercise xml:id="spanningtree.ex">
  <metadata>
    <link rel="cc:not_contained_in" href="#exam1999.omdoc"/>
  </metadata>
  ...
</exercise>
</variants>
</tgroup>
<tgroup layout="section"><variants>...</variants></tgroup>
<tgroup layout="section"><variants>..</variants></tgroup>
</theory>
</omdoc>

```

The exam is traversed to replace each hole with an appropriate exercise. To find a variant for the graph hole, the substitution first passes the abstract document to the infom collector.

8.3.1. Infom Collector

The **infom collector** takes as input a hole \square , the adaptation context Λ , the current document doc_{Δ} , and the database db . It returns the variant context Π , which specifies all infoms that are potential variants for the hole. The order of the infoms in Π reflects the order in which they were collected from doc_{Δ} and db .

Users can define the set of available sources for the substitution of the holes in the input document. These sources form the **extensional selection** as users have to explicitly point to a *variant infom* or document with variants (i.e., a *variant container*). The adaptation context Λ specifies a totally ordered set S_V of source names, which are represented as contextual key-value pairs.

For the content planning, the extensional options introduced in Section 4.4.1, i.e., F , Doc , CD , EC , T , and SD . These are extended with the Var , which initiates the collection of variants from the holes in an abstract document. The user can change the priorities of these options by ordering them. The respective input sources are treated as follows.

- F denotes an **external document** from which variants are collected.
- Doc denotes the **input document**. It collects all infoms within the documents.
- Var denotes the **variants elements**. It collects all infoms within holes of the input document. Technically, the infoms within the `variants` elements of the XML representation of the input document are gathered.
- CD denotes the **content dictionaries** defining symbols occurring in the input document.

8. Substitution of Document Parts

- *EC* denotes the **extensional context**, which associates a list of variants or containers of variants to any infom of the input document.
- *T* denotes **variant tags**, which provide the same functionality as the *EC* option without changing the input document. They allow readers full control of the substitution as they explicitly point to the appropriate document parts. *T* has to be used in combination with either *F* (provided in an external file) or *Doc* (embedded in the holes of the input document).
- *SD* denotes the **system defaults**. For example, when integrating a content planner with a repository or proof assistance system (Section 8.4.4), *SD* can represent the repository's content.

Example For the exam example introduced in Section 6.3.1, the set of source names $\mathcal{S}_V = \{Var\}$ is used, which collects all infoms from the `variants` elements in the exam. For the graph hole, the infom collector gathers the four graph exercises `friends.ex`, `centergraph.ex`, `shortestpath.ex`, and `spalgo.ex` as well as one tree exercise `spanningtree.ex`. The collected infoms are returned as variant context.

8.3.2. Context Collector

The **context collector** takes as input a hole \square , the adaptation context Λ , and the current document doc_Δ . It returns the effective adaptation context Λ_\square by extending the functionality of the context collector in the rendering algorithm (see Section 4.4.2).

The collection of context parameters can be guided by the intensional options introduced in Section 4.4.1. Note that context parameters have been extended, supporting users to specify any property or relation. As a best practice, *variant relations* should be used to guide the intensional selection. Users can define the set of available sources for the intensional adaptation. The adaptation context Λ specifies a totally ordered set \mathcal{S}_C of source names. Based on the intensional options from Section 4.4.2, the source names *GC*, *CCF*, *IC*, and *MD* are used. The user can change their priorities by ordering them. The respective input sources are treated as follows.

- *GC* denotes the **global context** which provides a global set of context parameters.
- *CCF* denotes the **Cascading Context Files**, which permit a fine-grained contextualisation analogously to Cascading Style Sheets [BLLJ08].
- *IC* denotes the **intensional context**, which associates a list of contextual key-value pairs ($d = v$) to any infom in the input document.
- *MD* denotes structured **metadata** in the input document.

Example The context collector can only provide semi-automated support for the users. The more formalisation of variant relations (and other relations) are provided, the better the automation and the less markup effort for the user. In the following, the exam use case (Section 6.3.1) is illustrated with and without additional formalisations. Both approaches result in the effect adaptation context with the two constraints (`not_contained_in = exam1999.omdoc`) and (`more_difficult_than = #friends.ex`)

Without Formalisation For the exam use case, the set of source names $\mathcal{S}_C = \{GC\}$ is used. Two global constraints have been set:

1. Omit all problems in `exam1999.omdoc` and,
2. The exam should be harder than `exam2000.omdoc`.

The context collector can not resolve these abstract constraints automatically. Consequently, a more specific global context has to be specified: the global context parameter (`more_difficult_than = #friends.ex`) is provided, which indicates that the selected graph exercise should be harder than the exercise `friends.ex`, which was the graph exercise chosen for the `exam2000.omdoc`. To omit all exercises in `exam1999.omdoc`, the global context parameter (`not_contained_in = exam1999.omdoc`) is given. The effective adaptation context for the graph hole includes the two parameters.

To support the requested substitution, the exercises in the abstract exam have to annotated respectively. All exercises not contained in `exam1999.omdoc` are thus marked with a context parameter (`not_contained_in = exam1999.omdoc`) and all exercises harder than `friends.ex` are enriched with the relation (`more_difficult_than = #friends.ex`). These additional efforts can be reduced when extending the context collector with a formalisation of relations.

With Formalisation A rule system, such as illustrated in Listing 39, supports a system to make two types of inferences³. The rule in Line 1-4 supports a **bottom-up inference**: given the relations between several infoms, relations of their parent infoms n, m are inferred. If all constituents of an infom n are more difficult or have at least the same difficulty as the constituents of an infom m and there exist one child of n that is more difficult than a child of m , then n is more difficult than m . Similarly, a rule can also be used for a **top-down inference**: given the relation of two nested infoms n and m , relations between their constituent infoms can be inferred. If n is more difficult than m than the antecedents in Line 2-4 has to be fulfilled. The antecedents holds if all children of n are more difficult than m . A rule could thus specify that if n is more difficult than m , all children of n are more difficult than the corresponding (i.e., equivalent) child in m .

The formalisations allows to decompose the global relation (`more_difficult_than = exam2000.omdoc`) between the two nested infoms `exam2000.omdoc` and `exam2009.omdoc` into the specific relation (`more_difficult_than = #friends.ex`) between the exercise `friends.ex` and the graph hole `graph_hole`. If the context collector can make such inferences and automatically decompose global constraints into specific ones, the markup effort for users can be reduced.

8.3.3. Variant Grabber

The **variant grabber** takes as input the effective adaptation context Λ_{\square} , the variant context Π_{\square} , the current adaptation result doc_{Δ} , and the hole \square . The variant grabber sorts and filters the variant context Π and returns the most appropriate infom according to Λ_{\square} . The selection of an appropriate variant from Π_{\square} is guided by the narrative and semantic transitions between infoms. The variant grabber implements a strict and a relaxed mode. In the **strict mode**, dependants are not added to a document if the required supporters are not yet part of the document. The dependent infoms are removed from Π_{\square} . The **relaxed mode** permits the insertion of dependant, which supporters are not yet part of the document. In the following, we specify the strict mode for the variant grabber.

³The specification and implementation of respective rules and inference mechanism remains for future research.

8. Substitution of Document Parts

Listing 47: Select infom for $(\square, doc_{\Delta}, \Lambda_{\square}, \Pi_{\square})$

```

1  $\Pi_{\square}$  = filter variant context for  $(\Pi_{\square}, \square, doc_{\Delta})$ 
2  $\Pi_{\square}$  = reorder variant context for  $(\Pi_{\square}, \Lambda_{\square})$ 
3
4 if  $\Lambda$  contains option omit.duplicates then
5    $\Pi_{\square}$  = remove all infoms from  $\Pi_{\square}$ , which are in  $doc_{\Delta}$ 
6 else if  $\Lambda$  contains prefer.duplicates then
7    $\Pi_{\square}$  = prioritise all infoms in  $\Pi_{\square}$ , which are in  $doc_{\Delta}$ 
8 fi
9
10 return first infom in  $\Pi_{\square}$ 

```

Listing 47 specifies the steps for the selection of the most appropriate infom. In a pre-processing step, all infoms, which semantic and narrative transitions are not satisfied, are removed from the variant context. For this the subroutine *filter variant context* in Listing 48 is called. The remaining infoms in Π_{\square} are sorted according to how well the infoms match the adaptation context. For this the subroutine *reorder variant context* in Listing 50 is called. In a third step, the reordered variant context is filtered/ordered further. For this, an option in the adaptation context is processed. It initialised that infoms are prioritised or omitted for the substitution, if they are already included in the document. For example, omitting redundant infoms avoids that an exercise is inserted twice into an exam. In contrast, preferring already included infoms can support a didactic intention of a lecturer, who wants to repeat examples as often as possible in his slides. The most appropriate infom in Π_{\square} is returned.

Listing 48: Filter variant context for $(\Pi_{\square}, \square, doc_{\Delta})$

```

1  $\Pi'_{\square} = \emptyset$ 
2
3 forall  $\iota$  in  $\Pi_{\square}$  do
4    $\iota$  = check infom for  $(\iota, \square, doc_{\Delta})$ 
5   add  $\iota$  to  $\Pi'_{\square}$ 
6 done

```

Listing 48 specifies the filtering of the variant context. It takes as input the variant context Π and returns the filtered variant context Π' . It calls the subroutine *check infom* in Listing 49, which returns the infom ι or none.

Listing 49: Check infom for $(\iota, \square, doc_{\Delta})$

```

1 B = extract all infoms before  $\square$  in  $doc_{\Delta}$ 
2 A = extract all infoms after  $\square$  in  $doc_{\Delta}$ 
3 T = extract all transitions of  $\iota$ 
4
5 forall t in T do
6   P = get prerequisites of t
7   C = get consequence of t
8   if not  $(P \subseteq B)$  or not  $(C \subseteq A)$  then
9     return none
10  fi
11 done
12
13 return  $\iota$ 

```

The hole and the document doc_{Δ} allow the variant grabber to identify all infoms before and after a hole. Thus, preconditions and consequences of semantic and narrative transitions of the infoms in Π_{\square} can be checked and infoms, which transition's requirements are not satisfied, can be removed from Π_{\square} . Listing 49 specifies the subroutine `check infom`, which verifies whether an infom should be consider or omitted. In a first step, the variant grabber extracts all infoms before (constructing the infom set B) and after (constructing the infom set A) the hole in the document. It then retrieves all transitions for the infom ι . If the prerequisites (represented as infom set P) of these transitions are not satisfied by B (i.e., are not a subset of B) or the consequences (represented as infom set C) are not satisfied by A (i.e., are not a subset of A), the subroutine returns `none` and the infom is removed from Π .

Listing 50: Reorder variant context for $(\Pi_{\square}, \Lambda_{\iota})$

```

1  W = empty map
2   $\Pi'_{\square} = \emptyset$ 
3
4  forall  $\iota$  in  $\Pi_{\square}$  do
5     $w(\lambda_{\iota}) = 0$ 
6
7    forall  $cp_j$  in  $\lambda_{\iota}$  do
8      if  $cp_k$  in  $\Lambda_{\square}$  and  $cp_k$  satisfies  $cp_j$  then
9         $w(cp_j) = \text{order of } cp_k$ 
10       fi
11       add  $w(cp_j)$  to  $w(\lambda_{\iota})$ 
12     done
13
14   add key-value pair  $\iota w(\lambda_{\iota})$  to  $W$ 
15 done
16
17 while  $\text{size}(\Pi'_{\square}) < \text{size}(\Pi_{\square})$  do
18    $\iota = \text{get first key from } W \text{ with max value}$ 
19   delete key-value pair from  $W$ 
20   append  $\iota$  to  $\Pi'_{\square}$ 
21 done
22
23 return  $\Pi'_{\square}$ 

```

Listing 50 specifies how the variant context is reordered. Technically, the algorithm implements a generalisation of the matching algorithm of the notation framework (Listing 19). In the first step, the map W is created, which maps the infoms ι in Π_{\square} with their weight. To compute a weight for an infom, the context parameters cp_j in its context annotation λ_{ι} are processed. All cp_j in λ_{ι} that satisfy a context parameter cp_k in Λ_{\square} (Listing 51) are weighted with the order of cp_k . The weight of an infom ι is computed by adding up the weights of the context parameters in λ_{ι} . In a second step, the weight map W is used to create the reordered variant context Π'_{\square} . All elements in W are added to Π'_{\square} as follows: the first key with the maximum value is selected, the key-value pair is deleted from W and is appended to Π'_{\square} .

Listing 51: cp_k satisfies cp_j

```

1   $d = \text{get the context dimension symbol of } cp_k \text{ and } cp_j$ 
2   $v_k = \text{get the context value of } cp_k$ 
3   $v_j = \text{get the context value of } cp_j$ 
4

```

8. Substitution of Document Parts

```
5 if d is a property symbol then  
6   if v_k equals v_j then  
7     return true  
8   else return false  
9   fi  
10 else if d is a relation symbol then  
11   ... //future work :-)  
12 fi
```

In the notation framework, the equality of two context parameters (Listing 19) was based on matching context dimensions and context values (represented as mathematical symbols). The matching algorithm for the substitution considers properties but also relation of infoms. We assume that the properties of these relations (transitivity, symmetry, or partial/total order, etc) have been formalised in a machine-processable form. Listing 51 specifies how the variant grabber evaluates whether a context parameter cp_k satisfies a cp_j in the adaptation context. In a first step, the context dimension d of the two context parameter cp_k and cp_j as well as the respective context values v_k and v_j are extracted. The context dimension is represented as mathematical symbol and is defined in a content dictionary. If it is a property symbol, the context values are compared using the equality method of the notation framework (Listing 19). If it is a relation symbol, its properties are processed in order to compute whether or not cp_k satisfies cp_j . For this, the infoms referenced by v_k and v_j have to be retrieved. Their context annotations provide the relations and allow to compute, e.g., the order of document parts. For example, if d represents the `more_difficult_relation` and $v_k = \#b$ and $v_j = \#a$, then cp_k satisfies cp_j if a is more difficult than b ⁴.

Example For the exam example, the effective adaptation context (with the two context parameters $\Lambda_{\square} = \{\text{more_difficult_than} = \# \text{friends.ex}, \text{not_contained_in} = \text{exam1999.omdoc}\}$) and the variant context with infoms `friends.ex`, `centergraph.ex`, `shortestpath.ex`, and `spalgo.ex` as well as `spanningtree.ex` are passed to the variant grabber.

The `centergraph.ex` exercise is removed from the variant context as its supporter, the `shortestpath.ex` exercise, is not part of the current document doc_{Δ} . The remaining exercises are ordered according to how well their context annotations match the adaptation context. A formalisation of the variant relation `more_difficult_than` allows the variant grabber to process the order of the document parts: the exercise `spalgo.ex` is the most difficult exercise and receives the higher priority. In the reordered variant context, it is followed by the exercises `shortestpath.ex`, `friends.ex`, and `spanningtree.ex`. The first reference is returned and the hole is substituted with `spalgo.ex`.

8.4. Summary & Evaluation of the Substitution

Documents consists of a presentation, structure, and content layer. Adaptation can take place on any of these layers. This work supports the adaptation on the content layer of documents by substituting the original parts of a document with user-specific ones. Technically, the content adaptation consists of two steps. In a first step, the document is abstracted, i.e., specific parts of the document are replaced by holes. In a second step, these holes are substituted (or filled) with user-specific content. Users can guide the abstraction and substitution. The

⁴Note that the pseudocode in Listing 51 suits as illustration only. Its completion is object for future work.

substitution step was implemented as extension of the notation framework (Part II). An elaborated mechanism supports the collection of variant document parts from various sources and the selection of an appropriate variant according to the user's content preferences.

The main contribution of this work is its applicability to narrative documents. Most content planning approaches focus on topic-oriented documents. For example, all eLearning systems, which the author analysed in the scope of her Ph.D., follow the learning object paradigm [WG102], which is an instance of the topic-oriented approach. In Section 2.2.7 two eLearning systems, WELSA and ACTIVEMATH, were chosen for a comparison with the herein described approach.

8.4.1. Comparison with ACTIVEMATH and WELSA

WELSA does not support a substitution of learning objects but implements similar processes for the prioritisation of variant learning objects based on the user's constraints (or learning preferences). In WELSA, recommendations are computed based on pedagogically verified adaptation rules and implicitly modelled learning preferences. Consequently, users have only limited control of the adaptation results: The adaptation is based on a predefined set of learning preferences, which can not be extended, set, or modified by the user. Moreover, the underlying representation of learning object in WELSA is rather poor. Only few mathematical dependencies and properties are explicated. Whether or not recommendations are provided depends on the few marked up dependencies: the relations *requires* and *isRequiredBy* prevent the ordering of learning objects.

The ACTIVEMATH adaptation takes a scenario or document, retrieves and arranges user-specific learning objects, and adapts the presentation of the resulting document to the reader. Technically, a planning engine resolves the configurations from the scenario wizard or the dynamic items. It draws on a set of predefined templates and retrieves the competencies and background of the reader from his learner model. These templates define the selection and ordering of learning objects, users have no control of the output: learning objects are collected from the ACTIVEMATH knowledge base, the selection is based on the captured competencies in the user's learner model.

The limited control of the user is a serious drawback of ACTIVEMATH. Although the templates have been verified with experts, they might not always be appropriate. For example, when creating an exam, a lecturer wants to define the difficulty level or omit specific exercises and does not want to rely on an implicit inference process. However, for both, the scenarios and the resolution of dynamic items, lecturers have no control over the actual structure and content of the output documents – these always adapt to individual user models. The generated material might even mislead students. If students choose the exams simulation, the output exam always adapts to their competencies. Is the student weak, the exam will be easy. This can dim the student's awareness: he might feel competent to take the real exam since he always succeeded with the easy ones. The proposed substitution approach improves the ACTIVEMATH course generation in that it allows users full control of the adaptation process.

Another disadvantage of the ACTIVEMATH approach is the limitation to instructional metadata: Authors can only describe their learning object with a predefined set of dependencies and properties. The underlying ontology of instructional objects is not extensible. Thus, users can not introduce new relations or properties. In contrast, this work proposes an extensible approach. Users can use any context dimension and value. The standardisation of such formalisations and context vocabularies remains future work. It can possibly be achieved in collaboration with the developers of the OMDOC metadata scheme [LK09].

Finally it should be mentioned that the adaptation routines in ACTIVEMATH solely considers instructional dependencies, although the representation of the mathematical materials

8. Substitution of Document Parts

(in OMDOC format) is highly-structured and extremely interlinked. For example, ACTIVE-MATH does not verify the selection of learning objects to assure that all mathematical prerequisites are provided. By taking mathematical and narrative dependencies into account, the herein proposed approach draws on the structure of mathematical knowledge to produce mathematical sound and coherent outputs. It can thus handle highly interconnected document parts and is not limited to self-contained learning objects. The benefit of the exploration of mathematical structures becomes apparent when reordering documents (Section 9).

8.4.2. Proof of Concept

To evaluate the proposed approach, the substitution algorithm has been implemented by a proof-of-concept prototype. In particular, the JOMDOC library [JOM08a] was extended by an *abstract document module* [JOM08b]. Refactoring that integrate analogous functionality between notation and variant module have been performed. Furthermore, the JOMDOC library has been integrated in the *panta rhei* system (Section 10.1) and the new services were discussed with a group of instructors.

8.4.3. Education Case Study

The application of the substitution algorithm for the generation of exams was discussed with a group of instructors of the JEM-Thematic-Network ECP-038208. In the following, their concerns and feedback are summarised.

Collaborative markup of documents The prerequisites for the substitution service is the thorough markup of documents, including the classification of document parts, the annotation of their relations and properties, as well as formalisation of these relations/properties. One participant was concerned that this would increase the authoring effort tremendously. He believes that it would be beneficial if the markup effort could be split between authors and readers.

Splitting the work between authors and readers requires a combination of inline and stand-off markup. Inline markup subsumes all markers within a document, which are usually inserted by the author as other users have no access rights to modify the document. Stand-off markup allows other users to associate metadata (properties and relations) with all parts of the document that can be uniquely identified. Possibly some of the extensional and intensional markup options can be used, e.g., tags (Listing 33).

Apart from the underlying representation, a collaboration of authors and readers requires a collaborative authoring infrastructure. For example, semantic wikis, such as SWIM (see Section 2.2.4), provide an online authoring environment for inline markup. Alternatively, tagging (Section 7.2.2) and web annotations [Koi05] support readers to add stand-off markup. The *panta rhei* system provides an environment for sharing marked-up materials and formalisations (Section 10.3).

Consistency of the Markup The document parts and their explicitly marked interrelation form a graph. We have assumed that these interrelation can be implicitly enriched based on a formal representation of relations and their properties. Although we could proof that the inferred enrichment are correct, one participant pointed out that the set of explicitly marked interrelations is provided by humans and can thus be error-prone and inconsistent.

A prerequisite for a collaborative authoring approach is thus to implement a consistency check, e.g., whenever a new relation or property is inserted. Based on such a preprocessing

step, users can be warned if the new metadata is inconsistent with previous annotations. Possibly, such inconsistencies can partly be resolved automatically. The *locutor* project focuses on change managements problem and can help to address this problem [Mül10].

Requirements From the discussion, a wish list of requirements was derived. Whether these requirements are already supported or, if not, how they possibly could be provided, is discussed below:

- *Prioritization of constraints*: Users can change the order of context parameters in the adaptation context and, doing so, denote their priority.
- *Don't use the same problem twice*: This requirement is supported by the `omit_duplicates` option of the variant grabber (Section 8.3.3).
- *Don't use a version of an included problem*: This requirement can be provided by extending the variant grabber to not only omit elements of the substitution history but also their versions. The prerequisite for this extension is that a respective variant relation (e.g., `is_version_of`) is formalised and marked.
- *Balance the exercises, all should have the same level of difficulty*: This requirement is already supported. Instead of entering a relation such as `more_difficult_than` users can enter a property such as `difficulty_level`, provided that the exercises contain such context parameters. For each exercise in the exam, an equivalent exercise with the same difficulty level can be chosen.
- *Generate equivalent exams*: The herein proposed approach provides the foundation for this requirement. It can be extended to provide many instead of one output document, while omitting to insert a variant twice.

Decomposing Variant Relations One participant was sceptical whether the inheritance of constraints from an exam to its exercises would confuse a user (see Section 8.3.2): “*What does it mean that an exam is more difficult than another exam? Does it mean that all exercises have to be more difficult or is it sufficient if just one of the exercises is more difficult? What if you can only find exercises that are equivalent in their level of difficulty?*”.

The concerns are legitimate but not object of this work. The proposed approach provides the foundation for document adaptations. It is up to the users to define the variant relations and relation properties. For example, they can provide a rule system, such as exemplified in Listing 39. Note that the listing only presents an extract of the required rule system, which has to include the definitions for all relations and concepts used within the rules. Naturally, users should not be expected to formalise all these rules from scratch. Instead, a content planner should be integrated with a system, which provides an initial rule system of fundamental relations and properties and which supports users to collaboratively refine it. The author envisions such a collection of rules analogously to the collaborative refinement of mathematical concepts and definitions by the OPENMATH community [OM-].

8.4.4. Services & Limitations

The proposed substitution framework provides the theoretic foundation for the adaptation of narrative documents. It has been integrated into the interactive environment *panta rhei* (Section 10.1) in order to support the exam generation use case. Its functionality can be used (and extended) to support a variety of services. In a case study on mathematical proof [CM09] and education [MK09b, MK08b, Mül07b] the author was able to gain intuition on other scenarios and services such as:

8. Substitution of Document Parts

1. **User-specific recommendation** of supplementary material (e.g., ‘typical’ examples [KMS92]) according to the user’s background,
2. **Interactive enrichment** of the documents with definitions, motivations, examples, or alternative proofs,
3. **Varying level of detail, expertise, or formality,**
4. **Multilingual presentations.**

In the following, the required functionality is discussed, which allows an interactive environment to support these services.

Substitution Services

A variation of the *level of detail, expertise, and formality* of documents as well as *multilingual presentations* can be easily supported with the proposed substitution workflow. Such services are similar to the adaptation of exams, where exercises are substituted with more difficult variants. The only difference lies in the kind of context parameter that is considered, i.e., the detail, expertise, formality, and language properties of the document parts have to be specified.

Selection Services

Services like the *interactive enrichments* of documents and *user-specific recommendations* build on parts of the substitution workflow. Instead of substituting an original part of the document with a user-specific selection, these services extend the original document with additional, user-specific selections.

The interactive enrichment allows users to manually select an alternative document part and to insert it into their documents. User-specific recommendation automatise this selection based on the user’s preferences. While the former services can easily be provided based on the substitution workflow, the latter requires an extension of this work with a component that maintains user information (Section 10.3).

Open Research Questions

Markup & Formalisation Figure 32 illustrates issues for future research. The proposed approach requires considerable, additional markup efforts by users. Apart from writing and annotating documents, they have to be modularised, semantic and narrative transitions have to be explicated, and variant relations and other context parameter have to be marked and formalised. As mentioned in Part II, such additional efforts will probably not pay off if a user decides to adapt a document only ones. They will rather pay off when maintaining a large collection of documents, which contents are heavily reused and written by numerous authors from different communities and with numerous equivalent examples, definitions, and proofs. Such multi-authored variety of marked-up document content is the prerequisite for the substitution services.

Fortunately, one can make such assumptions in a Ph.D. thesis. To proof the practicality of the approach, all steps in Figure 32 have to be solved. An important open issue is the formalisation of variant relation (and other context parameters) and the specification of a machine-processable inline or stand-off representation. Based on this, the proposed approach can be extended towards a declarative approach, in which a generic content planning mechanism adapts to the user’s formalisation of relations and properties. Open issues, such as in Section 8.3.2 and 8.3.3 can finally be addressed: The context collector can be extended with

8.4. Summary & Evaluation of the Substitution

an inference mechanism that allows to decompose relations between infoms. The matching of the variant grabber can be revised to prioritise infoms according to relation properties, such as transitivity, symmetry, or order.

Collaborative Environment The proposed services should not be provided as stand-alone tool but should rather be integrated in a collaborative environment, thus, allowing users to reuse marked-up materials and formalisations from their colleagues. Moreover, intuitive interfaces (e.g., editors) need to be provided that relieve the user from – as many as possible – formalisation and markup tasks.

Application for the Topic-Oriented World This work implements a first step for the bridging of topic-oriented and narrative world. As a start it applied the topic-oriented principles of modularisation and reuse to the narrative world and successfully supports the adaptation of narrative documents. Future research can observe, whether the findings of this work are beneficial for the topic-oriented world. For example, one could analyse whether a topic-optimised knowledge base could be enriched with the new transition markup to support the assembly of coherent documents. Such an extension could help topic-oriented systems like wikis to provide an import *and* export of narrative documents, while maintaining topics and transitions in their knowledge base.

Dynamic Documents The substitution algorithm is not restricted to the file system but can also be integrated with more intelligent databases [ZK09]. We can also consider the proof assistance systems that were mentioned in the case study on mathematical proofs in [CM09]. If these systems can convert their knowledge bases into a content-oriented format, such as OMDOC, their proofs and examples can be made available for the substitution. For examples, authors could reference the interface of a respective mathematical repository to retrieve alternative proofs, examples, or counter examples.

Holes, represented as `variants` elements, are similar to the *dynamic items* in ACTIVE-MATH, which, if selected, trigger the course generator to create user-specific content. They encode queries (called scenarios in the ACTIVE-MATH jargon), such as ‘illustrate with multiple examples’ or ‘practice with multiple exercises’. While dynamic items are automatically inserted by the course-generator and hard-code the instructions for the course generator, `variants` elements allow the user to guide the substitution process.

In Section 10.1 a web services is illustrated, which provides the foundation for extending `variants` elements towards dynamic items: instead of an adaptation context, users can be supported to insert queries to the web service in their documents. In the current implementation such queries are automatically inserted to retrieve images from the web service.

```
<span class="omdoc-omlet">
  <div id="fimg" class="omdoc-image">
    
  </div>
</span>
```

When extending `variants` elements to encode such queries, users would be able to outline the discourse structure of their documents and place `variants` elements with queries to fill this structure with content. The specification and implementation of such an extension remains future work. Possibly a collaboration with the TNTBASE project can speed up the implementation [ZKR10].

8.4.5. Chapter Summary

In Part II we have solely looked at the adjustment of mathematical notations. However, such adaptations can not be seen independently from the adaptation of the whole document. When entering constraints for the adaptation of notations, alternative document parts can be retrieved that fit better.

This chapter specified the extension of the notation framework for the substitution of document parts according to the semantic, narrative, and user context. The approach gives users full control of the adaptation process. Users can decide, which parts of their semantically marked up documents are adaptable. They can explicitly select the available variants and use context parameters to guide the selection from these variants.

The proposed substitution approach provides the theoretic foundation for the adaptation of narrative documents. It has been integrated into the interactive environment *panta rhei* in order to support the exam generation use case. In discussion with several instructors of the JEM network, issues for future work have been identified. These include support of collaborative markup by authors and readers as well as a change management service to assure the consistency of the collaboratively marked up contents.

Apart from the exam generation use case, the substitution approach implements the foundation for other interactive, user-specific services, such as user-specific recommendations of supplementary materials, the interactive enrichment of documents, the variation of document content with respect to detail, difficulty, and formality, as well as multilingual presentations of documents.

Several issues remain for future work. An important research task is the formalisation of variant relations and other context parameters and the extension of the proposed content adaptation approach towards an extensible, declarative one. Other issues include the capturing of user preferences for adaptive services, the extension of holes towards dynamic items, the implementation of a collaborative environment in which users can exchange marked-up materials and formalisations, as well as the application of transition markup to topic-optimised knowledge bases to support coherent document assemblies.

The main contribution of this work is its applicability to the narrative world. Documents have been modularised into infoms, for which semantic and narrative transitions as well as transitional texts are marked. Considering semantic and narrative transitions has improved the accuracy and coherence of the substitution output. The benefit of this exploration can be made much clearer when *reordering* mathematical documents.

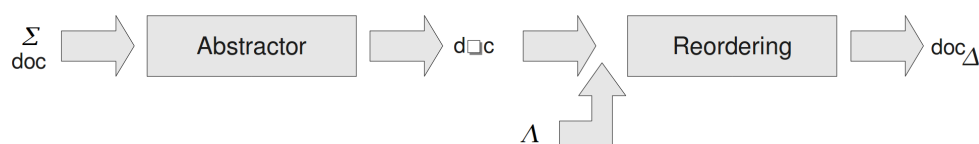
9. Reordering of Mathematical Documents

Two adaptation processes can be distinguished that change the arrangement of paragraphs in a document – the **reordering** and the **restructuring** of infoms. The former process preserve the *discourse structure* of the document (Section 7.1.1). It does not change the grouping of infoms into discourse containers (chapters, section, subsections, and mathematical theories), but solely permutes the infoms within a discourse container. The restructuring process arranges infoms arbitrarily and groups them into a new discourse structure. This work focuses on the *reordering* of documents.

In contrast, to the topic-oriented approach and the learning object paradigm, the proposed approach can be applied to documents that are created in document-centered authoring styles, in which transitional phrases are inserted to improve the coherence of the material. ACTIVE-MATH claims that the (topic-oriented) course planning produces coherent material of good quality since transitional texts are generated to connect the self-contained learning object. However, the author illustrated that generated phrases can never reach the quality of manually written text (Section 7.1). The herein described approach does not require authors to produce independent, self-contained information units but proposes a markup for transitions and transitional texts (Section 7.1.3). This markup is the basis for an automatised, user-specific reordering, which does not reduce the quality and coherence of documents.

9.1. Information Model

The reordering builds on the substitution workflow (Section 8) and is implemented as a two-stage process — the abstraction and the reordering of document parts.



In the first stage, the input document (doc) is abstracted, i.e., certain parts in the document are ‘made sortable’ according to the abstraction context Σ . All other parts are static. The document produced by the first step is called ‘abstract document’. If a part of a document is made ‘sortable’ it is actually annotated with an order, which signifies that its *components* can be rearranged. Users can skip the first step and provide a manually written abstract document.

In the second state, the components of all sortable infoms in the abstract document $d \square c$ are rearranged according to the adaptation context Λ . The adapted document doc_{Δ} is computed based on the following steps: the components (or children) of the sortable infom are collected, the components are reordered, and the original infom is substituted with its reordered copy.

These steps are similar to the processes of the substitution. Its functionality is thus reused. Only few modifications are needed. The *infom collector* provides a different functionality: instead of collecting variants according to the extensional options in the adaptation context, it collects the children of an infom. The context collector is reused. The variant grabber

9. Reordering of Mathematical Documents

is substituted by a *variant sorter*. The next section observe whether the extensional and intensional options of the substitution can be reused.

For the further discussion, we make the same assumption as in the substitution: It is assumed that we can draw on a huge corpus of marked up documents in a content-oriented format such as OMDOC, in which all document parts can be uniquely addressed and their properties as well as semantic and narrative transitions are thoroughly marked.

9.1.1. Extensional and Intensional Options

In the substitution, the extensional options were used to collect alternative document parts, while the intensional options have guided the selection between them. The best matching alternative document part was selected. The reordering draws on the intensional options, only. For example, user-specific context parameters, such as the preferred language, level of detail, or formality, can be provided. Document parts satisfying these user-specific constraints are preferred by the user and are thus placed first in the document. In addition, a predefined set of context parameters is used to initiate one of three ordering strategies: `consider_context`, `consider_sem_trans`, and `consider_nar_trans`.

9.1.2. The Ordering Strategies

The predefined parameters in Section 9.1.1 initiate one of three ordering strategies: the context ordering, the semantic ordering, and the narrative ordering.

Ordering Strategy	Context Ord.	Semantic Ord.	Narrative Ord.
consider context annotation	yes	no	yes
consider semantic transitions	no	yes	optionally
consider narrative transitions	no	no	yes

Context ordering The context ordering arranges document parts according to how well their context annotations match with the adaptation context. It does not expect authors to explicate semantic/narrative transitions and thus imposes less authoring costs.

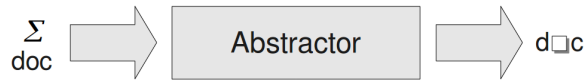
Semantic ordering The semantic ordering arranges document parts according to their semantic transitions, where prerequisites (the supporters) are placed first. For example, two theories are ordered according to the theory morphisms (or imports), which connect these theories. If theory A imports B , then imported theory B is placed first. The import from B to A is interpreted as semantic transition. The semantic ordering requires authors to mark the semantic structure of their documents and thus increases their authoring efforts.

Narrative ordering The narrative ordering considers the narrative transitions in a document. Optionally it can also consider semantic transitions. It imposes extensive authoring efforts: semantic and narrative transitions have to be marked. In return, it produces the most coherent output as transitional texts are preserved to guide the reader through the document.

The three ordering strategies should be used in combination. A reordering algorithm can first try a narrative ordering and, if it fails, either apply the semantic ordering (if semantic transitions should be considered) or directly refer to the context ordering. Such a combination leaves it to the author to decide how much effort he wants to invest in the markup. For example, he can solely explicate narrative transitions between the chapters and sections in his document and initiate that all remaining document parts are ordered semantically or according to their context parameters.

9.2. The Abtractor

We can reuse the abstraction as described in Section 8.2. The abtractor takes as input a document *doc* and an abstraction context Σ .



The abstraction context is represented as set of infom references $\#_i$, which identifies all *sortable* infoms in the document, which components can be rearranged. The abtractor returns the abstracted document $d\Box c$, in which all sortable infoms have been annotated with an order, which signifies that its *components* can be rearranged. Users can skip the first step and provide a manually written abstract document.

Annotating the order of infoms Each infom that is referenced by the abstraction context is annotated with the contextual key-value pair (`order = true`). It can be represented using the `ic` attribute in the OMDOC namespace or the OMDOC metadata markup (Section 4.4.2).

Listing 52: XML representation for (`order=true`)

```

<omgroup ic="order:true">
  <theory>
    <metadata>
      <meta property="cc:order">cc:true</meta>
    </metadata>...
  </theory>
  <theory />
</omgroup>
  
```

Listing 52 illustrates both markup alternatives. They initialise the ordering of the children of the `omgroup` and `theory` infom.

<pre> <omdoc ..> <theory xml:id="algorithm"> <imports from="#spanning_tree" /> <definition xml:id="v1" .../> <example xml:id="v2" .../> <example xml:id="v3" .../> <example xml:id="v4" .../> </theory> <theory xml:id="spanning_tree"> <definition xml:id="v5" ic="type:definition;style:formal"> <CMP>A spanning tree</CMP> </definition> <definition xml:id="v6" ic="type:definition;style:formal"> <CMP>A spanning tree</CMP> </definition> <example for="v5 v6" xml:id="v7"> ... </example></theory></omdoc> </pre>	<pre> <omdoc ..> <theory xml:id="algorithm" ic="order:true"> <imports from="#spanning_tree" /> <definition xml:id="v1" .../> <example xml:id="v2" .../> <example xml:id="v3" .../> <example xml:id="v4" .../> </theory> <theory xml:id="spanning_tree" ic="order:true"> <definition xml:id="v5" ...> ... </definition> <definition xml:id="v6" > ... </definition> <example for="v5 v6" xml:id="v7" ic="type:example;style:illustrative"> <CMP><omlet data="#span-img" .../> </CMP> </example> </theory></omdoc> </pre>
--	---

Figure 47.: Abstraction of GENCS lecture notes 2009

Example Figure 47 shows an extract of the lecture notes before (left hand side) and after (right hand side) the abstraction. The original document contains two theory elements, which embed definitions, examples, and text. The abstraction context $\Sigma = \{ //theory \}$ is given. It defines that all theories are sortable, i.e., their constituents can be reordered. All remaining elements are static. In the abstract document, all theories are marked with an `ic` attribute, with value `order:true`. The markup for the narrative transition between the two theories is illustrated in Listing 31.

9.3. The Reordering Algorithm

The reordering algorithm is based on the substitution algorithm and is, thus, a generalisation of the rendering algorithm in Section 4. The variant grabber is replaced with a *variant sorter*. All other component names are reused. Figure 48 illustrates the components of the reordering.

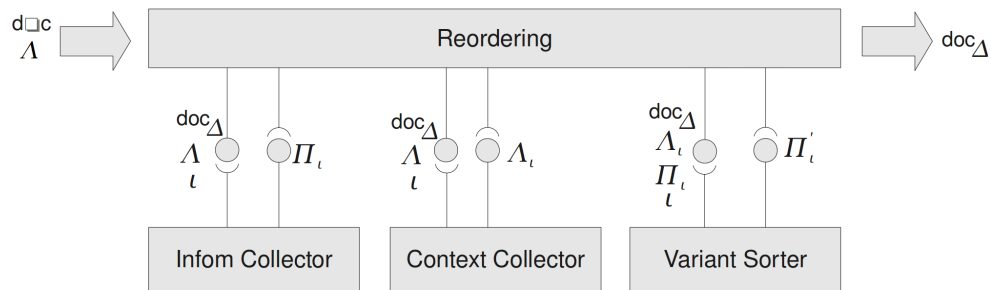


Figure 48.: Components of the reordering algorithm

The general reordering algorithm is specified as follows: The **reordering** takes as input an abstract document $d\Box c$ and an adaptation context Λ . The algorithm outputs a variant of $d\Box c$ (doc_Δ), in which the components of all sortable infoms have been rearranged.

Listing 53: Producing doc_Δ

```

1   $doc_\Delta = d\Box c$ 
2
3  forall sortable infoms  $\iota$  in  $doc_\Delta$  do
4     $\iota' = \text{reorder infom for } (\iota, doc_\Delta, \Lambda)$ 
5     $doc_\Delta = \text{substitute } \iota \text{ with } \iota'$ 
6  done
7
8   $doc_\Delta = \text{remove all obsolete transition infoms in } doc_\Delta$ 

```

Listing 53 specifies how the user-specific document doc_Δ is constructed from the abstract document $d\Box c$. The document doc_Δ denotes the current adaptation result. It is initialised with the abstract document $d\Box c$ and incrementally modified (Line 3-6): doc_Δ is traversed from left to right and the constituents of every sortable infom ι are rearranged. For the reordering of the constituents of ι , the subroutine `reorder infom` (Listing 54) is called. In a final step, a *consistency check* is applied to doc_Δ , which removes all obsolete transition infoms (Listing 45).

Listing 54: Reorder infom for $(\iota, doc_{\Delta}, \Lambda)$

```

1  $\Pi_{\iota}$  = construct variant context for  $(\iota, \Lambda, doc_{\Delta})$ 
2
3 if  $\Pi_{\iota}$  is empty then
4   return  $\iota$ 
5 fi
6
7  $\Lambda_{\iota}$  = compute effective adaptation context for  $(\iota, \Lambda, doc_{\Delta})$ 
8  $\Pi'_{\iota}$  = reorder variant context for  $(\iota, \Lambda_{\iota}, \Pi_{\iota}, doc_{\Delta})$ 
9  $\Pi'_{\iota}$  =  $\emptyset$ 
10
11 forall  $c$  in  $\Pi_{\iota}$  do
12    $c'$  = reorder infom for  $(c, doc_{\Delta}, \Lambda)$ 
13   add  $c'$  to  $\Pi'_{\iota}$ 
14 done
15
16 return reorder infom for  $\Pi'_{\iota}$ 

```

Listing 54 specifies the reordering algorithm. In a first step, the *infom collector* is called (Line 1). It takes as input the infom ι , the adaptation context Λ , and the current document doc_{Δ} and collects the constituents (or children) of the infom. The variant context Π is returned and includes these infoms in the order they appeared in infom ι . Next the *context collector* is called (Line 7). It takes as input the infom ι , the adaptation context Λ , and the current document doc_{Δ} . It returns the effective adaptation context Λ_{ι} by reusing the context collector of the substitution (Section 8.3.2). Note that specific context dimensions are used to guide the ordering, i.e., `consider_context`, `consider_sem_trans`, and `consider_nar_trans`, which can have a context value `true` or `false`. In a third step, the *variant sorter* is called (Line 8). It takes as input the infom ι , the effective adaptation context Λ_{ι} , the variant context Π_{ι} , and the current document doc_{Δ} . The reordered variant context is returned. The reordering algorithm recurses over each sortable infom in Π_{ι} (Line 11-14). The recursion terminates if a variant context includes no more sortable infoms (Line 3-5). Finally, the infom is reordered according to Π'_{ι} and returned (Line 16).

While the infom collector and context collector are reused by any of the three ordering strategies, the *variant sorter* receives a different functionality. The following section introduce the different strategies for the variant sorter. These are discussed in separation, though, a reference implementation such as *panta rhei* (Section 10.1) should provide an integrative approach that combines the strategies.

Example The figure to the left illustrates the infom graph for the GENCS lecture notes in Section 6.3.2. The semantic transitions in the infom graph have been extracted from the OM-DOC representation in Figure 47, i.e., by parsing the `for` attributes and `imports` elements. Accordingly, the theory `algorithm` (A) depends on the `spanning_tree` (S). The constituents of the two theories also depend on each other.

The examples of the `algorithm` theory ($v2, v3, v4$) depend on the definition ($v1$), while the example ($v7$) in the theory `spanning_tree` depends on both definitions ($v5, v6$). In addition, the markup for narrative transitions has been parsed, one narrative transition ($n1$) was extracted. It relates the theories in opposite direction to their semantic transition ($s6$).

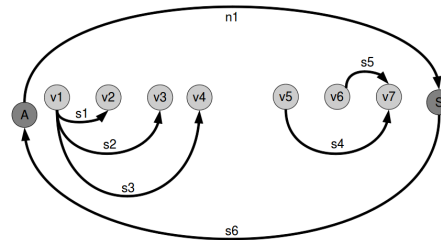


Figure 49.: Infom graph for GENCS

9.3.1. The Context Ordering

The context ordering orders the infoms in a document according to how well their context annotations match with the adaptation context. To do so, it reuses the functionality of the variant grabber (Section 8.3.3).

Listing 55 shows the reordered GENCS lecture notes, where the abstract document in Figure 47 and the global adaptation context $\Lambda = \{(type = example), (style = illustrative), (style = text)\}$ was used as input. The order of the theories remains unchanged but their constituents have been rearranged according to how well their context annotation match with the adaptation context: Examples with images (i.e., *illustrative* examples) are preferred and placed first. Next, examples with text and examples with pseudocode are shown. Finally, definitions and other paragraphs (such as the `omtext` in the `algorithm` theory) are inserted. The rendered document does not correspond to Figure 31 since the order of the two theories remains unchanged.

Listing 55: Representation of the reordered lecture notes

```

<omdoc ..>
  <theory xml:id="algorithm" ic="order:true">
    <metadata><dc:title>Algorithms</dc:title></metadata>
    <imports from="#spanning_tree" />
    <example for="v1" xml:id="v4" ic="type:example;style:illustrative">
      <CMP>
        <omlet data="#img" action="display" style="width:400;height:366" show="embed" />
        <private xml:id="king"><data format="image/jpeg" href="slides/img/kruskal.gif" />
        </private>
      </CMP>
    </example>
    <example for="v1" xml:id="v3" ic="type:example;style:text">
      <CMP>Kruskal algorithm ...</CMP>
    </example>
    <example for="v1" xml:id="v2" ic="type:example;style:pseudocode">
      <CMP>Pseudocode for ...</CMP>
    </example>
    <definition xml:id="v1" ic="type:definition;style:formal">
      <CMP>An algorithm is a ....</CMP>
    </definition>
    <omtext type="transition" for="#n1">
      <CMP>We have used the term spanning tree without defining it. Let us do that now.</CMP>
    </omtext>
  </theory>
  <theory xml:id="spanning_tree">
    <metadata><dc:title>Spanning Tree</dc:title></metadata>
    <example for="v5 v6" xml:id="v7" ic="type:example;style:illustrative">
      <CMP><omlet data="#span-img" ...></CMP>
    </example>
    <definition xml:id="v5" ic="type:definition;style:formal">
      <CMP>A spanning tree ....</CMP>
    </definition>
    <definition xml:id="v6" ic="type:definition;style:formal">
      <CMP>A spanning tree ....</CMP>
    </definition>
  </theory>
</omdoc>

```

To conclude, the context ordering only provides a primitive ordering of the document. Semantic and narrative transitions are not considered. It should only be used as fallback if the other ordering strategies fail.

9.3.2. The Semantic Ordering

The semantic ordering arranges the infoms in a document according to their semantic transitions. The ordering is done in two stages. In a first stage, the variant sorter constructs an infom graph from the elements in $\Pi\iota$. In a second stage, a semantic traversal (Section 7.3) is applied to construct a traversal path P , which visits all nodes of the infom graph by processing only the edges with semantic labels.

Listing 56: Semantic traversal of infom graph \mathcal{G}

```

1   $P = \langle \rangle$ 
2
3  while not all non-empty nodes of  $\mathcal{V}$  in  $P$  do
4     $p =$  last node of  $P$ 
5     $v =$  get successor for  $(p, P, \mathcal{G})$ 
6
7    if  $v$  is none then
8       $v =$  get node for  $(P, \mathcal{V})$ 
9    fi
10
11   add  $v$  to  $P$ 
12 done

```

Listing 56 specifies the semantic traversal algorithm for constructing a traversal path P through \mathcal{G} . It takes as input the infom graph \mathcal{G} and returns P . The traversal path is constructed as follows: As long as not all non-empty nodes in \mathcal{V} are visited by P , the next node is selected. The algorithm first selects the last node in P and calls the subroutine `get successor` in Listing 57, which returns the successor of p in P . The subroutine returns `none` if p is `none` or no successor for p could be found. If no node v could be selected, the subroutine `get node` in Listing 58 is called and the return value (a node or `none`) is added to the traversal path.

Listing 57: Get successor for (p, P, \mathcal{G})

```

1  if  $p$  not none then
2     $v =$  get node for  $(P, \mathcal{V}_{sd}(p))$ 
3
4    if  $v$  is none then
5       $p' =$  predecessor of  $p$  in  $P$ 
6       $v =$  get successor for  $(p', P, \mathcal{G})$ 
7    fi
8
9    return  $v$ 
10 else return none
11 fi

```

Listing 57 specifies the algorithm for finding a successor of p . It takes as input the node p , the path P , and the infom graph \mathcal{G} and outputs the successor for p . The successor is selected as follows: If p is `none`, `none` is returned. Otherwise, a node v is selected by the subroutine `get node` in Listing 58. It processes $\mathcal{V}_{sd}(p)$, the set of dependants of p , and

9. Reordering of Mathematical Documents

returns the most appropriate dependant. If no appropriate node can be found, the algorithm is called recursively for the predecessor of p .

Listing 58: Get node for (P, \mathcal{V})

```

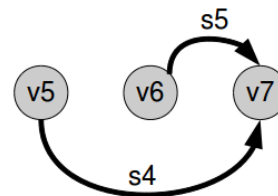
1   $\mathcal{V}' =$  get nodes from  $\mathcal{V}$  where
2     $v$  is not in  $P$  and //remove to produce a walk
3     $deg_{sti}(v) == 0$  and
4     $deg_{si}(v)$  is maximal and
5     $deg_{so}(v)$  is maximal and
6
7    forall  $d_i$  in  $\mathcal{V}_{sd}(v)$  do
8      forall  $s_i$  in  $\mathcal{V}_{ss}(d_i)$  do
9         $deg_{sti}(s_i) == 0$ 
10     done
11   done
12 done
13
14 if size of  $\mathcal{V}' == 0$  then
15   return none
16 else return first node in  $\mathcal{V}'$ 
17 fi

```

Listing 58 specifies the selection of a node given the traversal path P and the set of nodes \mathcal{V} . It selects a node v from \mathcal{V} that satisfies the following conditions:

- v must not be already on the traversal path. This condition can be removed to produce a traversal walk.
- The dependant's semantic traversal in-degree $deg_{sti}(v)$ is zero, i.e., all supporters of v have already been visited.
- The semantic in-degree of v is maximal. A node v with a semantic traversal in-degree of zero and a maximal in-degree is a dependent node in \mathcal{V} , which has a maximal number of supporters. These supporters are all on the path. The node is thus more connected to the nodes in P than other nodes.
- The semantic out-degree of v is maximal. A node v with a maximal out-degree is the supporter for several other nodes. Selecting this node assures that the semantic traversal can process further semantic edges and does not have to add disconnected nodes to the path in the next traversal iteration.
- $\mathcal{V}_{sd}(v)$ is the set semantic dependants. For a dependant d_i in $\mathcal{V}_{sd}(v)$, $\mathcal{V}_{ss}(d_i)$ is the set of its supporters. v is preferred, if all supporters s_i in $\mathcal{V}_{ss}(d_i)$ have a traversal in-degree of zero.

The sequence diagram in Figure 50 illustrates the traversal of the graph to the right. It was constructed for the `spanning_tree` theory of our GENCS example. The graph includes two self-contained nodes (v_5, v_6) and one dependant node (v_7), which supporters are v_5 and v_6 . Since P is still empty, the node p is `none`. The subroutine `get_successor` (Listing 57) is called for $p = \text{none}$, $P = \langle \rangle$, and \mathcal{G} and returns `none`. Since `none` node is selected, the subroutine `get_node` (Listing 58) is called for P and $\mathcal{V} = \langle v_5, v_6, v_7 \rangle$. From all nodes in \mathcal{V} , the nodes v_5 and v_6 fulfil all conditions. The first node (v_5) in \mathcal{V}' is returned. In the second iteration, `get_successor` is called for $p = v_5$,



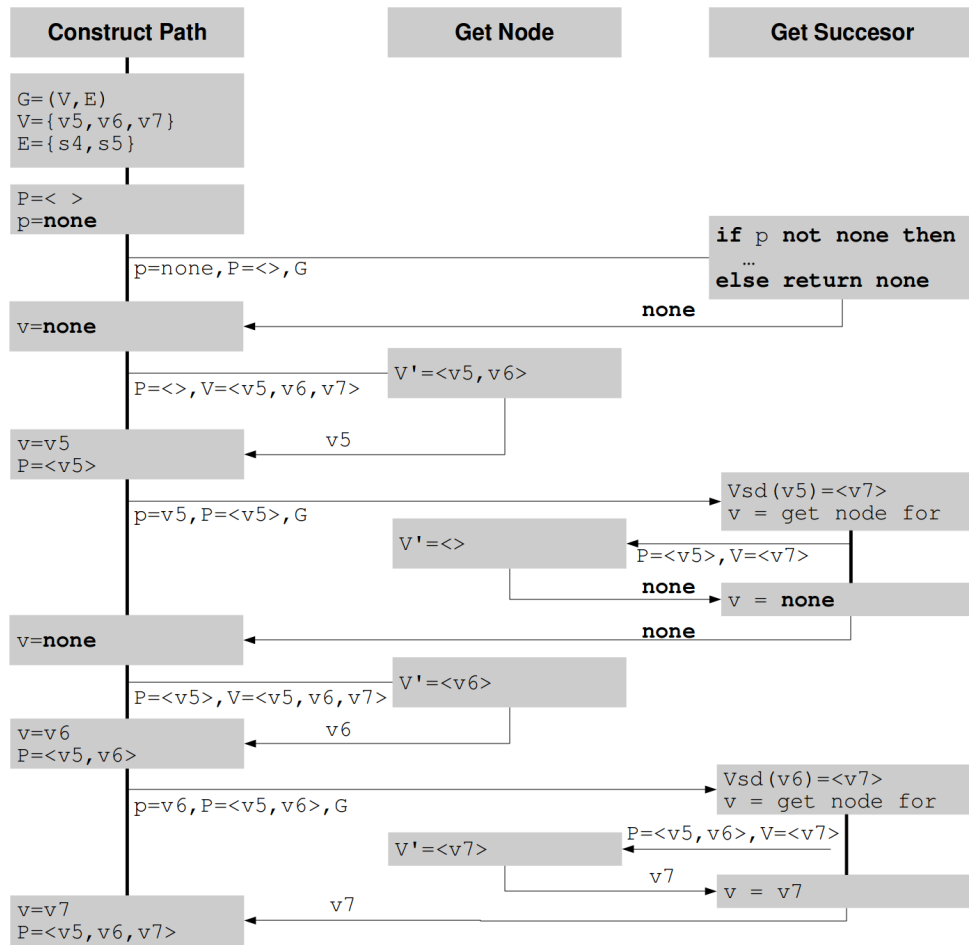


Figure 50.: Semantic ordering of the spanning_tree theory

$P = \langle v_5 \rangle$, and \mathcal{G} . It calls `get node for` P and $\mathcal{V} = \langle v_7 \rangle$, where \mathcal{V} includes all dependants of v_5 . None of the dependants fulfils the conditions and `none` is returned and passed on to the main method `construct path`. The method calls `get node for` $P = \langle v_5 \rangle$ and $\mathcal{V} = \langle v_5, v_6, v_7 \rangle$. The node v_6 is returned. In a final iteration, `get successor` is called for $p = v_5$, $P = \langle v_5, v_6 \rangle$, and \mathcal{G} . It calls `get node for` P and $\mathcal{V} = \langle v_7 \rangle$, where \mathcal{V} includes all dependants of v_6 . This time v_7 fulfils all conditions. It is passed on to the main method `construct path`. The path $P = \langle v_5, v_6, v_7 \rangle$ was constructed.

Listing 59: Representation of the reordered lecture notes

```

<omdoc>
  <theory xml:id="spanning_tree" ic="order:true">
    <metadata><dc:title>Spanning Tree</dc:title></metadata>
    <definition xml:id="v5" ic="type:definition;style:formal">
      <CMP>A spanning tree ....</CMP>
    </definition>
    <definition xml:id="v6" ic="type:definition;style:formal">
      <CMP>A spanning tree ....</CMP>
    </definition>
  </theory>

```

9. Reordering of Mathematical Documents

```

</definition>
<example xml:id="v7" for="v5 v6" ic="type:example;style:illustrative">
  <CMP><omlet data="#span-img" ...></CMP>
</example>
</theory>
<theory xml:id="algorithm" ic="order:true">
  <metadata><dc:title>Algorithms</dc:title></metadata>
  <imports from="#spanning_tree" />
  <definition xml:id="v1" ic="type:definition;style:formal">
    <CMP>An algorithm is a ....</CMP>
  </definition>
  <example xml:id="v2" ic="type:example;style:pseudocode">
    <CMP>Pseudocode for ...</CMP>
  </example>
  <example xml:id="v3" ic="type:example;style:text">
    <CMP>Kruskal algorithm ...</CMP>
  </example>
  <example xml:id="v4" ic="type:example;style:illustrative">
    <CMP>
      <omlet data="#king" action="display" style="width:400;height:366" show="embed" />
      <private xml:id="king">
        <data format="image/jpeg" href="slides/img/kruskal.gif" /></private>
      </CMP>
    </example>
    <omtext type="transition" for="#n1">
      <CMP>We have used the term spanning tree without defining it. Let us do that now.<
        CMP>
    </omtext>
  </theory>
</omdoc>

```

Listing 59 shows the result of the semantic ordering for the abstract document in Figure 47. The order of the theories has changed: `spanning_tree` is placed first followed by the `algorithm` theory, which imports `spanning_tree`. The order of their constituents has also changed.

The resulting flow of material is not optimal: The `algorithm` theory contains the transitional text “*We have used the term ‘spanning tree’ without defining it. Let’s do that now.*”. The transitional text remains in the document and reduces the coherence of the output document. Consequently, solely considering semantic transitions to reorder a document is not sufficient. In the next section we discuss how narrative transitions can be taken into account to produce a more coherent results.

9.3.3. The Narrative Ordering

The narrative ordering arranges infoms according to the narrative transitions between them. The narrative ordering is done in two stages. In a first stage, the variant sorter constructs an infom graph \mathcal{G} from the elements in Π_ℓ . In a second stage, a narrative traversal is applied to construct a traversal path P , which visits all nodes in Π_ℓ by processing the narrative edges. Optionally, the narrative ordering considers semantic transitions between the document parts. For this, a hybrid traversal is applied in the second step. In the following, we first specify the hybrid traversal and then outline the narrative traversal.

Hybrid TraversalListing 60: Hybrid traversal for $(\mathcal{G}, \Lambda_\iota)$

```

1   $P = \langle \rangle$ 
2
3  while not all non-empty nodes of  $\mathcal{V}$  in  $P$  do
4     $p =$  last node of  $P$ 
5     $P' =$  get narrative path for  $(p, P, \mathcal{G}, \Lambda_\iota)$ 
6
7    if  $P'$  is none then
8       $\mathcal{V}' = \mathcal{V} \setminus P$ 
9       $\mathcal{G}' = (\mathcal{V}', \mathcal{E})$ 
10      $P' =$  get semantic path for  $\mathcal{G}'$ 
11   fi
12
13   append  $P'$  to  $P$ 
14 done

```

Listing 60 specifies the hybrid traversal algorithm. It takes as input the infom graph \mathcal{G} and the context annotation Λ_ι . It returns P . As long as not all nodes in \mathcal{V} are visited by P , the following steps are repeated. The algorithm first selects the last node in P and calls the subroutine `get narrative path` in Listing 61. It returns the longest path from the set of narrative walks \mathcal{N} in P , preferably a walk that starts with p . If `none` path can be selected, a subgraph \mathcal{G}' is constructed, which includes all nodes of \mathcal{V} , which are not yet on the path P . The subroutine `get semantic path` in Listing 56 is called and applies a semantic traversal of the subgraph \mathcal{G}' . Finally, the path P' is appended to P . The append function omits all nodes at the beginning of P' that occur in this order at the end of P . For example, $P = \langle v_5, v_6, v_7 \rangle$ and $P' = \langle v_7, v_8 \rangle$ are merged to $P = \langle v_5, v_6, v_7, v_8 \rangle$ (Figure 51).

Listing 61: Get narrative path for $(p, P, \mathcal{G}, \Lambda_\iota)$

```

1   $P' =$  longest path  $\langle v_1, \dots, v_j, \dots, v_i \rangle$  in  $\mathcal{N}$  where
2     $P'$  starts with  $p$  and
3     $\lambda_1, \dots, \lambda_n$  best match with  $\Lambda_\iota$  and
4     $\langle v_2, \dots, v_i$  are not in  $P$  xor
5     $v_1, \dots, v_j$  in  $P = \langle u_1, \dots, u_k \rangle$  where  $j < k$  and  $\langle v_1, \dots, v_j \rangle$  equals  $\langle u_{k-j}, \dots, u_k \rangle$ 
6  done
7
8  if  $P'$  is none then
9     $P' =$  longest path  $\langle v_1, \dots, v_j, \dots, v_i \rangle$  in  $\mathcal{N}$  where
10      $\lambda_1, \dots, \lambda_n$  best match with  $\Lambda_\iota$  and
11      $\langle v_1, \dots, v_i$  are not in  $P$  xor
12      $v_1, \dots, v_j$  in  $P = \langle u_1, \dots, u_k \rangle$  where  $j < k$  and  $\langle v_1, \dots, v_j \rangle$  equals  $\langle u_{k-j}, \dots, u_k \rangle$ 
13   done
14 fi
15
16 return  $P'$ 

```

Listing 61 specifies the algorithm for finding a narrative path. It takes as input the node p , the path P , the infom graph \mathcal{G} , and the context annotation Λ_ι . It outputs a path P' . The path is selected from \mathcal{N} , the set of narrative walks in \mathcal{G} . In a first step, the algorithm tries to select the longest path $P' = \langle v_1, \dots, v_j, \dots, v_i \rangle$ from \mathcal{N} . The path has to start with node p , the nodes context annotations $\lambda_1, \dots, \lambda_n$ should best match with Λ_ι (Listing 62), and either the nodes v_2, \dots, v_i must not be on P or, given the path $P = \langle u_1, \dots, u_k \rangle$, the sequence $\langle v_1, \dots, v_j \rangle$ at the

9. Reordering of Mathematical Documents

beginning of P' has to be equal to the sequence $\langle u_{k-j}, \dots, u_k \rangle$ at the end of P . The latter condition can be removed to construct a walk. If no path can be selected, the algorithm tries to select a longest path from \mathcal{N} with arbitrary start node v_1 , where the context annotations $\lambda_1, \dots, \lambda_n$ best match with Λ_ℓ (Listing 62). The path or none is returned.

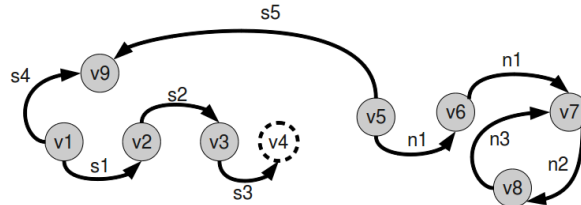
Listing 62: Compute match value for (P, Λ_ℓ)

```

1   $w(P) = 0$ 
2
3  forall  $v$  in  $P$  do
4     $w(\lambda_v) = 0$ 
5
6    forall  $cp_j$  in  $\lambda_v$  do
7      if  $cp_k$  in  $\Lambda_\square$  and  $cp_k$  satisfies  $cp_j$  then
8         $w(cp_j) = \text{order of } cp_k$ 
9      fi
10     add  $w(cp_j)$  to  $w(\lambda_v)$ 
11   done
12
13    $w(P) = w(P) + w(\lambda_v)$ 
14 done
15
16 return  $w(P)$  : size of  $P$ 

```

The herein proposed approach supports authors to provide a variety of transitions between the infoms of their documents, from which an appropriate alternative is selected and an appropriate transitional text is displayed (if provided). The display of only appropriate transition infoms is guaranteed by applying a *consistency check* on the final ordering result doc_Δ , which removes all obsolete transition infoms (Listing 45). The selection of an appropriate transition is performed during the ordering. For this, the context annotations of the nodes on a narrative walk are matched with the adaptation context. Listing 62 illustrates the matching, which reuses the functionality of the variant grabber (Listing 50): the match value of a path is the average weight of its nodes.



The sequence diagram in Figure 51 illustrates the hybrid traversal of the graph above. For simplicity we omit the matching of context annotations. At the beginning of the traversal, none node p is selected and the subroutine `get narrative path` (Listing 61) is called for p , P , \mathcal{G} , and Λ_ℓ . It returns the longest path $\langle v_5, v_6, v_7 \rangle$ in \mathcal{N} . The path is appended to P . In the next iteration, p is initialised with the last node v_7 in P . The subroutine `get narrative path` returns the longest path $\langle v_7, v_8 \rangle$ in \mathcal{N} , which fulfils the conditions in Line 2-5: the path starts with v_7 and the sequence $\langle v_7 \rangle$ at the beginning of P' is equal to the sequence $\langle v_7 \rangle$ at the end of P . The path P' is appended to P , where the redundant node at the beginning of P' is omitted. In a final iteration, p is initialised with v_8 . The subroutine `get narrative path` returns none. A graph \mathcal{G}' is initialised with the remaining nodes in \mathcal{V} and the subroutine `get semantic path` is called (Listing 56). It returns the semantic path $\langle v_1, v_2, v_3, v_9 \rangle$, which is appended to P .

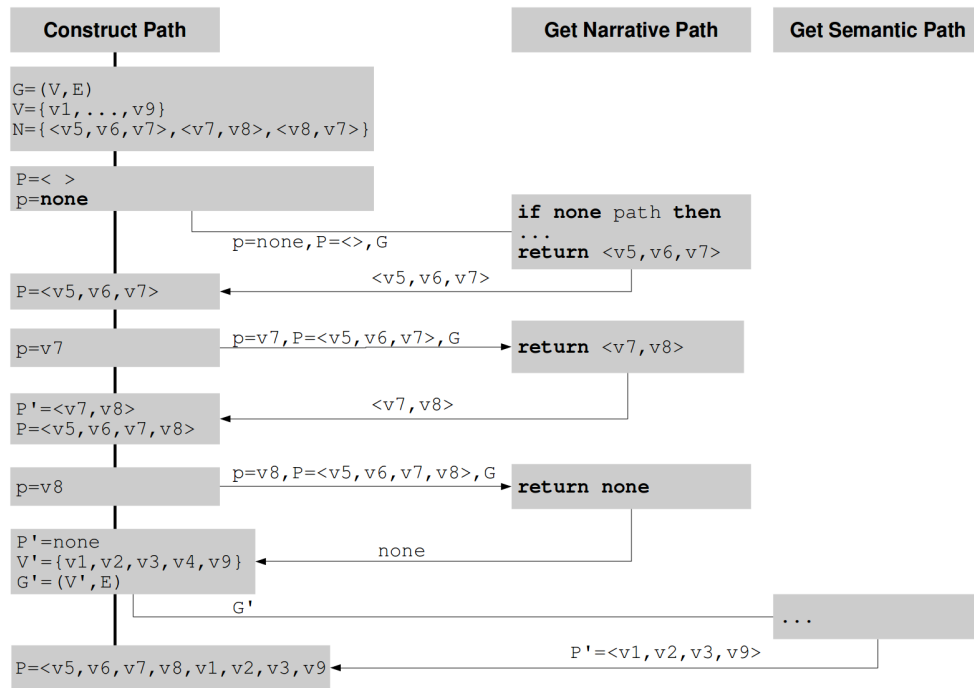


Figure 51.: Hybrid traversal of an infom graph

Narrative Traversal

Listing 63: Narrative traversal for (\mathcal{G}, Λ_t)

```

1  P = ⟨⟩
2
3  while not all non-empty nodes of  $\mathcal{V}$  in P do
4    p = last node of P
5    P' = get path for (p, P,  $\mathcal{G}, \Lambda_t$ )
6
7    if P' is none then
8       $\mathcal{V}' = \mathcal{V} \setminus P$ 
9      P' = glue a path for  $\mathcal{V}'$ 
10   fi
11
12   append P' to P
13 done
  
```

Listing 63 specifies the narrative traversal. As long as not all nodes in \mathcal{V} are visited by P , the algorithm first selects the last node in P and calls the subroutine `get narrative path` (Listing 61). Instead of applying a semantic traversal on the remaining nodes in \mathcal{G} (such as done by the hybrid traversal), the narrative traversal simply glues all remaining nodes together (Listing 64). The glue is visualised by a specific infom, the *glue-infom*, which includes the warning message. The XML representation of a glue-infom is presented in Listing 65: It is represented with a `note` element of type `ednote`.

9. Reordering of Mathematical Documents

Listing 64: Glue a path for \mathcal{V}, Λ_t

```

1  $\mathcal{V}' = \text{reorder } \mathcal{V} \text{ according to } \Lambda_t$ 
2  $P = \langle \rangle$ 
3
4 forall  $v$  in  $\mathcal{V}'$  do
5   append glue-infom to  $v$ 
6   append  $v$  to  $P$ 
7 done
8
9 return  $P$ 

```

Listing 64 specifies the algorithm for glueing a set of nodes together. In a first step, the algorithm applies a context ordering by drawing on the functionality of the variant grabber (Section 8.3.3). An empty path P is initialised. In a second step, all nodes in the reordered set of nodes \mathcal{V}' are appended with a glue child and added to P . Finally, P is returned.

Listing 65: Representation of the reordered lecture notes

```

<omdoc ..>
  <theory xml:id="algorithm" ic="order:true">
    <metadata><dc:title>Algorithms</dc:title></metadata>
    <imports from="#spanning_tree" />
    <definition xml:id="v1" ic="type:definition;style:formal">
      <CMP>An algorithm is a ....</CMP>
    </definition>
    <note type="ednote">
      <CMP>WARNING: NO APPROPRIATE TRANSITION AVAILABLE.
      <link href="">Click here to insert a transition.</link></CMP>
    </note>
    <example xml:id="v2" for="v1" ic="type:example;style:pseudocode">
      <CMP>Pseudocode for ...</CMP>
    </example>
    <note type="ednote">
      <CMP>WARNING: NO APPROPRIATE TRANSITION AVAILABLE.
      <link href="">Click here to insert a transition.</link></CMP>
    </note>
    <example xml:id="v3" for="v1" ic="type:example;style:text">
      <CMP>Kruskal algorithm ...</CMP>
    </example>
    <note type="ednote">
      <CMP>WARNING: NO APPROPRIATE TRANSITION AVAILABLE.
      <link href="">Click here to insert a transition.</link></CMP>
    </note>
    <example xml:id="v4" for="v1" ic="type:example;style:illustrative">
      <CMP>
        <omlet data="#king" action="display" style="width:400;height:366" show="embed" />
        <private xml:id="king"><data format="image/jpeg" href="slides/img/kruskal.gif" /></private>
      </CMP>
    </example>
    <omtext type="transition" for="#n1">
      <CMP>We have used the term spanning tree without defining it. Let us do that now.</CMP>
    </omtext>
  </theory>

```

```

<theory xml:id="spanning_tree" ic="order:true">
  <metadata><dc:title>Spanning Tree</dc:title></metadata>
  <definition xml:id="v5" ic="type:definition;style:formal">
    <CMP>A spanning tree ....</CMP>
  </definition>
  <note type="ednote">
    <CMP>WARNING: NO APPROPRIATE TRANSITION AVAILABLE.
    <link href="">Click here to insert a transition.</link></CMP>
  </note>
  <definition xml:id="v6" ic="type:definition;style:formal">
    <CMP>A spanning tree ....</CMP>
  </definition>
  <note type="ednote">
    <CMP>WARNING: NO APPROPRIATE TRANSITION AVAILABLE.
    <link href="">Click here to insert a transition.</link></CMP>
  </note>
  <example xml:id="v7" for="v5 v6" ic="type:example;style:illustrative">
    <CMP><omlet data="#span-img" ...></CMP>
  </example>
</theory>
</omdoc>

```

Listing 65 shows the result of the narrative ordering (based on a narrative traversal) for the abstract document in Figure 47. The order of the theories remains unchanged, since semantic transitions are ignored. A narrative transition leads from the `algorithm` to the `spanning-tree` theory. Since no transitions connect the constituents of the theories, a path was constructed, which simply glues these paragraphs together. The *glue* includes a warning message, which encourages users to insert missing narrative transitions and, thus, to improve the coherence of their documents. The output document is more coherent than the output by the semantic ordering. Obviously, the GENCS example document is too simple. Section 9.4.3 thus discusses a more complex example, a straw-man solution.

To conclude, the narrative ordering produces the most coherent output. In order to generate the preferred GENCS lecture notes for the students, a combination of the ordering strategies is necessary. This can be implemented by supporting users to associate a specific ordering strategy with different document parts or document part categories. For example, a student might want to order theories in a narrative ordering that considers semantic dependencies (i.e., a narrative ordering based on a hybrid traversal) and their constituents in a context ordering. A combination of narrative ordering and context ordering produces her preferred document (Figure 31).

9.4. Summary & Evaluation of the Reordering

This chapter discussed the adaptation on one of three document layers, the structure layer. The proposed approach does not change the discourse structure of a document but solely permutes the constituents of document parts according to the user's structure preferences. Technically, the structure adaptation consists of two steps. In a first step, the document is abstracted, specific document parts are marked as sortable, which initialises that their constituents are rearranged. In a second step, these constituents are reordered according to one of three ordering strategies: the context ordering, the semantic ordering, and the narrative ordering.

9. Reordering of Mathematical Documents

The context ordering only provides a primitive solution and should be used as fallback. The semantic ordering arranges the infom in a document according to semantic transitions and might lead to inconsistencies due to transitional phrases in the document. The narrative ordering considers the narrative transitions between the infoms in a document and leads to the most coherent output. Optionally, it can consider semantic dependencies to implement a hybrid ordering of the document parts.

The main contribution of this work is its applicability to narrative documents. As mentioned before, most content planning approaches (in particular in eLearning) focus on topic-oriented documents. The following section compares two systems, ACTIVEMATH and WELSA, with the herein proposed ordering approach.

9.4.1. Comparison with ACTIVEMATH and WELSA

One could argue that the context ordering (e.g., the ordering of examples and definitions) is analogous to the approach in WELSA. In WELSA, the relations *requires* and *isRequiredBy* can be used to prevent the adaptation of learning objects: If two learning objects are related in either relation, the system does not change their order. However, if authors thoroughly represent the mathematical structure, many paragraphs in the text will depend on others, e.g., all examples in a theory might depend on a specific definition. In WELSA such relations are usually not provided by the author, thus allowing the system to reorder them. Given more structured material, WELSA would not be able to do any adaptation. This work improves the WELSA approach by separating the mathematical structure from the *narrative flow* and *discourse structure* of documents.

The adaptations in ACTIVEMATH are solely based on didactic relations, mathematical prerequisites are neglected (although they are available in the underlying representation format). The sequencing of learning object is either randomised or adapts to the didactic prerequisites. The author believes that there is high potential in considering the mathematical structure of a text as it allows us to produce consistent and mathematical sound results.

It should be mentioned that although given an extensive markup of mathematical structures (theories) and the discourse structure, material with many transitional phrases (such as the GENCS example) is less reusable and adaptable than independent, self-contained modules (or learning objects). ACTIVEMATH and WELSA impose constraints on their content: all transitional texts have to be omitted. Consequently, the system's learning objects can be combined arbitrarily and no restrictions have to be marked by the authors. To restore the coherence of the adapted documents, ACTIVEMATH generates *bridging texts* from templates. In contrast, the herein proposed approach supports authors to provide a variety of transitions between their document parts, from which an appropriate alternative is selected and an appropriate transitional text is displayed (if provided). So far, the herein described approach does not generate transitional phrases automatically, e.g., if transitions have not been associated with a text. An integration of the template-based generation by ACTIVEMATH could possibly provide a default for missing transitional phrases.

9.4.2. Proof-of-Concept

The reordering has been implemented in the *adaptor* library [ada09], which integrates JOMDOC [JOM08a] for the handling of OMDOC and for drawing on functionality of the substitution processes (e.g., the infom collector and context ordering). It implements and evaluates the ordering strategies that will eventually be integrated into JOMDOC as soon as the required extensions of the OMDOC format are approved. The JOMDOC and adaptor library have been integrated in the *panta rhei* system (Section 10.1).

9.4.3. Education Case Study

[Koh06, Section 8.3] emphasise that straw-man theories are an important didactic mean. When introducing a new concept, lecturers often first introduce a naive reduced approximation N of the real theory F , only to show that an example E_N is insufficient. Then, a first approach, a straw-man theory S , is proposed and an example E_S shows that it is insufficient. Based from the information of the failed attempt the theory F is introduced and an example E_F illustrates that it works. We can find such straw-man solutions in the GENCS material. Can we handle these straw-man theories during the reordering or does the adaptor destroy the intended flow?

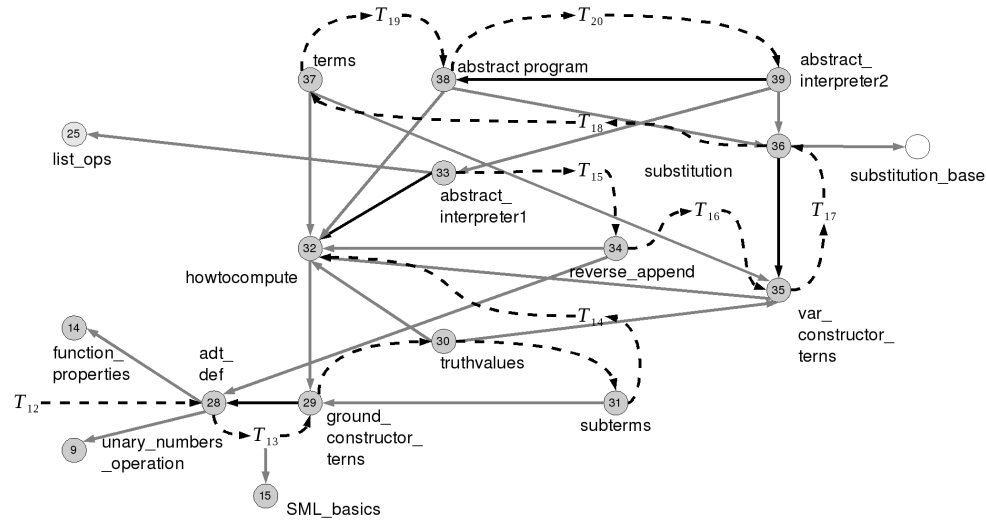


Figure 52.: Straw-man theories as didactic mean

Figure 52 shows the path through another part of the GENCS theory graph. It connects to the excerpt in Figure 33 via the theories `unary_numbers_operation` (9), `function_properties` (14), `SML_basics` (15), and `list_ops` (25). The `abstract_interpreter2` theory (39) is the target theory. We start with `abstract_interpreter1` (33) and provide the example `reverse_append` (34) that shows that it doesn't work. We extend the first attempt stepwise (with the `substitution` (36), `terms` (37), or `abstract_program` (38) theories) until we receive the complete mathematical foundations for the abstract interpreter. Below we summarise the transitional phrases in the material and provide the transition rules that induces the path. T_{12} defines all theories on the path, the particular order of the theories is defined by the subsequent transitions. For example, the sequence 29, 30, 31 is defined by T_{13} , the sequence 32, 33, 34, 35, 36, 37, 38, 39 is defined by T_{14} .

- T_{12} : “What’s next? We will now develop a theory of the expressions we write down in functional programming languages.”
Transition rule: $set(28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39) \rightarrow set(26, 27)$
- T_{13} : “With this definition, we now have a mathematical object for (sequences of) data type declarations in SML language. This is not very useful in itself, but serves as a basis for studying what expressions we can write down at any given moment in SML

9. Reordering of Mathematical Documents

language. We will cast this in the notion of constructor terms that we will develop in stages next.”

Transition rule: $sequence(29, 30, 31) \rightarrow 28$

- T_{14} : “Now that we have established how to represent data, we will develop a theory of programs, which will consist of directed equations in this case. We will do this as theories often are developed; we start off with a very first theory will not meet the expectations, but the test will reveal how we have to extend the theory. We will iterate this procedure of theorizing, testing, and theory adapting as often as is needed to arrive at a successful theory.”

Transition rule: $sequence(32, 33, 34, 35, 36, 37, 38, 39) \rightarrow set(28, 29, 30, 31)$

- T_{15} : “Let us now see how this works in an extended example; we use the abstract data type of lists from example 4.32 (only that we abbreviate unary natural numbers).”

Transition rule: $34 \rightarrow set(32, 33)$

- T_{16} : “Now let’s get back to theory, unfortunately we do not have the means to write down rules: they contain variables, which are not allowed in ground constructor rules. So what do we do in this situation, we just extend the definition of the expressions we are allowed to write down.”

Transition rule: $35 \rightarrow set(32, 33, 34)$

- T_{17} : “Now that we have extended our model of terms with variables, we will need to understand how to use them in computation. The main intuition is that variables stand for arbitrary terms (of the right sort). This intuition is modelled by the action of instantiating variables with terms, which in turn is the operation of applying a substitution to a term.”

Transition rule: $36 \rightarrow set(32, 33, 34, 35)$

- T_{18} : “Unfortunately, constructor terms are still not enough to write down rules, as rules also contain the symbols from the abstract procedures.”

Transition rule: $37 \rightarrow set(28, 29, 30, 31, 32, 33, 34, 35, 36)$

- T_{19} : “We we have to strengthen [...]. Furthermore, we have to get a grip on [...]. We formalize this notion with the concept of an abstract program, i.e., a sequence of abstract procedures over the underlying abstract data type that behave well with respect to the induced signatures.”

Transition rule: $38 \rightarrow set(28, 29, 30, 31, 32, 33, 34, 35, 36, 37)$

- T_{20} : “Now, we have all the prerequisites for the full definition of an abstract interpreter.”

Transition rule: $39 \rightarrow set(28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38)$

The transition rules explicate the informal cross-references in the text. For example, the prerequisite of the rule for T_{14} is extracted from the phrase “we have established to represent data”, which refers to the theories 28, 29, 30, 31. The consequence of T_{14} is derived from the outline of the straw-man solution in the following sections. It thus refers to all subsequent theories in the order they are included in the material. In contrast, the prerequisites of the rule for T_{15} are rather small and refer to the approximation of the target abstract interpreter theory. The consequences refers to the example that shows that this naive theory is not sufficient.

Given a markup of the above outlined transitions, the reordering algorithms can be applied to generate an ordering of the lecture notes that corresponds to the initial intention of the author. For future lectures, the ordering algorithm could support the lecturer to insert his content in arbitrary order and to generate an reordered preview on his material. Possibly, this could speed up the creation of his lecture notes. Unfortunately, the proposed algorithms has only been verified with a subset of the GENCS theories. The expected output corresponds

to the order in the lecture notes. The reordering of larger parts of the GENCS material was not yet been performed by the proof-of-concept prototype. A respective evaluation remains future work.

9.4.4. Chapter Summary

The novelty of this work is the representation of the discourse structure and narrative flow of a document in separation to its mathematical structure. The markup of transitions allows users to modularise their documents, while preserving coherence and consistency.

While the notation framework and substitution are well elaborated, the value of the re-ordering chapter lies in its initial algorithms and ideas that leave room for interesting further research issues. The proposed algorithms have only been evaluated on a small corpus of documents from the GENCS lecture and should thus first be applied to a large collection of documents. Discussion with experts from mathematics, in particular, logics, can help to carefully revise, improve, and extend the initial ordering strategies.

Nevertheless, the proposed approach provides the foundation for service like the planning of guided tours or the assembly of documents in general. In order to support these services, the restructuring of documents has to be addressed. Collaboration with the ACTIVEMATH project might lead to valuable insights.

Further research issues include the official integration of the proposed markup for transitions and transitional texts into OMDOC and the respective refactoring of the *adaptor* library into JOMDOC. Among others, one has to verify whether transitions rules should rather be represented in well-known and widely accepted formats like OPENMATH or RULEML [Rul].

9. *Reordering of Mathematical Documents*

Part IV.

Adaptation in Practice

10. The Panta Rhei System

Acknowledgement: *The prototypes presented in this part are collaborative work with Andrei Aior-dachioaie, Stefania Dumbrava, Josip Dzolong, Michael Kohlhase, Darko Makreshanski, Normen Müller, Alen Stojanov, and Jakob Ücker. Description and evaluations of these systems have been published in [MK09a, MK09b, Mü09a, MK08a, Mü08b, Mü08a, MK08c, WM07, MK07, Mü07b].*

The two words, *panta rhei* — *everything flows*, summarise the philosophy of Heraclitus, who argued that the world is a permanently becoming and vanishing one, in which everything constantly changes. This also applies to Web 2.0 technologies, which revolutionized the WWW and transformed it into a more social, user friendly, emergent, and flexible network. Users became media producers and web applications became more open and social, while at the same time improving their mutual integration. Thanks to their high usability in terms of content creation, software tools, such as web annotations and social bookmarking tools, have acquired a mass of user-specific information that users can share among each other. However, in order to find, explore, and track information, users have to be able to filter and structure web content. The technique of sharing and tagging bookmarks offers an interface for this. It allows systems to match bookmarks according to their topics and to improve searching. However, the user still has to dig through the tag clouds and has to filter relevant and useful information. Essentially, users lack the coherent, consistent, and well-researched structure that conventional media like books and courses provide. These put the bare facts into a *narrative* context that guides the reader, facilitates understanding, and avoids information duplication.

The *panta rhei* system is an interactive, collaborative document reader. Elaborated annotations of the content allow users to pose questions on-the-fly. By managing the users' annotation in a threaded structure, users can discuss materials and collaboratively answer questions. The system also offers a rating view on the content. This allows users to rate the relevance or helpfulness of documents and other users' entries. *panta rhei* furthermore offers rankings of documents and postings based on the user community's ratings and adapted views on the content. In consequence, *panta rhei* supports the social and emergent nature of collaborative knowledge acquisition and exchange but does also offer narrative structures to most easily explore and absorb information.

Before designing and implementing the *panta rhei* system, a requirements analysis was conducted, which resulted in the use case diagram in Figure 53.

1. Users can import documents (in OMDOC format) into the system,
2. Users can select/view/read documents,
3. Users can annotate documents and discuss them in a forum,
4. Users can rate documents and forum postings,
5. Users can adapt documents according to their preferences. In particular, users can
 - a) configure the mathematical notations in a document,
 - b) substitute paragraphs in a document,
 - c) reorder the parts of a document.

10. The Panta Rhei System

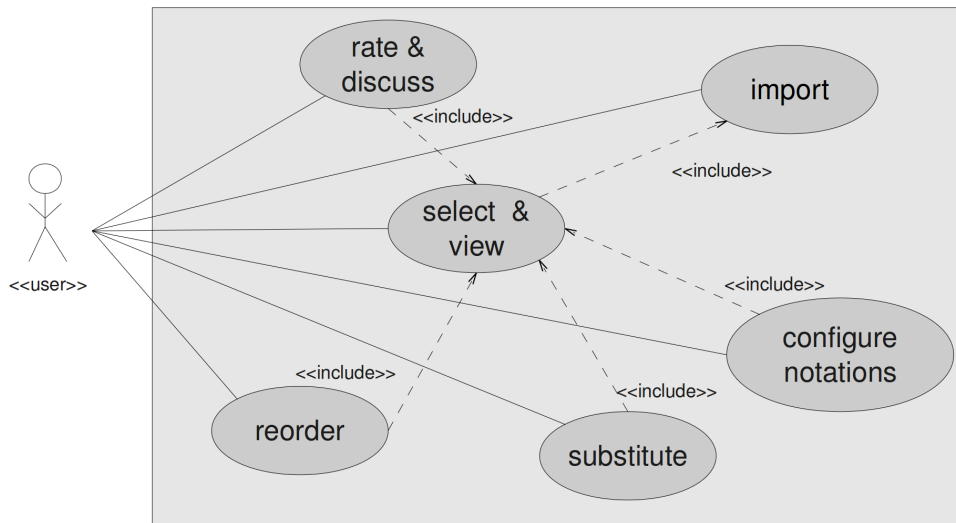


Figure 53.: Use cases for the *panta rhei* system

Figure 54 shows the architecture of the *panta rhei* system, which follows a model-view-controller (MVC) approach. It includes four main components: the user interface (the view), the adaptor (the controller), and the user profile and knowledge base (the model). The user interface supports users in browsing the presented documents and forum postings, to add forum postings and ratings, as well as to import new documents. The adaptor is responsible for the personalisation of any material in the system. The documents, postings, and ratings are stored in the system's knowledge base. The user profile includes tags and the user's karma as well as preference settings, which are used by the adaptor to configure the presented material.

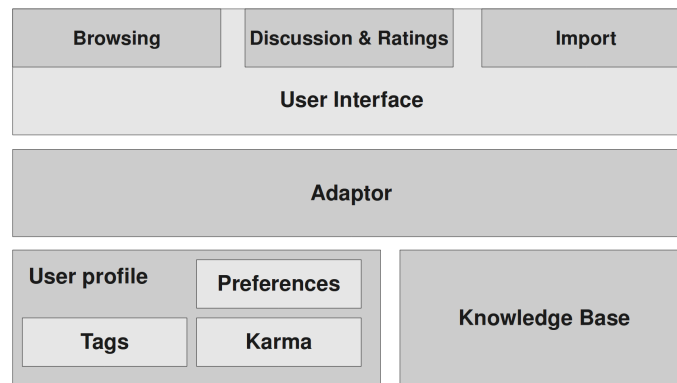


Figure 54.: The *panta rhei* architecture

The implementation of the *panta rhei* system was an iterative process that resulted in several prototypes, which each implement a part of the envisioned architecture. Section 10.1 presents the **adaptable prototype**, which focuses on the adaptor and implements the adaptation services described in Part II and III. Section 10.2 introduces the **social prototype**, which focuses on the document-centered discussions and ratings. It supports individual and community ratings and uses the latter to support a social ranking of documents and postings. Section 10.3 focuses on the user profile and outlines first ideas on extending *panta rhei* with user and group modelling techniques and services.

10.1. The Adaptable *Panta Rhei*

Acknowledgement: The prototype described in this section is collaborative work with Josip Dzolonga, who implemented the initial architecture. Special thanks go to Normen Müller for his advice.

The implementation of the adaptable *panta rhei* prototype focused on the adaptability of online material and supports users in configuring their documents interactively by setting a selection of extensional and intensional context options. The following sections introduce the adaptation features of the *panta rhei* system. After logging into the system, users can select from a list of documents that have been imported into the system. The *notation demo* document illustrates the adaptation of mathematical notations, the *exam generation* exemplifies the substitution of text paragraphs, and the *GENCS lecture* demonstrates the reordering.

10.1.1. Adjusting Mathematical Notations

The first document (the notation demo) demonstrates the adaptation of mathematical notations. The initial version of the document in Figure 55 has not yet been adapted to any context settings. All notations in the document are rendered with the system defaults (*SD* option, see Section 4.4.1). For example, $power(plus(1, x), n)$ presents $(1 + x)^n$, $binomial(n, k)$ denotes the binomial coefficient, and $imaginary$ the imaginary unit.

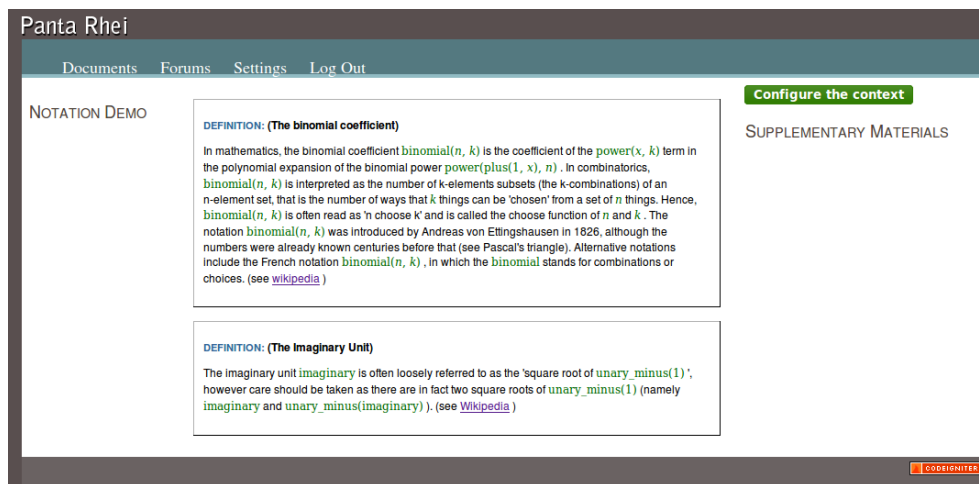


Figure 55.: The notation demo document.

Users can adapt these notations by configuring the context of their interaction. A click on `configure the context` opens the popup in Figure 56. Note that the demo currently only supports an extract of the extensible and intensional markup options from Part II and III but can be easily extended. We initialise the collection of notations from the document (*Doc* option) as well as the consideration of metadata (*IC* and *MD* option) and select *physics* for the area of application (*GC* option).

The adaptation of notations is based on the workflow described in Part II: The *Doc* option initialises the collection of notation definitions from the given document and the *IC* and *MD* options select an appropriate rendering based on the context annotations in the input document. The global context allows us to specify further context parameters, such as the area of application. The prioritization of the notation option is hard-coded in the system. For example, *IC* and *MD* have a higher priority than the *GC* option.

10. The Panta Rhei System

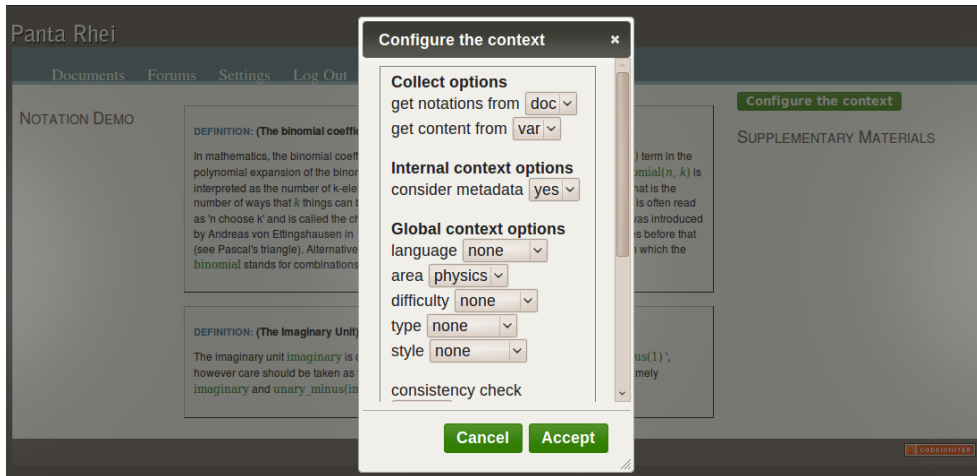


Figure 56.: The context configuration popup

We can see in Figure 57 that the notations of the notation demo document have been changed. For example, the binomial coefficient in the first line of the definition is displayed as $\binom{n}{k}$, while the binomial coefficient in the last line is presented as C_k^n . We have chosen `physics` and thus the imaginary unit is presented with j rather than i .

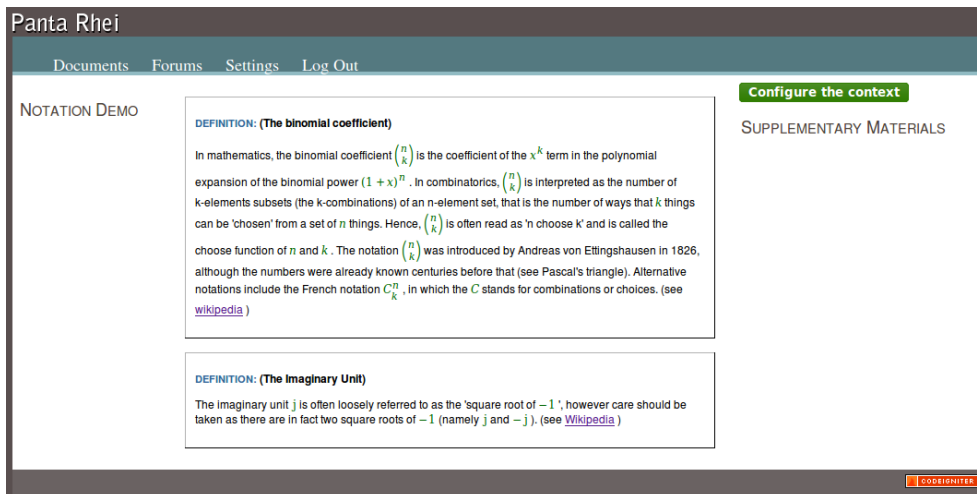


Figure 57.: The adapted notation demo document.

10.1.2. Substitution of Document Parts

The second document (the exam generation document) demonstrates the substitution of document parts. Figure 58 displays the demo document: all exercises are in English and have a high level of difficulty.

We can substitute the initial exercises by setting global context parameters, e.g. the language, area, or difficulty. In addition, we can click on an exercise and request all variants for this exercise. In Figure 58 three alternative exercises for the exercise algorithm for shortest path are displayed underneath the supplementary materials heading.

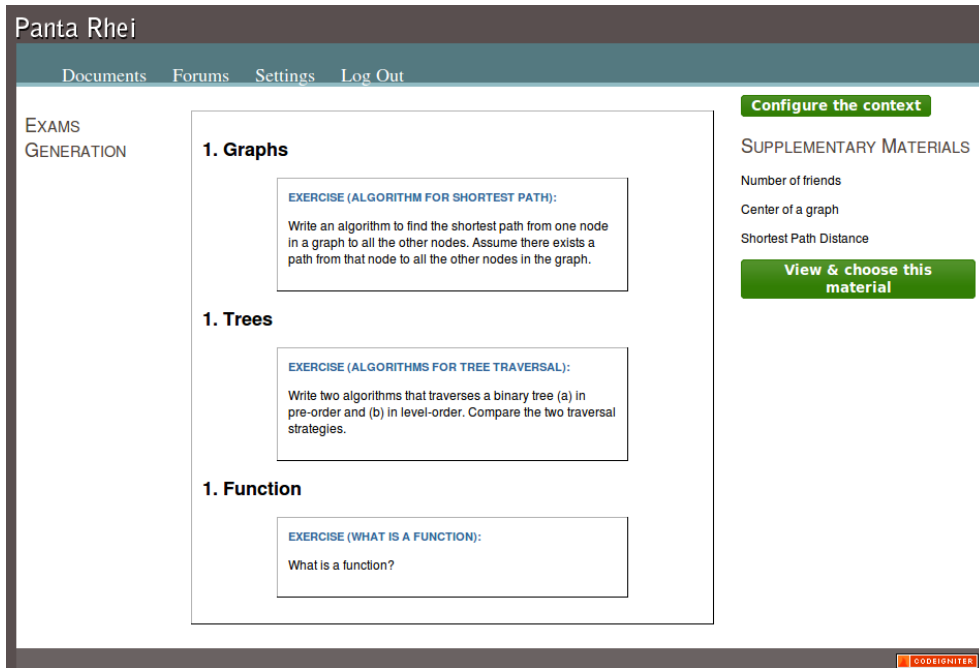


Figure 58.: The exam generation document.

A click on `view & choose this material` (Figure 58) opens the popup in Figure 59. The popup presents the content of the three alternative exercises. In the `choose` drop-down list, users can select, which of these additional exercises should replace the initial one in the document. We select `'center of a graph'`. The new exam is presented in Figure 60.

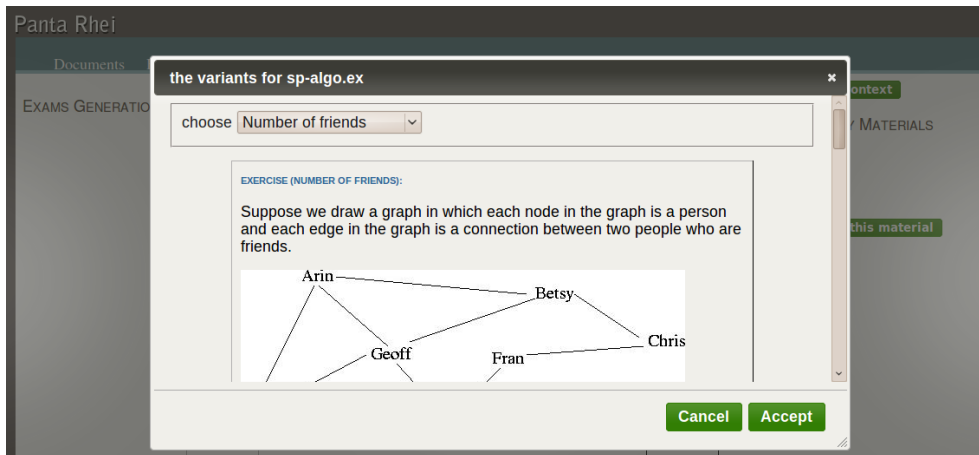


Figure 59.: View & select a variant exercise.

10. The Panta Rhei System

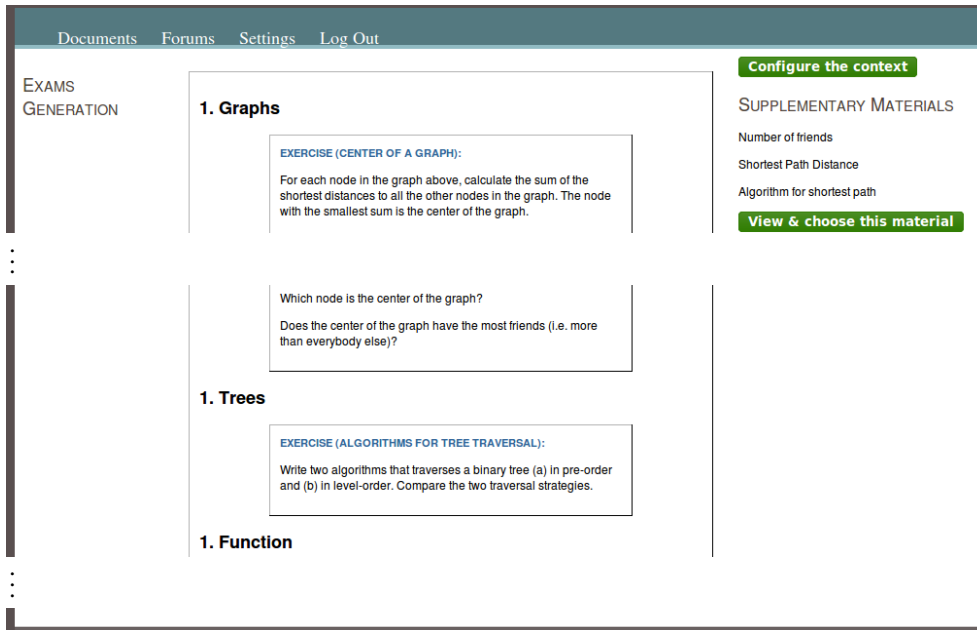


Figure 60.: The adapted exam generation document.

10.1.3. Reordering of Document Parts

The GENCS lecture notes demonstrate the reordering of document parts. The document contains two theories, algorithms and the spanning tree theory. The algorithms theory imports the spanning tree theory. It also includes a narrative transition to the spanning tree section. For the reordering, users can choose between the three ordering strategies: the *context ordering*, the *semantic ordering*, and the *narrative ordering*.

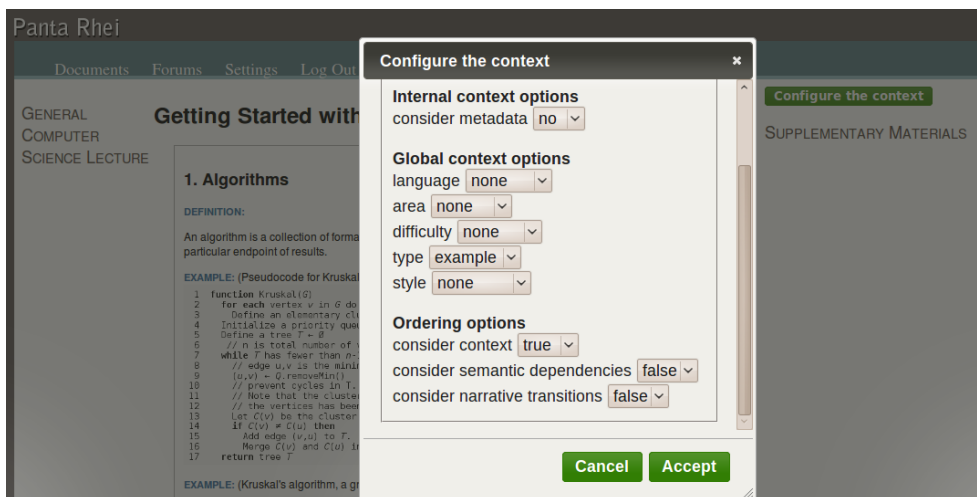


Figure 61.: Context configuration for the ordering of documents.

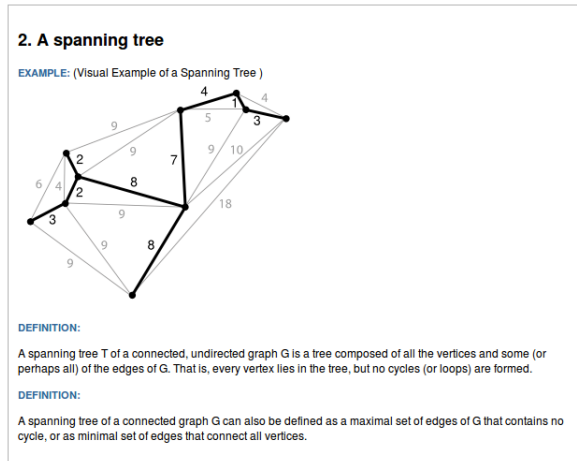


Figure 62.: Context ordering of the spanning tree theory.

Figure 61 presents the context configuration for the ordering of documents. Users can choose between the different ordering options: consider context, consider semantic dependencies, and consider narrative transitions. We set consider context true and select example as type. Figure 62 presents an extract of the adapted document: the constituents of the spanning tree theory are ordered according to how well their context annotations match the context configuration. The examples are preferred and placed first.

Panta Rhei

Documents Forums Settings Log Out

GENERAL
COMPUTER
SCIENCE
LECTURE

Getting Started with "General Computer Science"

Configure the context

SUPPLEMENTARY
MATERIALS

...

1. A spanning tree

DEFINITION:

A spanning tree T of a connected, undirected graph G is a tree composed of all the vertices and some (or perhaps all) of the edges of G . That is, every vertex lies in the tree, but no cycles (or loops) are formed.

DEFINITION:

...

...

2. Algorithms

DEFINITION:

An algorithm is a collection of formalised rules that can be understood and executed, and that lead to a particular endpoint of results.

EXAMPLE: (Pseudocode for Kruskal's algorithm)

```

1 function Kruskal(G)
2   for each vertex v in G do
3     Define an elementary cluster C(v) = {v}.
4   Initialize a priority queue Q to contain all edges in G, using the weights as keys.
5   Define a tree T = ∅. //T will ultimately contain the edges of the MST
6   // n is total number of vertices
7   while T has fewer than n-1 edges do
8     // edge u,v is the minimum weighted route from/to v
9     (u,v) = Q.removeMin()
10    // prevent cycles in T, add u,v only if T does not already contain a path between u and v.
11    // Note that the cluster contains more than one vertex only if an edge containing a pair of
12    // the vertices has been added to the tree.
13    Let C(v) be the cluster containing v, and let C(u) be the cluster containing u.
14    if C(v) ≠ C(u) then
15      Add edge (v,u) to T.
16      Merge C(v) and C(u) into one cluster, that is, union C(v) and C(u).
17  return tree T
                
```

EXAMPLE: (Kruskal's algorithm, a graph algorithm for spanning trees)

...

Figure 63.: Semantic ordering of the lecture notes.

10. The Panta Rhei System

Figure 63 shows the semantically ordered document. The spanning tree theory is inserted first as it is imported by the algorithms theory.

The screenshot shows the Panta Rhei system interface. The top navigation bar includes 'Documents', 'Forums', 'Settings', and 'Log Out'. A 'Configure the context' button is visible in the top right. The main content area is titled 'Getting Started with "General Computer Science"'. On the left, there is a sidebar with 'GENERAL COMPUTER SCIENCE LECTURE'. On the right, there is a 'SUPPLEMENTARY MATERIALS' section. The main content is divided into sections:

- 1. Algorithms**
 - DEFINITION:** An algorithm is a collection of formalised rules that can be understood and executed, and that lead to a particular endpoint of results.
 - EXAMPLE: (Pseudocode for Kruskal's algorithm)**

```
1 function Kruskal(G)
2   for each vertex v in G do
3     Define an elementary cluster C(v) = {v}.
4   Initialize a priority queue Q to contain all edges in G, using the weights as keys.
5   Define a tree T = ∅ // T will ultimately contain the edges of the MST
6   // n is total number of vertices
7   while T has fewer than n-1 edges do
8     // edge u,v is the minimum weighted route from/to v
9     (u,v) = Q.removeMin()
10    // prevent cycles in T, add u,v only if T does not already contain a path between u and v.
11    // Note that the cluster contains more than one vertex only if an edge containing a pair of
12    // the vertices has been added to the tree.
13    Let C(v) be the cluster containing v, and let C(u) be the cluster containing u.
14    If C(v) ≠ C(u) then
15      Add edge (v,u) to T.
16    Merge C(v) and C(u) into one cluster, that is, union C(v) and C(u).
17  return tree T
```
 - EXAMPLE: (Kruskal's algorithm, a graph algorithm for spanning trees)**

Randomly add a pair to the tree if it won't create a cycle. Repeat until a spanning tree has been created
 - EXAMPLE: (Visual Example of Kruskal's algorithm, a graph algorithm for spanning trees)**
- TRANSITION**

We have used the term "spanning trees" without defining it. Lets do that now.
- 2. A spanning tree**
 - DEFINITION:** A spanning tree T of a connected, undirected graph G is a tree composed of all the vertices and some (or perhaps all) of the edges of G. That is, every vertex lies in the tree, but no cycles (or loops) are formed.
 - DEFINITION:** A spanning tree of a connected graph G can also be defined as a maximal set of edges of G that contains no cycle, or as minimal set of edges that connect all vertices.
 - EXAMPLE: (Visual Example of a Spanning Tree)**

At the bottom, there are several warning messages:

- [0] WARNING: NO APPROPRIATE TRANSITION AVAILABLE. [Click here to insert a transition.](#)
- [1] WARNING: NO APPROPRIATE TRANSITION AVAILABLE. [Click here to insert a transition.](#)
- [2] WARNING: NO APPROPRIATE TRANSITION AVAILABLE. [Click here to insert a transition.](#)
- [3] WARNING: NO APPROPRIATE TRANSITION AVAILABLE. [Click here to insert a transition.](#)
- [4] WARNING: NO APPROPRIATE TRANSITION AVAILABLE. [Click here to insert a transition.](#)

Figure 64.: Narrative ordering of the lecture notes.

Figure 64 shows the result of the narrative ordering, where narrative transitions have been considered and semantic ones have been ignored. Since the algorithms theory includes a narrative transition to the spanning tree theory, it is placed first. The transition is

visualised with a transitional text. The footnotes in the document inform users about missing transitional texts. For example, an instructor might want to use this feature to improve the coherence of his lecture notes. An extension of *panta rhei* could support him to click on the warnings and to insert the missing transitions.

10.1.4. Implementation

The adaptable prototype of *panta rhei* was implemented following the MVC paradigm. It is composed of the user interface *panta* [pana] (the viewer) and the backend *janta* [jan09] (the model and controller).

panta is implemented in PHP using the CODEIGNITER [Cod] framework and makes use of Asynchronous JAVASCRIPT and XML (AJAX [Gar05]) techniques. It draws on *janta* for handling documents and user data. Forum discussions and annotations are stored in the frontend's MYSQL database.

janta is a RESTful web service that works a servlet using JERSEY [Jer] as reference implementation for the Java API for RESTful web services (*JAX-RS API* [JAX]). It uses HIBERNATE [Hib] as Object-Relational Mapping (ORM). *janta* provides an extensible content interface: Currently the system is using a MYSQL database to store all user data and the file system to maintain documents. It can easily be extended to integrate with a more sophisticated repository, such as TNTBASE [ZK09]. The specification of the *janta* API is available at <https://trac.kwarc.info/panta-rhei/wiki/janta>.

Focus of the *janta* project is the processing of configurable queries for documents, document parts, and personalised documents as well as authentication and permission handling. *janta* primarily handles content in the OMDOC format. It integrates the Java library JOMDOC [JOM08a] to convert OMDOC content into XHTML, to support the adaptation of mathematical notations, and to support the substitution of document parts. For the reordering services, *janta* integrates the *adaptor* library [ada09].

The *panta rhei* system has a unix-like authorisation management. Users can be associated with different user groups. Users and user groups can have read and write permissions to any addressable resource in the system. To verify whether a user has permission to read/write a resource, the following steps are performed by the *janta* component: The subroutine `get permission` in Listing 66 computes the permission of a user u for a resource r . It first checks if there is a record in the `UserPermission` table that specifies the permission between the user and the resource. If no entry can be found, the subroutine verifies whether there is a record in the `GroupPermission` table that associates a group of the user with the resource. The routine falls back on the default permissions of the resource (every resource has a default read and write permission).

Listing 66: Get permission for (u, r)

```

1  if exists record  $u, r \Leftrightarrow p$  in UserPermission table then
2    return permission  $p$ 
3  fi
4
5   $G$  = get all groups of user  $u$ 
6  forall  $g$  in  $G$  do
7    if exists record  $g, r \Leftrightarrow p$  in GroupPermission table then
8      return  $p$ 
9    fi
10 done
11
12 return permission of  $r$ 

```

10. The *Panta Rhei* System

The fine-grained permission handling is an important feature of the system. For example, tutors can be granted access to solutions of exercises, while students have no permission to read or write them.

10.1.5. Evaluation

The adaptable *panta rhei* prototype was evaluated based on interviews with students, lecturers, and researchers of the Jacobs University Bremen, the University of Bremen, the Technical University Ilmenau, and the DFKI Bremen. The interviewees were asked to solve three tasks (the adjustment of notations, the substitution of document parts, and the reordering of document parts) by following a given set of instructions. After each part, questions concerning usability, applicability, feature requests, and improvements were asked. In the following, the participants' feedback is summarised.

Adjusting Mathematical Notations

Acceptance and usefulness of the feature Many participants immediately confirmed that they would use the feature: *“This is exactly what I want”*.

Nevertheless, most participants emphasised that they would not trust the automatic adaptation of notations but are willing to try such a feature and to control the output. One participant stated: *“If that works okay (for at least 200 times) then I might start to trust the system and let it adapt my slides automatically”*. Another one said: *“I don't trust any system, but I would use the feature and then verify whether it is good. I'd play with the system and see what is possible.”*

One participant underlined that the consistent and correct rendering of notations is an essential service for semantically marked up mathematical documents: *“I want to benefit from semantic services but not if that reduces the quality of my document”*. In marked up mathematical documents, formulae and symbols are represented in OPENMATH (or Content-MATHML). This allows users to draw on semantic services, such as the verification of a proof by a proof assistance. However, to display these content-oriented representations of documents they have to be converted into a presentation format, such as XHTML+Presentation-MATHML: *“With this feature, users can guide this conversion and assure that all notations are correct and inconsistencies are omitted”*.

Application to other scenarios Most participants immediately referred to Wikipedia [wik] as use case for the notation feature: *“It is a place where information is shared and where the feature could help users to understand this information and, particularly, its background. The feature would be useful for any system where information is gathered collaboratively”*. Many participants also pointed to eLearning scenarios and suggested to apply the features to documents, such as slides, textbooks, and lecture notes. Also the reading of scientific eBooks was mentioned.

Context configurations Most improvements referred to the user interface. For example, having to scroll through the context menu was considered tedious. Some participants also had difficulties to get acquainted to the terminology, e.g., naming like `doc` and `consider metadata` caused confusions.

One participant pointed out that it was hard to trace the changes to the document and suggested to place the context menu next to the document. This would help users to reflect on their selection and to better understand how an adaptation was triggered by the various

context parameters. Such an enrichment of the user-computer interaction with meaningful explanation was referred to as *semantic interaction*.

Another participant suggested to allow users to define the content of the ‘configure the context’ popup declaratively. Such an extension would support users to define/formalise their own configuration options.

General improvements One participant suggested to add an upload of a user’s notation definitions: “*These should have the highest priority in the system and should be used as default.*” If the user’s private notation preferences should fail, the participant would refer to the author’s notations.

Some participants were missing an option to explicitly select notations, preferably, by right-clicking on a symbol or formulae and by choosing between a number of alternatives. The display of alternatives could then also be extended with an ‘add a new notation’ option, which would allow the user to extend his individual pool of notation definitions while reading a document. The explicit configuration should be memorised by the system and be applied to all other documents in the system.

To ease adaptation, one participant suggested to add a user profile in which general characteristics like nationality, background, or profession can be entered and from which a context configuration for the adaptation can be inferred. Alternatively, the system could track the user’s interaction and infer a context configuration.

Substitution of Document Parts

Acceptance and usefulness of the feature Participants involved in teaching considered the exams generation useful, while others had difficulty to relate to the example.

One participant was sceptical whether such a feature would be useful in an eLearning scenario: “*I would be overwhelmed if there are too many exercises. This is the same with traditional textbooks where chapters sometimes end with a list of exercises. I don’t know which one to choose.*” Instead, semantics on the exercises (a learning goal, the difficulty level, etc) would make it easier for users to choose an exercise and to understand what skill they train or what learning goal they achieve by solving the exercise.

Another participant emphasised that this work supports a *constructive learning approach*. Constructionism, an educational theory developed by Seymour Papert [HP91], argues that humans generate knowledge and meaning from their experiences. Instead of creating course material for the student, constructionism emphasises that learners should construct mental models to understand the world around them. Accordingly, learning can happen most effectively when students interact with the course material and can actively guide its adaptation. Constructivism also refers to the philosophy of the World Wide Web in which users browse and explore a network of highly interconnected resources instead of following predefined selections and navigations. Constructionism contradicts with the approach followed by learning environments like WELSA and ACTIVEMATH. These systems are based on the assumption that students prefer the recommendations of a system. For example, [Pop09c] claims that most students will follow a given outline for the course, whether this structure is well-thought or not. Future research could observe the different approaches and explore the extension of *panta rhei* towards a *constructive learning environment*.

Application to other scenarios One participant referred to eBooks: “*The 2000-page book ‘Thinking in Java’ [Eck98] includes many fundamental and advanced aspects of Java. Advanced programmers might wish to omit the elementary parts and focus on the hard chap-*

10. The Panta Rhei System

ters”. The substitution feature supports users to produce an instance of the book while browsing the original document or its table of contents.

Another participant pointed to promotions and advertisements of companies: “A company might wish to update a presentation once in a while. It should always be equivalent but include new, up-to-date information. A slide generator could support this”.

Context configurations Most improvements referred to the user interface. For example, the context values of the context dimensions should be thoroughly revised. One participant pointed out that difficulty `none` and `low` might actually result in the same selection and that `any` and `none` difficulty might be confusing. One participant would like to configure a context not only for the whole document but for each document part.

General improvements Most participants emphasised that it should be possible to adapt the document in a positive and a negative way, i.e., to remove chapters the user’s knows or to add examples he finds more appropriate: “Instead of substituting document parts, I’d rather have a feature that allows me to enrich the document, e.g., to add my own example if I think that the example in the book is too easy or not appropriate in some other way”.

Other participants want to easily restore the original content “The substitution should not change the original document. Instead, each user should only create his individualised view on the document”.

The interface for retrieving alternative exercises (or document parts in general) was criticised as being to tedious: “I don’t want to loose time finding the right example”. Instead of having to view the material in a popup, check-boxes next to the exercises can support users to more conveniently select a new exercise. One participant suggested to enter a browsing view for all parts with variants that allows users to conveniently view all alternatives by clicking on a `previous` and a `next` button. Another participant would add a combo-box above each exercise with a short description of each alternative exercise from which one can be selected and is then substituted with the original exercise. It was also suggested to separate the screen to display the document in one part and the exercise corpus, from which users can drag & drop the appropriate exercises into their documents, in the other part.

Reordering of Document Parts

Acceptance and usefulness of the feature The reordering feature raised several questions as most participants had difficulty to understand the impact of this feature. The terminology had to be explained and was revised iteratively during the interviews. One participant emphasised that if using such a complex system and its interesting features, he’d be – without question — willing to study a user manual or use a help button to study the configuration options and their consequences.

After providing an explanation, all participants raised interest in the markup of transitions and the opportunity to separate a document’s narrative flow from the semantic context of its content. However, since the markup efforts is increased, some participant are sceptical whether the user’s benefit of this feature can outweigh his authoring costs.

Most participants agreed that the ordering should only be presented as preview. They want to be in control over the document and only receive recommendations.

Application to other scenarios One participant would use the reordering feature to write his PhD thesis. “I just did such a refactoring. I changed the order of some sections since I realised that an earlier section actually depended on a later one. I would be willing to add metadata to each part and ask the system to recommend an order. It should only be an

recommendation! I don't want the system to enforce any changes on my documents. I want to decide."

Others consider the feature primary useful for eLearning: *"I would definitely not use it to write my Ph.D. thesis, but maybe as lecturer. In particular, if I am maintaining my slides and notes over several years and incrementally improve and extend them. Since I want to reuse them several times I'd be willing to invest additional efforts."*

Another participant said that he'd probably be willing to invest some extra time when writing an instruction manual for a mobile phone (or an Ikea product). The system should help to create the document and to reuse paragraphs for new products.

Context configurations Participants ask for a combination of all ordering options. One participant suggested to remove the `true/false` selection and to use check-boxes instead. The warnings (for missing transitions) were consider helpful.

General improvements Some participants complained that they could not easily trace changes of the example document. They believe that this could be improved by highlighting ednotes and transitions as well as presenting a short notification on what has been changed.

One participant suggested to allow users to drag & drop parts within the document. While pulling a part, the system should compute whether it is allowed at a specific position in a document or not. Such verifications could be visualised by highlighting or dimming parts of the document as well as by changing the icon of the mouse cursor. Another participant asked whether users can be supported to order documents according to a specific goal, e.g., a learning style or learning goal. These goals could be use to filter the transitions in the document. It was also suggested to display the alternative paths and to support modifications of a graph representation of the document.

Summary of the Interviews

Most participants stated that they do not immediately trust an automated adaptation of their documents but rather prefer a manual configuration that creates user-specific (pre)views but does not change a document's original structure, content, and presentation. For example, participants requested an upload of their own notation definitions and the reuse of the notation definitions of other users (e.g., the author of a document). Some participant want to explicitly select notations from alternatives and wish that these selection are memorised by the system. Other suggested to add a user model to automatically infer context configurations. A participant suggested to allow users to declaratively define the context configuration options. In all cases, participants underlined the need for a permanent storage of their preference and context settings.

For the substitution, participants outlined that one should be able to add and remove document parts as well as to substitute them. The preferences for adaptation modalities differ. One participant wishes to adapt a document while reading it, others suggested check-boxes, combo-boxes, previous/next buttons, and a drag & drop option to manually guide the substitution. Drag & drop was also outlined as option to manually order documents. Alternatively, a graph representation of the document was requested.

Participants, who would like to use the context configuration, suggested to specify context fine-grained for any part of a document. They also asked to remove the popup and to place the configuration view next to the document to allow users to trace the changes of the document more easily. To inform users about changes, colours and notification were also suggested. A display of metadata on document parts (e.g., the notations or exercises) was proposed in order

10. The *Panta Rhei* System

to enrich the user-computer interaction with semantics and to ease the selection of appropriate notations, examples, or exercises.

Some participants were convinced that the proposed features can improve eLearning materials and that they could possibly be applied to other documents like scientific eBooks, advertisements, and textbook as well as to systems like Wikipedia, in which content is gathered and authored collaboratively.

Open Research Issues

User interface design Previews of documents are already supported by the system. Actually, the system can only support views and does not change the original content, structure, and presentation of content. Possibly a more intuitive interface design could clarify that users are actually configuring individual views.

The explicit selection of notations from alternatives can be implemented by drawing on the substitution processes, which can already identify a list of alternative/variant document parts. The backend simply has to be extended with a respective API for this service. Most efforts have to be invested on the user interface side and can possibly be achieved in collaboration with the JOBAD project.

Adding modalities like check-boxes, combo-boxes, previous/next buttons, and a drag & drop option to manually guide the substitution as well as modalities to add document parts and to remove them from documents solely requires an extension of the user interface but not of the core algorithms. The same holds for the association of context configuration with fine-grained parts of a document. Such features are already supported by the backend, though not yet available in the frontend. The display of metadata as well as graph representation of documents also require an extension of the user interface. Since metadata and graph representations are already used for the core computation, only the backend's API has to be extended. Replacing the context configuration popup with an inline display is purely an issue on the user interface side. However, the devil is in the details and, thus, all revisions of the user interface and *janta* API should not be underestimated. Respective extensions should be addressed in collaboration with the SWIM and JOBAD project and by drawing on technologies like Javascript, AJAX, and Flash [Cen].

The importance of neatly presented documents and the visualisation of metadata — not only properties but also relations between document parts — is also expressed by the following quote: “A clear appearance of such components [structural components like chapters or sections and mathematical components like proofs and lemmas] as well as explicitly specified relations between such components enhance the readability of the document and makes the navigation of a text more enjoyable” [KMRW07]. Future work could focus on the semantic enrichment of the user-computer interaction and observe whether or not it can improve the understanding of the user and his acceptance of adaptation services.

Upload of notation definitions This feature requires an extension of the user interface as well as a modality that pools notation definitions together and preserves a pointer to these objects for the individual user, who initialised them. Section 10.3 discusses a respective extension of *panta rhei*.

User modelling Integrating automatic inferences of context configuration from a user model refers to a whole research area and requires extensive knowledge of appropriate projects and literature. First ideas on extending *panta rhei* towards an adaptive systems are discussed in Section 10.3.

Declarative context configuration Supporting users to declaratively define their context configuration is an interesting issue and can possibly build on the achievements of the *locutor* project. In *locutor*, users can declaratively specify a document model for their documents, which includes the dependencies, equivalences, and change management rules [Mül10]. Possibly, these declarative specifications can help to specify a syntax for context configurations.

In addition, users could be supported to formalise new context parameters. The formalisation of such metadata is addressed by the SWIM project, in particular, with the new metadata scheme for OMDOC [LK09]. An extension of the adaptation workflows could also provide users with a set of predefined properties, such as transitivity, symmetry, or partial order and allow them to reference them when defining their own context dimensions, values, and variant relations.

Consistency checks & change notifications The manual substitution and ordering of document parts can cause inconsistencies. For example, an exercise might depend on another exercise. Such inconsistencies can be avoided by preventing users to substitute dependent exercises. The respective core computations have been implemented and only have to be made available for the manual configuration services. Moreover, some participants have complained that changes to the document were not easily to detect and suggested to use colours and notifications to inform users of what happened. The *locutor* project has analysed the constituents of a change management process. Possibly, an integration with the project's core libraries [*loc*] can help to visualise changes and to fix inconsistencies: The author believes that if changes can be detected automatically and propagated, it should also be possible to highlight and explain them for the user.

10.2. The Social *Panta Rhei*

Acknowledgement: The illustrations in this section have been published in [MK07, Mül07b]. Special thanks go to Andrei Aiordachioaie, Stefania Dumbrava, Josip Džolonga, Darko Makreshanski, Alen Stojanov, and Jakob Ücker for improving the user interface and course content.

A spin-off product of the modularisation of narrative documents are *document-centered discussions*. Mathematical document formats, such as OMDOC, are particularly designed to represent fine-grained components of documents, such as proofs, definitions, or examples, and their dependencies. A prerequisite for the markup of relations and dependencies is that each part in the document can be uniquely addressed, e.g., using addressing schemes like `xml:id` [MVW05].

The fine-grained addressing of document parts builds the foundation for a collaborative markup as well as user modelling approach as mentioned in Part II and III. Properties and relations of document parts can be marked stand-off by specific annotations (Section 7.2.2). Applications can also use the unique identifiers to associate other information (ratings, tags, or forum postings) with the document parts. If this information is stored for each individual user, inferences can be made to derive context parameters for the adaptation of documents. Having modelled the individual users, they can be clustered according to specific preferences and behaviours into groups. Based on these group models, collaborative adaptation services can be provided. The social *panta rhei* prototype provides an initial infrastructure for such services. For example, it supports community ratings and uses the latter to rank search results. The experiments with the social prototype have encouraged the author to discuss the extension of *panta rhei* towards an adaptive system (Section 10.3).

10. The Panta Rhei System

Original motivation The social prototype was primarily designed to support the introductory computer science lecture (GENCS) at Jacobs university. The system augments the GENCS lecture with *online materials* and *online discussion facilities*. The former supports students to train missing skills in parallel to the lecture, while the latter encourages students to collaboratively solve problems with other students or to contact tutors to provide supplementary exercises and examples. An instance of the system is used as *precourse*, which allows students to study the lecture material before they arrive at Jacobs University and to identify prerequisites they might have to rework early.

The course materials The GENCS lecture notes are generated from a semantically enriched L^AT_EX source in the s_TE_X format [Koh08c]: Using L^AT_EX_{ML} [Mil], the s_TE_X sources are transformed into the OMDOC format [Koh06]. An XSLT transformation supports the conversion from OMDOC to XHTML, during which the XHTML is enriched by markers that provide CSS-based adaptive presentations of documents. The encoded narrative structure in the lecture material is extracted and is used to create a navigation menu for the online material. For the precourse, *panta rhei* offers an online authoring interface to allow tutors to create a course structure and to link examples, exercises, and multiple-choice questions to these sections.

The screenshot displays the 'panta rhei' web interface. At the top, there are navigation links: 'user manual | lecture | myPantaRhei' and a 'log out' button. The main content area is divided into three sections: 'Table of Contents' on the left, 'Lecture Notes' in the center, and 'Forum Postings' on the right. The 'Table of Contents' includes links for 'notes', 'assignments', 'preconditions', 'welcome', 'graphs', 'acyclicGraphs', 'cyclicGraphs', and 'trees'. The 'Lecture Notes' section is titled 'Graphs' and contains text explaining the importance of graphs and trees in computer science, followed by formal definitions for undirected and directed graphs. The 'Forum Postings' section is titled 'Forum Threads for graphs' and lists several threads with 'new' indicators. At the bottom, there is a 'myRating' section with checkboxes for 'relevance', 'soundness', and 'presentation', and an 'update' button. A footer at the very bottom contains copyright information: '© 2007 Copyrights KWARC | Panta Rhei project | by Christine Müller'.

Figure 65.: The panta rhei document view

Browsing the course For both scenarios (the course system and precourse) *panta rhei* supports the following student activities: the browsing of course material, the discussion of course material, as well as the search for postings and course material. Additionally, the system allows to rate forum postings (with values *helpful*, *correct*, *trustworthy*) and course pages (with values *relevance*, *soundness*, and *presentation*). In [KK06], these ratings are called **value judgements** and are defined as essential property in order to model user groups (Section 10.3). In the following, the functionality of the social *panta rhei* prototype is illustrated. Note that the first implementation focused on the core functionality of the system, while later developments have improved the user interface (Figures 71, 72, 73, 74, 75).

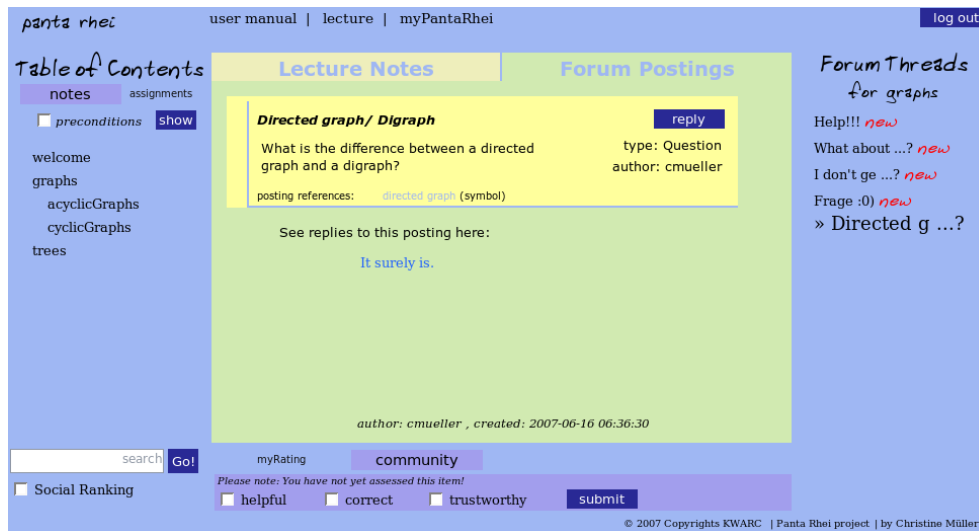


Figure 66.: The panta rhei forum

Figure 65 and Figure 66 present two screen-shots: The *table of content* on the left displays the sections and subsections of the GENCS lecture and its assignments. Depending on the selected tab, the main section in the center of *panta rhei* displays the hyper-media presentation — either a course page, an assignment, or a forum posting. The section on the right-hand side displays a list of *forum threads* that are linked to the course page and which *panta rhei* can rank and filter. Users can rate course pages and forum postings by using the rating form below the main section (Figure 68 and 69).

Addressing scheme The social *panta rhei* prototype implements a nested addressing scheme to associate postings and ratings fine-grained to the presented documents. The parts of the documents that can be annotated in this way are called **annotation item** (anits). An anit is described by a name (e.g. *directed graph*), its category (e.g. *symbol*), an author (e.g., *cmueller*), and a unique id. Ids are represented using the fragment identifier syntax of DITA (Section 2.1.1). Accordingly, an id is the concatenation of the element's unique `xml:id` [MVW05] and the ids of its XML parents. For example, the id `lecture/graphs/def2/directedGraph1` is formed by the document `lecture`, the section `graphs`, the paragraph `def2`, and the symbol `directedGraph1`.

The metadata (name, type, id, and author) of the anits is extracted during the conversion from OMDOC to XHTML and is stored in the system's knowledge base. The XHTML representation of each anit is enriched with code snippets. These are visualised as specific locator button (called *post-its*), which (on mouse click) initialises the pop-up of a posting or rating of the anit.

Discussion of course materials To post a question or comment for an anit (e.g. document parts like concepts, definitions, or examples), students have to click on a *post-it*. A popup is presented, which displays the author's name, the type of posting (question, comment, answer, etc), as well as the metadata of the anit. After submitting the annotation, *panta rhei* creates a forum posting that points to the respective anit. Figure 67 illustrates a question about the concept of a `directed graph` on the `graphs` course page. When clicking on the `Post` button, a forum posting is created.

10. The *Panta Rhei* System

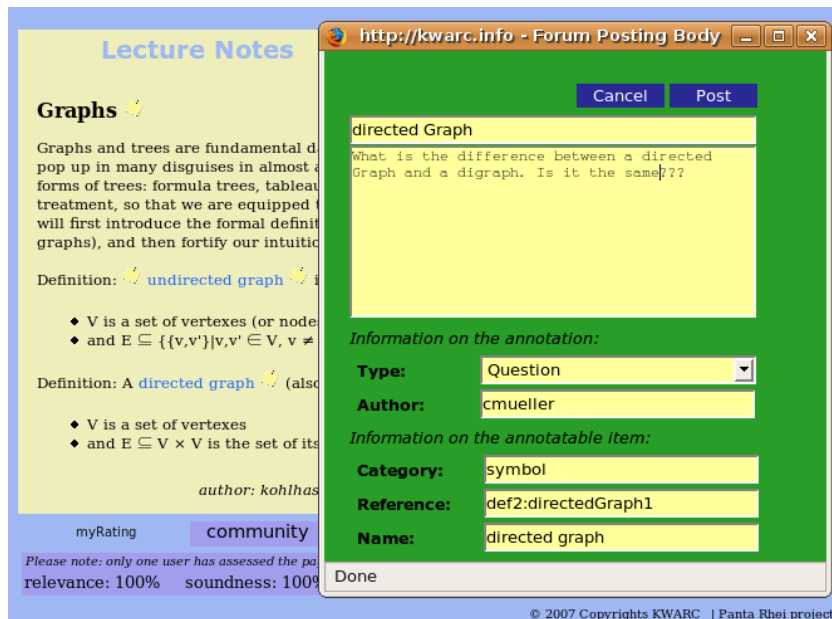


Figure 67.: Annotating document parts.

Rating content and forum postings Figure 68 displays a user’s rating for the course content, which indicates his view on the relevance, soundness, and presentation of the respective page. In addition, users may view a community rating, i.e., an average rating of all users as illustrated by Figure 69.

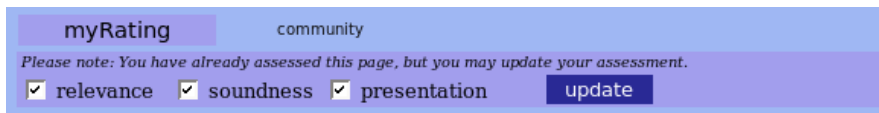


Figure 68.: User rating

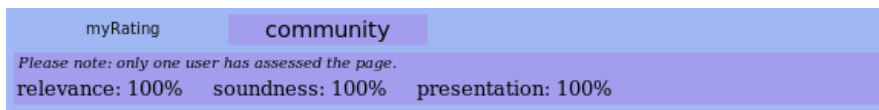
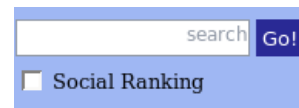
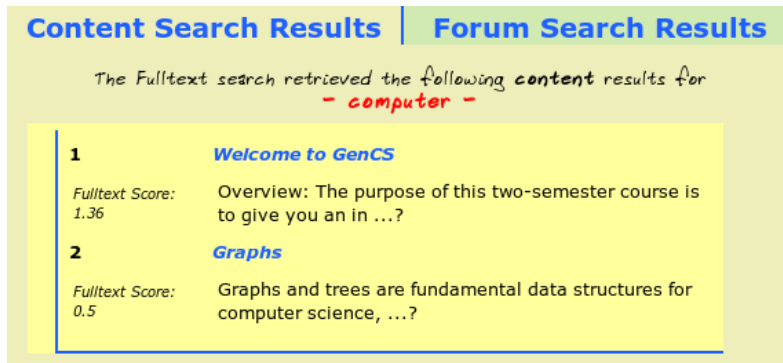


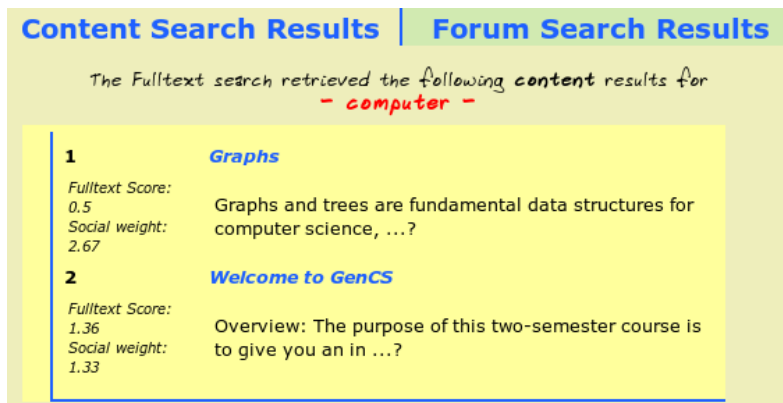
Figure 69.: Community rating

Searching document content *panta rhei* implements a *fulltext search* as well as a *social ranking* of search results based on the overall rating of pages. The figure below displays the results for the full text search for the query string `computer`. The similarity measure of the search is displayed on the left of the results. The `welcome` page is displayed first.





When applying a social ranking, the page graphs is displayed first (see figure below). It has received more positive ratings from the *panta rhei* community then the welcome page.



Searching for questions Based on the fine-grained addressing scheme, users can query for all postings of a document, a section, a paragraph, or a specific symbol in a document. These queries are implemented by links on the page: when users select a respective link (e.g., the directed graph link), *panta rhei* returns a list of threads that are linked to the respective anit and which are ordered by their creation timestamp (see left image in Figure 70).

Users can also set the display preferences of the posting list. They can choose to highlight all postings before a certain timestamp (yesterday, 3 days ago, a week ago, semester start, etc) or to hide all postings after the timestamp. For example in the right image in Figure 70, the list of postings has been filter: Only postings posted within the last three days are displayed. All postings that have been created since the user’s last login are additionally marked as new.



Figure 70.: List of postings

Alternatively, users can initiated that the *karma* of the postings’ author is considered to rank and filter postings. The extension of user profiles with user karma was inspired by the SLASHDOT website [sla]. SLASHDOT is a web portal for sharing technology-related information, so called ‘nerdy’ news. The system implements a collaborative reviewing approach

10. The Panta Rhei System

to assess the quality of user-submitted news and comments. The user karma expresses the reputation of a user and is used to rank the comments in the system: news entries, which authors have a higher karma, receive a highest priority. The karma of a user is computed from his ratings for comments and other users' ratings of his comments. The application of the SLASHDOT approach to *panta rhei* is collaborative work with Andrei Aiordachioaie, who specified and implemented the respective extension of *panta rhei* during his Bachelor Thesis [Aio08] at Jacobs University

The screenshot shows the 'panta rhei' interface with a slide titled 'Assignment: Boolean Algebra'. The slide content includes:

- Header: *panta rhei* alpha, everything flows, nothing stands still
- Navigation: slides, assignment, lecture, documentation, profile, news, forum, community
- User: Logout, cmueller
- Slide Title: Assignment: Boolean Algebra pdf
- Problem 1 (Parsing boolean expressions)
- Text: Given the following SML data types for Boolean formulae


```
datatype boolexp = bez | beo ("0 and 1")
  | bep of boolexp * boolexp ("plus")
  | bet of boolexp * boolexp ("times")
  | bec of boolexp ("complement")
  | bev of int ("variables")
datatype mybool = mytrue | myfalse
```
- Text: Write an SML function `beparse : string -> boolexp` that takes a string as input and transforms it into an `boolexp` representation of this formula, if it is in `Eqval` and raises an exception if not.
- Note: As there is no ASCII representation for the complement operation we used in the definition in class, we use `-x` for the complement of `x` in the input syntax. So the relevant clause in the definition is now:


```
- E11 -> (x, -(x)), (x + b), (x * b) | x, b ∈ E10
      bool
```
- Hint: For this you will need to write a couple of auxiliary functions, e.g. to convert lists of characters into integers and strings. A main function will have to look at all the characters in turn and decide what to do next.

Figure 71.: The panta rhei slide interface.

The screenshot shows a forum thread titled 'Thread: Assignment 2'. The thread content includes:

- Thread Title: Thread: Assignment 2
- Forum Home > Thread: Assignment 2
- Mark this thread as important
- Assignment 2 (Score: 25) by Teodora Chitiboi (tchitiboi) (Karma: 21) posted 7 days 12 hours ago
- Text: You can find the second assignment [here](#).
- Reply to This
- Re: Assignment 2 (Score: 20) by Philipp Weiß (pweiB) (Karma: 20) posted 7 days 11 hours ago
- Text: Just a question on problem 2.7 (Inductive Functions): Is there a typo at the last equation, shouldn't it be $g(n, s(m)) = s(sf(n, m))$?
- Reply to This
- Darko Makreshanski (TA) : Re(2): Assignment 2 (Score: 23) : yes, that's a typo, it should be 'n'
- Adrian Cirstea (student) : Re: Assignment 2 (Score: 20) : problem 2.7, first thing: f and g are the same for the two
- Alin Iacob (TA) : Re(2): Assignment 2 (Score: 23) : yes; it's a set of 6 axioms, defining the two functions
- Adrian Cirstea (student) : Re: Assignment 2 (Score: 20) : problem 2.7, first thing: f and g are the same for the two
- Teodora Chitiboi (TA) : Re(2): Assignment 2 (Score: 21) : In problem 2.7 the functions f and g are given by an
- Shishir Gautam (student) : Re: Assignment 2 (Score: 20) : Is question no. 2.4 correct?
- Alin Iacob (TA) : Re(2): Assignment 2 (Score: 23) : yes, the last formula has an equality sign, not equivalence like in
- Re(3): Assignment 2 (Score: 20) by Qiwei Sun (qsun) (Karma: 20) posted 2 days 12 hours ago
- Text: Dear TA... Do we need to use math talk for p2.2 to proof it's wrong?
- Reply to This
- Alin Iacob (TA) : Re(4): Assignment 2 (Score: 23) : No, it's fine to explain in normal (but precise) words. The
- Darko Makreshanski (TA) : Re(4): Assignment 2 (Score: 23) : You don't have to use math talk

Figure 72.: The panta rhei forum.

A never ending story The development of the *panta rhei* prototype included several iterations during which the interfaces and performance of the system was improved and its functionality was adopted. Figure 71 shows the new interface for the lecture material, Figure 72 and 73 the new forum interface.

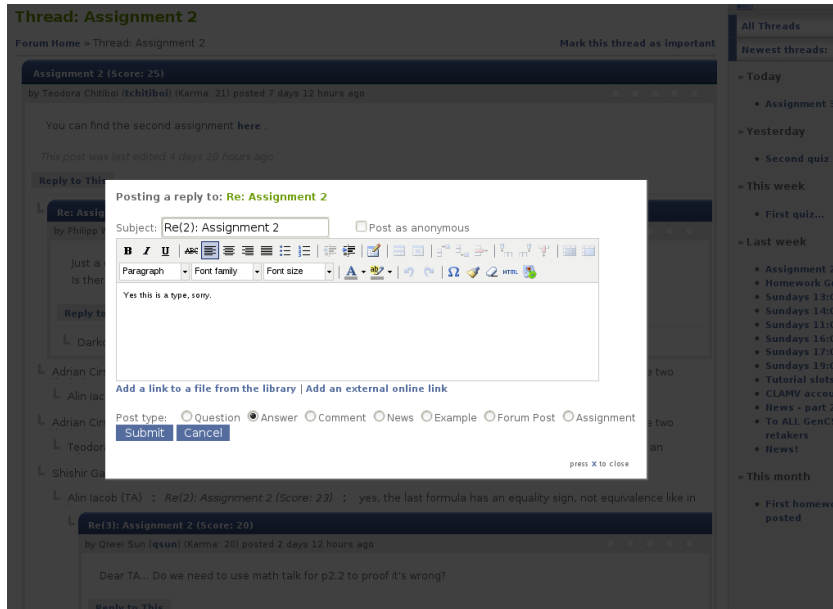


Figure 73.: The panta rhei forum.

Figure 74 illustrates the alternative annotation interface that was added to the system. In addition to the post-its, students can simply highlight a paragraph of text. The selection initiates the display of a discuss and rate button. A click on the buttons opens up a popup in which students can insert their questions or ratings. To support fine-grained ratings by selection, the original rating footer was replaced by the rating tableaux in Figure 75.

An **annotation** is an addition made to **information** in a book, **document**, online record, **Youtube** video, or other information. Commonly this is used, for example, in **draft documents**, where another reader has written notes about the quality of a document at a certain point, "**in the margin**", or perhaps just underlined or highlighted passages. **Annotated bibliographies**, give descriptions about how each source is useful to an author in constructing a paper or argument. Creating **in the margin** comments, usually a few sentences long, establishes a summary for and expresses the relevance of each source prior to writing.

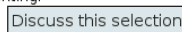


Figure 74.: Annotation by selection.

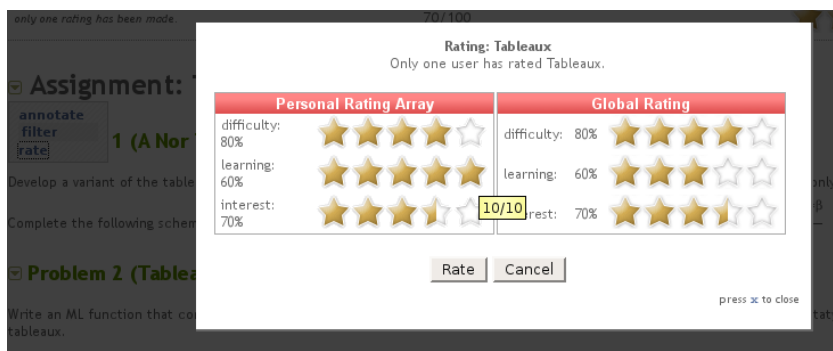
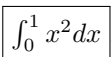


Figure 75.: Rating tableaux.

10.2.1. Implementation

The social *panta rhei* prototype [panb] was implemented in PHP and JAVASCRIPT. All content of the system, including the lecture notes, assignments, quizzes, forum postings, and user ratings are stored in a MYSQL database. Slides and assignments are imported from the file system. The thoughtful reader might wonder why the social *panta rhei* does not draw on the *panta-janta* infrastructure as used for the adaptable prototype (Section 10.1.4). The reason for this design decision is an historical and pragmatic one. The social *panta rhei* was the first system implemented by the author. Using solely PHP supported a rapid development of a first interface that was object of several discussions and feedback loops. Familiarising with the CODEIGNITER and JERSEY framework took considerably more time but led to a more scalable and stable infrastructure that can be used in further research projects.

To provide a convenient editing of mathematical content in postings and course materials, the HTML editor TINYMCE [Tin] has been integrated. A severe drawback of TINYMCE is that it cannot handle XHTML (and thus neither MATHML nor OPENMATH). Consequently, students and tutors can not conveniently enter mathematical formulae. In order to provide a math-friendly editor, several opportunities have been explored. In a first approach the ASCI-IMATHML.js [ASC] was integrated, a set of JAVASCRIPT files that convert a L^AT_EX-based syntax to MATHML. However, due to its limited coverage of mathematical notations and difficulties for students with Internet Explorer, this solution was omitted. Next the mathematical text rendering system MATHTRAN [Matd] was analysed, which uses JAVASCRIPT to contact a web service that converts T_EX embedded in the `src` attribute of an HTML `img` tag into an image. For example, `` is converted into . Images have their drawbacks as they can, e.g., not be arbitrarily scaled. Thus, the solution was also abandoned. The integration of the L^AT_EXML-based web services from the ArXMLiv project [ArX] was considerably tedious to implement and eventually an extension of TINYMCE was chosen, which was developed within the SWiM and SENTIDO project [LP08]. Unfortunately, the current release of this editor is still rather unstable and could thus not be successfully deployed with the system. Lacking a good math editor tremendously reduced the usability of the system, in particular, for the precourse case study. Tutors had to switch of the TINYMCE editor in order to paste XHTML with Presentation-MATHML since editing the XHTML content with the old TINYMCE editor removes all unknown tags, including, all mathematics in OPENMATH or MATHML.

Another technical issue referred to the addressing scheme used by *panta rhei*. Currently, the system requires unique identifiers in form of `xml:id` [MVW05]. Future work should invest time to integrate more advanced addressing schemas, such as the MMT syntax for OMDOC [Rab08, RK08] or the syntax used by the *locutor* system [loc].

10.2.2. Evaluation

An analysis of the student's activities was carried between September 1st 2007 to December 15th 2007. By November 10th 2007, 70 students were registered for the GENCS course: 68 students signed up for the system, from which 27 posted in the forum. From 297 postings, 136 were created by the students and 158 by the 8 teaching assistants and the professor. Moreover, 30 postings were referencing slides and assignments, i.e., they were created using *panta rhei*'s annotation feature. 46 postings were initialized in the *panta rhei* forum. 221 postings were replies. Students were also typing their postings with the available types: 13 advices, 51 answers, 14 changes, 34 comments, 1 example, 29 explanations, and 114 questions were

posted. 5 threads were posted in the scope `slides`, 45 threads for `assignment`, and 47 for `scope news`.

By the end of the case study, on December 15th 2007, all students had signed up for the system, from which 46 posted in the forum. From 508 overall-postings, 253 have been created by the students and 237 by the 8 teaching assistants and the professor. Moreover, 78 postings were referencing slides and assignments. 46 postings were initialized in the *panta rhei* forum. 384 postings were replies. Students were also typing their postings with the available types: 20 advices, 105 answers, 16 changes, 58 comments, 1 example, 36 explanations, and 212 questions were posted. 5 threads were posted in the scope `slides`, 82 threads for `assignment`, and 69 threads for `scope news`.

The increase of postings at the end of the semester can be explained with the increasing difficulty of assignments as well as questions concerning the midterm and final exam. The integration of the e-mail notification increased the acceptance of the system by the GENCS students. Looking at the statistics, students first primarily used the forum, but later also made use of the annotation feature. The rating feature of the slides was not used at all. Moreover, students mainly posted questions and answers. Examples were not posted. The higher number of questions is most likely a consequence of making `question` the default type. Students were more interested in discussing the assignments, only 5 forum threads were referencing the lecture notes. This can be explained by the rather poor quality of the automatically generated slides by the time of this evaluation. In consequence, although students prefer the online material (especially the menu and browsing), they are still referring to the PDFs to prepare for the exams. At the time of the first survey (and up-to-now), the system is only tested for Firefox2.0. However, a lot of first year students were using Internet Explorer, which caused a lot of frustration in the first weeks. Moreover, the first JAVASCRIPT plugins were not running properly on all platforms.

To conclude, due to technical difficulties not all features of the system could be used by the students. In informal discussions, students reported that they would prefer an online browsing of the slides over the PDF. They also stated that the precourse motivated them to study before the start of the semester and that the forum was very helpful for solving assignments and preparing for quizzes and exam. However, an in depth evaluation needs to follow after eliminating all technical flaws. In particular, it should be verified, whether the document-centered discussion in *panta rhei* can help to reduce discrepancies between students and to improve their learning experience.

10.3. The Adaptive Panta Rhei

Acknowledgement: This section presents collaborative work with Michael Kohlhase. Parts have been published in [MK09a, MK09b, Mül09a, MK08a, Mül08b, Mül08a, MK08c, WM07].

User-specific adaptation services require information on the user context, e.g., the user's preferences, backgrounds, skills, and environmental constraints. The adaptable *panta rhei* uses session variables to maintain the user's context configurations. These have to be reentered for each adaptation request. The social *panta rhei* stores user ratings and uses these to compute the rating for documents and postings by the whole user community. Neither prototype implements **profiles**, which build the basis for services like the explanation of differences between the notations in a document and the user's preferences/background (Section 5.1.1) or user-specific recommendations of content (Section 8.4.4).

Profiles relieve users from having to enter their context configuration for each request. Once captured in profiles, these configurations can also be shared among users, supporting in

10. The *Panta Rhei* System

particular newcomers in getting acquainted to the adaptation services. They also allow us to cluster users into *groups of shared practices*, where each group profile gathers user profiles with similar context preferences. Group profiles are essential for collaborative adaptation services.

This chapter presents the author's initial ideas on the modelling of user and group profiles and their exploration for adaptive services. Note that the modelling approach has not yet been implemented or evaluated. The following sections should thus be considered as proposal for further research.

10.3.1. Introduction into Communities of Practices

In the late 80s, [LW91] coined the term *Communities of Practice (CoP)* to denote social structures that have been around since the beginning of humankind: The concept refers to informal groups of people who share a concern or a task and learn to improve their skills by interacting regularly. Continuous *participation* allows the members of the CoP to develop **shared practices**, while *reification* of knowledge facilitates a more efficient exchange. Nowadays, the concept is a well-known and widely accepted theory with an impact on various disciplines: Meant to be useful for the debate on education, the concept has been applied to domains such as government, science, as well as industry and is of interest to both, researchers and practitioners.

Though mathematics is used as test tube, the author applies the theory of CoPs to the Science, Technology, Engineering, and Mathematics (STEM) disciplines. STEMicians are defined as mathematical practitioners, who understand mathematics as the basis for several disciplines¹. The STEM community is heterogeneous and joins individuals with different specialities and backgrounds [KW05]. Although outsiders may get the impression that mathematical practitioners form a homogeneous, unified community and share the same practice all over the world, they actually form various sub-communities that differ in their *preferred notations* (Part II), *basic assumptions* [Rab08], and *motivating examples* [KK06]. Deepening knowledge and learning takes place as individuals participate in various communities, while frequently “*switching the role of novice and expert depending on the current situation*” [KW05]. The author believes that CoPs provide a common context, which helps their members to cope with different community repertoires. As CoPs act as “*platforms for building a reputation*” [Wen05a], they also provide a notion of trustworthiness, relevance, and quality on which less-experienced members can build on.

The author has observed that STEMicians primarily interact via their artefacts including documents in a more traditional understanding such as publications, manuals, and textbooks as well as documents in a wider interpretation such as forum postings, ratings, and tags. It is assumed that interactions, and more generally *practices*, are inscribed into artefacts and that these can be extracted to model an individual's adaptation context as well as the contexts of communities of practices.

This work applies semantic XML technologies to reify (or markup) – i.e., to turn into representable objects – and capture the inscribed practices from documents. Based on the markup of practices, the design of a flexible management process for practices is addressed. For example, notation preferences of authors, aggregators, and readers are captured in user profiles and support adaptive notation services. In addition, semantic distances between user profiles are computed, which eventually support the clustering of users with shared practice into groups. Henceforth these groups are called **virtual communities of practices**.

¹Mathematics is often said to be ‘the language of science’. Galileo Galilei even went so far as to say “*Mathematics is the language in which God has written the universe.*” [Gal23].

10.3.2. Reifying Practices

Semantic technologies enhance informal information towards meaningful and machine-interpretable representations, which facilitate the reification of practices. In Part II, the author proposed to encode notational preferences declaratively in notation definitions. These notation definitions are embedded in semantically marked-up documents and can easily be extracted using XML technologies. She has thus proved that practices are inscribed into documents, that they can be turned into representable objects, and that they can be captured for adaptation services.

Future work has to address the reification of further practices and their exploration for user-specific services. For example, the markup of narrative transitions is a reification of an important practice in writing, the *narrative practice*. Another practice in mathematics are the *basic assumptions* that underlie an argument [Rab08]. These can be captured by processing the theory markup in OMDOC. A theory groups symbols, definitions, proofs, and examples and provides them with a common context. It can import other theories to include previously defined symbols. These imports encode pointers to concepts that underlie the theory and thus reify basic assumptions. *Value judgements* [KK06] denote another practice that expresses the relevance, trustworthiness, or quality of content. Ratings and annotations reify value judgements and allow us to capture them in profiles. Another mathematical practice is already reflected in user models of ACTIVE MATH and MATHDOX. Using XML markup (in form of the LEAMEL markup language [CCJS04]), these projects represent problem solution strategies and, thus, reified the *practice of solving problems* in mathematics.

The reification of notation practices is worked out best and is used to illustrate the further discussions. Future work has to evaluate whether the proposed modelling approach can also be applied to other practices.

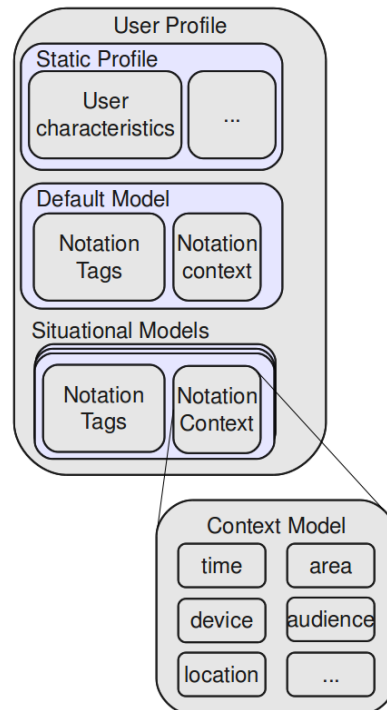
10.3.3. Modelling of User Profiles

Having reified notation practices, we need to design the user profiles that should maintain them. We can observe that notation preferences critically depend on the user's current situation. For example, an author might choose a *display output* of notations for his presentations but a *text output* for his textbook. A professor might try to use simpler notations in a lecture than in a talk for her research community. We should also consider a mobile scenario: Laptops or smart phones are used on the road. Thus, the adaptation of documents to different devices becomes an issue.

Having made these observations, we take on the work of [NWBKR08], who use static and situational user models, and adapt their approach to model notation practices.

Design of User Profiles

The figure to the right presents the constituents of user profiles in *panta rhei*: A *static profile*, a *default model*, and a set of *situational models*.



10. The *Panta Rhei* System

The **static profile** represents static user characteristics such as the age, languages, or nationality. The adaptation services consider these general information on the user as well as the specific contexts as specified in the *default model* and *situational models*.

The **default model** provides the general background of a user and is used as fallback for the adaptation if no concrete situational model applies. For the modelling of notation preferences, it includes a *notation context* representing the user's general notation background and *notation tags* representing the user's general notation behaviour.

Situational models represent the user's behaviour in specific situations like a lecture, a conference talk, or private studies at home. For the modelling of notation preferences, situation models consist of a *notation context* representing the user's current situation and *notation tags*, which reference notations that are used within the concrete situation.

Notation contexts consist of a set of context-parameters as defined in Part II and III. They are represented as key-value pairs. The key provides a context dimension such as date, time, location, device, task, area, audience, language, event, or layout. The value denotes a context value such as 2010-06-04, 10:10:00, Bremen, iphone, talk, mathematics, information systems, English, XY-conference, or display-output. Note that the *notation context* is not limited to these parameters but should follow an extensible approach.

Notation tags are annotated, weighted references to notation preferences (Section 4.4.1). The weights express how often and recent a user interacts with a specific notation. The status property expresses whether the referenced notation is used, preferred, disliked, known, etc².

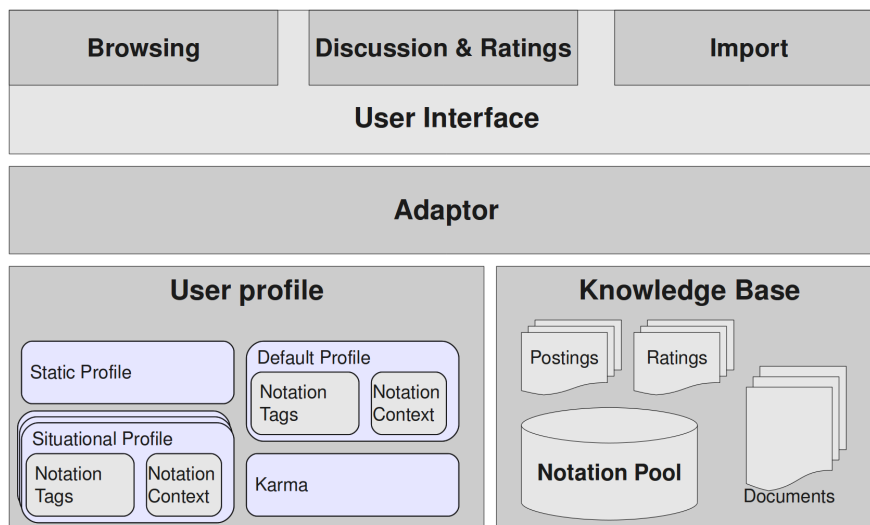


Figure 76.: The architecture of the adaptive panta rhei

Figure 76 shows the architecture of the adaptive *panta rhei* system. The user profile includes the static, default, and situational profiles. The knowledge base includes documents, postings, and a *pool of notation definitions*. These are collected from all documents as well as from user-specific notation definitions that have been imported into the system. The notation tags in the user profiles point to these notations definitions, in particular, to their rendering elements.

²Possibly FOAF [FOA] can be used/extended to formalise these values.

Initialisation of User Profiles

User profiles can store context configuration and relieve users from entering them for each interaction (as required in the adaptable *panta rhei* prototype, Section 10.1). They can also be initialised automatically by applying an implicit or explicit user modelling approach [BKN07].

The **implicit modelling** approach measures the user's interest and preferences by exploiting his activities in the system. Assumptions on the user behaviour are made, no extra efforts are required from the user. The author has observed two implicit techniques, web mining and tagging/rating behaviour analysis.

Web mining [Liu07] extracts usage patterns from log data to model certain aspects of the behaviour of users or communities. Analysing logs allows us to extract information about how often users interact with certain notations. For example, we can identify which notations are explicitly changed by the user and assume that this expresses her preference for these notations. Based on target hits (or page hits) we can observe the user's interaction with certain pages, which allow us to infer her interactions with the included notations. For example, we could assume that if a user accesses a page often, her familiarity with the included notations increases. The collected information can be added as notation tags to the user profiles. The `weight` of these tags expresses the frequency of the interaction, while the `status` helps to distinguish a change of notations (`used`) and the reading of notations (`known`).

Tags for specific document parts allow us to infer the user's familiarity with the notations of these parts. **Ratings** express the user's personal opinion on the quality of the page and thus can be interpreted as an explicit approval/disapproval of the included notations. Tags and ratings can be used to create notation tags, where the `weight` expresses the frequency and recentness of a tag/rating and the `status` expresses the user's opinion: `known` is added for tags, while `preferred` and `disliked` is added for ratings.

Though an implicit user modelling approach has not yet been implemented for *panta rhei*, the technical foundation is already provided. The JOMDOC library tracks all `rendering` elements that are applied to generate the user-specific notations. This information is persevered by adding `tag` elements and `ec` attributes to the output expressions and documents. Whenever a user tags or visits a document part, these enrichments are extracted to create notation tags.

The **explicit modelling** approach requires direct modification of the user profile by the respective user. The author proposes an import of semantically marked-up documents to allow users to initialise their user models. She has also explored questionnaires, in particular a notation selector, as one of many explicit modelling techniques.

The **import** for semantically marked-up documents is already provided in *panta rhei*. It is assumed that these documents include notation definitions that are used to display the mathematical expressions in the document. The import can be extended to allow users to initialise their user profiles with the notation tags that reference the rendering elements in a document, which are used during the conversion. Again we can draw on the JOMDOC library for tracking the respective renderings.

Questionnaires support users to explicitly describe their native language, location, or audience of a talk. Taking on the ideas from [SW06], the author has implemented a **notation selector** in *panta rhei*. Users can use the notation selector to select specific notations they know, like, use, or dislike. The users' selections are added as notation tags, context parameters like language or background are added to the notation context of the default or situational profile.

10. The Panta Rhei System

Notation Selector

1. Which Nationality are you?
 or other:

2. What were your majors in your previous education?
 Biology Chemistry Computer Science English
 History Mathematics Physics
 or other:

3. Please provide the mathematical area you are familiar with.
 algebra analysis computability geometry
 model theory proof theory set theory topology

⋮

7. Which notations below are you familiar with?
 Please rate the following mappings respectively: TRUE (selected) OR FALSE (not selected)

#	Concept	Notation	Rating	#	Concept	Notation	Rating
1	imaginary	j	<input type="checkbox"/>	2	sum	$\sum_{k=1}^n$	<input type="checkbox"/>
3	sum	$\sum_{k=1}^n$	<input type="checkbox"/>	4	imaginary	i	<input type="checkbox"/>
5	binomial coefficient	$\binom{n}{k}$	<input type="checkbox"/>	6	binomial coefficient	C_n^k	<input type="checkbox"/>

8. Please associate the following notations with a mathematical concept.

#	Notation	Concept	#	Notation	Concept
1	kgV	<input type="text" value="???"/>	2	$\binom{n}{k}$	<input type="text" value="???"/>
3	C_n^k	<input type="text" value="???"/>	4	i	<input type="text" value="???"/>
5	j	<input type="text" value="???"/>	6	lcm	<input type="text" value="???"/>

Figure 77.: Screenshot of the notation selector

Figure 77 presents a screenshot of the notation selector which presents the alternative notations for four mathematical objects: The *binomial coefficient* (four alternative notations), the *least common multiple* (a German and an English notation), the *sum* (notations for display and text output), and the *imaginary unit* (notations for mathematics and physics). By completing the following sections, users can explicitly initialize their user profiles.

- General description: In question 1 to 3, users are asked to provide general information on their nationality, previous education, and mathematical background. The user input is added as context parameters to the static profiles.
- Preferred notations (Question 4): *Which of the notations below do you prefer or like?* Concepts and their valid notations (concept-notation pairs) are provided. Users are asked to select all mappings that they prefer. Their answers are converted into notation tags annotated by a `status` attribute with value `likes`.

- Disliked notations (Question 5): *Which of the notations below do you dislike or reject?* Users are asked to select all concept-notation pairs that they dislike. A notation tag with status `dislikes` is created.
- Usage of notations (Question 6): *Which notations have you used before?* Users are asked to select all concept-notation pairs that they have been using. A notation tag with status `uses` is created.
- Knowledge about notations (Question 7): *Which notations below are you familiar with?* Users are asked to select all concept-notation pairs that they are *familiar* with. A notation tag with status `knows` is created.
- Testing concept-notation mappings (Question 8): *Please associate the following notations with a mathematical concept.* This task requires users to *associate* a given list of notations with one concept from a selection of available concepts. A notation tag with status `knows` or `!knows` is created.

Another alternative for initialising user profiles is to support users to share their profiles. In particular, newcomers could create an initial profile by borrowing the configuration of another user: Researchers could use the profiles of colleagues, students could use the profile of their instructors or friends.

10.3.4. Modelling Group Profiles

Once user profiles have been modelled, they can be used to segment users into groups of shared practices (called virtual CoPs). Here we can build on a statistical method from the field of data analysis, called *cluster analysis* [BSMG02]. Cluster analysis is used to partition an object sets (i.e., the user community) into clusters (i.e., groups), so that the data in each cluster share some common trait - often with proximity according to a distance measure. In particular, cluster analysis aims at aggregating objects (i.e., users) into partitions so that:

- The distance between the elements of the same partition is minimal,
- The distance between the elements of different partitions is maximal.

In the following section, an important prerequisite for the cluster analysis is addressed: the computation of *distances* between the objects. For illustration purpose, we base this computation on a specific characteristics of the objects: the notation practice.

From Practices to Distance Values

Figure 78 illustrates the notation practice of two users A and B ³. A 's notation definitions specify a notation for the binomial coefficient $\binom{n}{k}$, a notation for multiplication $*$, a notation for addition $+$, and a notation for the cross product \times . B 's notation definitions comprise a different notation for the binomial coefficient C_k^n , two notations for multiplication \times and $*$, no notation for addition and cross product, but a notation for subtraction $-$ and the Cartesian product \times .

To compute a distance between these two notation practices, we make use of the XML differencing of the *locutor* system [loc]. The XML diff takes as input two XML files and an equality model. It returns a semantic difference (aka *semantic diff*). The XML files present the notation definitions. The equality model supports a parametrisation of the differencing.

³See Listing 14 for the XML representation of notation definitions.

10. The Panta Rhei System

Notation preference user A	Binomial coefficient	Multiplication	Addition	Cross Product
	$\binom{n}{k}$	*	+	X
Notation preference user B	Binomial coefficient	Multiplication	Subtraction	Cartesian Product
	C_k^n	* x	-	X

Figure 78.: Notation preferences of user *A* and *B*

For example, we can specify that two notation definitions are equal, if their `prototype` elements and `rendering` elements are equal and, thus, ignore the context annotations of the rendering elements. Moreover, the equality model can ignore or define an order of elements. For example, the order of the `notation` elements in the XML files should be ignored but the order of their `prototype` and `rendering` elements should be considered.

A semantic diff consists of a set of actions, which define the activities that are required to update an XML document (e.g., *A*'s notation definitions) towards another document (e.g., *B*'s notation definitions). These include `remove`, `insert-after`, `insert-before`, and `append` statements. Listing 67 provides the *semantic diff* of the notation definition from *A* to *B*.

Listing 67: *Semantic Diff* based on notation practice

```
<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">
  <!--remove notation of A for binomial coefficient-->
  <xupdate:remove select="/notations/notation[1]/rendering" />
  <!--add notation from B for binomial coefficient-->
  <xupdate:insert-after select="/notations/notation[1]/prototype" >
    <xupdate:element name="rendering"> ...
  </xupdate:element>
  </xupdate:insert-after>
  <!--add notation from B for multiplication-->
  <xupdate:insert-before select="/notations/notation[2]/rendering" >
    <xupdate:element name="rendering"> ...
  </xupdate:element>
  </xupdate:insert-before>
  <!--remove addition-->
  <xupdate:remove select="/notations/notation[2]" />
  <!--remove cross product-->
  <xupdate:remove select="/notations/notation[3]" />
  <!--append subtraction and Cartesian product-->
  <xupdate:append select="/notations" child="last()">
    <xupdate:element name="notation">
      <xupdate:attribute name="name">subtraction</xupdate:attribute>
    </xupdate:element>
    <xupdate:element name="notation">
      <xupdate:attribute name="name">crossProd</xupdate:attribute>
    </xupdate:element>
  </xupdate:append>
</xupdate:modifications>
```

To use a semantic diff in a standardized clustering algorithm and to eventually compute clusters of similar users (or virtual CoPs), we need to map its actions to a numeric value. In a first step, the actions have to be interpreted. For example, a `remove-statement` can indicate that *A* knows a concept or notation that *B* is not aware of. Vice versa, and `append-statement` or `insert-statement` signifies that *A* is still missing certain background in order to understand *B*'s notation system. One could argue that the insertion of `prototype` elements is more severe than the appending of `rendering` elements. The former means that a users is missing the meaning and understanding of a mathematical object, while the latter only states that he is used to different notations but might be able to transfer his previous experiences to adjust to other notations. Alternatively, missing entries in the notation definitions of *A* do not necessarily mean that *A* doesn't know a concept or notations, but could also be interpreted as different interest or focus: *A* simply doesn't like or use a specific concept/notation but very well knows about it. The interpretation of semantic diff therefore depends on the users as well as his current context and needs careful considerations.

Given an acceptable interpretation of the actions in a semantic diff, we have to map them to numeric values and compute the overall distance from these values. For illustration, the formulae below is used to compute a similarity measure d , which considers the *number* of actions, the action *type*, and the action *depth*. The depth refers to the position of the XML elements/attributes referenced by the semantic diff. For example, an action on the root element of an XML tree is considered as more severe as an action on the leaves.

$$d = \sum_{k=1}^n \frac{w(\text{type}(a_k))}{\text{depth}(a_k)}$$

A distance value $d = 0$ signifies that the two notation definitions are equal. For our example, we compute a numeric difference based on $n = 6$ actions:

$$d = \frac{w(\text{rm})}{3} + \frac{w(i)}{3} + \frac{w(i)}{3} + \frac{w(\text{rm})}{2} + \frac{w(\text{rm})}{2} + \frac{w(\text{app})}{1}$$

Given that all actions are equally weighted with $w(a_k) = 1$, we result in a measure $d = 3$. Note that the previous computation is for illustrative purpose only and needs to be evaluated.

From Distances to Virtual CoPs

The author suggests to apply a *hierarchical classification approach*. Hierarchical classification approaches construct partitions based on a given set of objects. They can thus support two kinds of services: the modelling of a group from a fixed set of users or the exploration of group memberships across the whole user community. The former services supports users in registering a group, e.g., a research group or course, and to create a group profile from a fixed set of members. The latter supports users in finding users in the whole user community that have similar practices, e.g., similar notation preferences, background, or value judgements. Possibly these users can provide helpful references or are interested in collaborations.

Two hierarchical classification approaches can be distinguished: divisive and agglomerating approaches. The former start with a single class of all objects and refine the classification top-down, while the latter start with singleton classes and gradual coarsen the partitions. The selection of a concrete clustering approach remains for future work.

Alternatively to cluster analysis, group profiles can be constructed from scratch by manually specifying a context configuration for the group (e.g., the corporate identity of a company) or by importing the community's repertoire and extracting practices from their documents. The next section explores further services that can be provided based on user and group profiles.

10.3.5. Services based on Profiles

Given that all the above has been specified and implemented, we can support the previously outlined adaptive services. These include the upload of notation definitions into a central pool during which user profiles are initialised, and the explanation of differences between the displayed notations and the user's background/preferences. We can also think about extending the modelled practice to support user-specific recommendations.

Figure 79 illustrates the application of profiles to an eLearning scenario. They can be used to manage the dynamics of a course. Imagine a professor that creates a group profile for his course. The initial course profile is initialised from his profile (1) and includes tags of recommended readings, his ratings, and his notation preferences. All registered students can use the course profile to view an initial presentation of the course material. Their student profiles are initialised from the group profile (2). During the interaction of the students with the system, the student profiles are extended and revised (3). The course profile is also updated based on the common interaction of the students and the professor (4). These changes are used as feedback by the professor (5). For example, he can decide to speed up or slow down the pace of his course based on the student's ratings. Vice versa, changes to the professor's profile (6) influence the course (7) and inform the student's profiles (8).

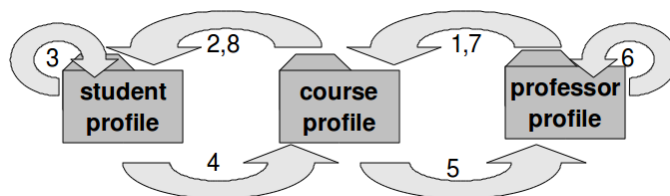


Figure 79.: Sharing user and group data.

If user and group profiles are made public, they can also be used to present the views of users and, thus, implement an extension of the **CONNECTIONS lenses** (Section 2.2.2). In **CONNECTIONS**, lenses are simply collection of documents that are approved or preferred by one or more users. Lenses in *panta rhei* adapt these documents according to the practices of one or more users. This is particularly helpful for novices, in particular, if profiles can be used to initialise a novice's profile.

10.3.6. Portability of User and Group Profiles

Since STEMicians use various tools to accomplish different tasks, their repertoire of artefacts is scattered across various system-internal database and so is the repertoire of their communities of practices. The author claims that we need to consolidate artefacts and integrate existing STEM tools to facilitate STEMicians to more easily manage and share their data across systems. Since the STEM community is very document-centered, the author proposes an integrative approach that pools STEM tools with various specialized functionality around a common corpus of artefacts in a common markup format, which then suit as a *semantic integration platform*. For the course of this work, the author focused on systems that build on the OMDOC format.

Several tools have already been integrated via the OMDOC format and support various activities. In particular, the OMDOC universe comprises the XML repository TNT-BASE [ZK09], the change management system *locutor* [loc], the semantic wiki SWIM (Section 2.2.4), the course management system **ACTIVEMATH** (Section 2.2.3), the automated prover **VERIFUN** [WS03], the semantic search engine **MATHWEBSEARCH** [Mate], a theory-

browser [NK07], a logic translator [Rab08], the invasive OMDOC editors in Microsoft PowerPoint and Word [Koh07], the web editor Sentido [Pal06], and the document reader *panta rhei*. If we assume that the markup formats in Section 2.1 can all be translated to OMDOC, the systems in Section 2.2 can also be considered.

Some of these systems already support adaptation (TNTBASE, SWIM, ACTIVEMATH, *panta rhei*, etc), the remaining systems (CPOINT, CWORD, CONNEXIONS, etc) can integrate the proposed adaptation workflows from Part II and III. To support the exchange of user and group files between these system, we need a stand-alone representation for these internal data structures. Such a representations are henceforth called **portfolios** and **copfolios**. The former integrates all ratings, postings, documents, preference setting, etc of an individual user, the latter includes a community's repertoire. As portfolios are special cases of copfolios (i.e., copfolios for singleton groups), the author only refers to the term 'copfolio'.

The author proposes to maintain copfolios centrally in an community of practice toolkit (COPIT). Each system has to provide an export to and import from COPIT in order to access these copfolios. As COPIT supports the sharing of user-specific data across systems and among other users, we need to take the authentication and rights management into account. The author suggests to use a decentralised identity service via URL-based identities (like OPENID [Ope]) to uniquely identify STEMicians across systems. Systems can adapt their authentication process to these open identities or rely on their own user management.

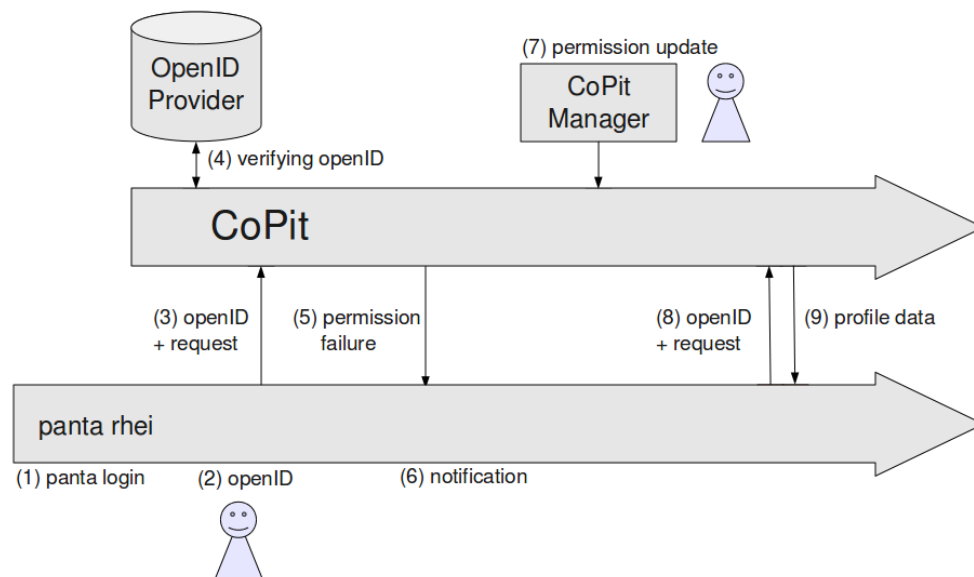


Figure 80.: Authentication and rights management based on OPENIDs

Figure 80 illustrates a potential OPENID-based scenario. The user `cmueller` logs into the *panta rhei* system (for the first time) using his *panta rhei* account (1). She accesses his profile page in the system, provides her OPENID, e.g., <http://cmueller.myopenid.com>, and assigns *panta rhei* to import her profile data from her copfolio (2). *panta rhei* calls COPIT, provides the OPENID, and request the profile data of `cmueller` (3). COPIT prompts an OPENID Provider (e.g., <http://myopenid.com>) to verify the OPENID (4). It then verifies the access rights for the *panta rhei* system on the `cmueller` copfolio. Since *panta rhei* has no access, the request fails (5). *panta rhei* informs the user to grant the required permissions (6). The user uses the COPIT manager to grant *panta rhei* read access on her profile and

10. The Panta Rhei System

preference data (7). *panta rhei* re-initializes its request (8). This time the system has access to the respective parts in the `cmueller` copfolio, receives the profile data, and initializes its internal profile (9). Since *panta rhei* provides email notifications, the system can interpret the subscription preferences in the user's copfolio. Being a notation aware system, *panta rhei* can also interpret the user's notation preferences.

10.3.7. Implementation & Evaluation

Unfortunately, the author was not able to implement and evaluate the proposed user and group modelling techniques and services. The above discussions should thus be considered as proposal for further research.

10.4. Chapter Summary

The *panta rhei* system has been an excellent *object-to-think-with* [Koh08a]. It allowed the author to verify and to extend the theoretical parts of her thesis but also to achieve and build new theories. Nevertheless, the *panta rhei* prototypes only implement parts of the envisioned system and need a lot of love and care before they can be extensively used in real scenarios.

The author suggest to explore synergies and potentials of future collaborations with projects such as ACTIVEMATH, WELSA, MATHDOX, TNTBASE, SWIM, and JOBAD.

11. Conclusion & Future Work

11.1. Conclusion

Modern web technologies have empowered users to create and share documents across the world. Today, users are confronted with an immense amount of documents, including documents in a more traditional understanding such as publications, manuals, and textbooks as well as documents in a wider interpretation such as forum postings, ratings, and tags. They struggle to find appropriate resources and to acquire the essential knowledge conveyed in these documents. Essentially, the web's usability depends on whether or not it can respond to the individual user context and personalise web documents respectively.

Personalising documents is widely addressed by applications in industry and research. For example, eLearning systems address the personalisation of online documents, particularly, the presentation and content planning of documents. They propose the most advanced adaptation services and have been thoroughly discussed in this thesis. Unfortunately, all of these systems apply a topic-oriented approach: They can only handle narratively self-contained information units (called topics or learning objects) that omit transitional phrases and cross-references. The resulting topic-oriented documents can not match the quality of narrative documents, which provide a better coherence and guidance of the reader. However, narrative documents can not be easily modularised and are less suited for reuse.

This work addresses the adaptation of narrative documents. It applies the topic-oriented principles of modularisation and reuse to the document-centered world. Narrative documents are modularised into infoms, for which all transitional phrases and cross-references are marked. Infoms and their semantic/narrative transitions as well as variant relations are modelled as graphs. These graphs are processed during the content planning of narrative documents during which appropriate infoms are selected and arranged according to their semantic/narrative transitions as well as the user's context. Since narrative transitions are visualised by words and phrases, they can reduce the adaptability of infoms. To improve the exchangeability of infoms, infoms are thus enriched with alternative transitions and cross-references. With these enrichments, appropriate narrative transitions can be selected according to the combination/arrangement of document parts into user-specific documents. The narrative context of documents is thus no longer static but becomes dynamic.

The novelty of this work is the representation of the discourse structure and narrative flow of a document in separation to its mathematical structure. The markup of transitions allows users to modularise their documents, while preserving coherence and consistency. Users are in full control of the adaptation workflows and can decide whether semantic, narrative, or their individual constraints should be prioritised.

To illustrate the proposed adaptation services for narrative documents, mathematical documents are used. This decision has required the author to take an essential aspect of mathematical text into account: Mathematics is a mixture of natural language text, symbols, and formulae. Symbols and formulae can be presented with different notations. These notations can complicate communication and acquisition processes since notations are context-dependent and can considerably vary among different communities and individuals. These variations can cause ambiguities and misunderstandings.

11. Conclusion & Future Work

In this situation, the author developed a comprehensive framework that allows users to configure notations for mathematical documents regarding a semantic, narrative, and user context. To prioritise these context and to guide the adaptation, a combination of extensional and intensional options is provided with which users can obtain a context-dependent association of notations with mathematical objects. This gives users full control over the adaptation processes. A generalised/extended form of the notation framework is used for the content planning of narrative documents. The resulting framework supports the adaptation on all document layers (content, structure, and presentation) and empowers users to guide any step of the adaptation workflow: the specification of which document parts should be adapted and which should remain unchanged, the collection of adaptation objects (notation definitions, infoms, and context parameters), and the user-specific selection of the most appropriate objects to be applied in the rendering, substitution, and ordering of document parts.

With the adaptation framework, the author has implemented the foundation for interactive, user-specific notations services, such as the display of alternative notations, the explanation of differences between the notations in a document and the user's preferences/background, the reading of notations to the users, the display of natural language terms, definitions, and examples as well as the change of notations on-the-fly while reading a document. Apart from the notation services, the framework supports content planning services, such as user-specific recommendations of supplementary materials, the interactive enrichment of documents, the variation of document content with respect to detail, difficulty, and formality, as well as multilingual presentations of documents. The arrangement of document parts according to user, semantic, and narrative constraints builds the basis for services like guided tours and other user-specific document assemblies. Some of these interactive services are already provided by systems such as *panta rhei*, SWIM, and JOBAD and could thus be evaluated.

Since mathematics is the foundation of many other disciplines, the author expects that the findings of her work can be applied to other domains and a wide range of documents. Some examples were already pointed out during a series of evaluation interviews. The participants were convinced that the proposed features can improve eLearning materials and that they can be applied to other documents like scientific eBooks, advertisements, and textbooks as well as to systems like Wikipedia, in which content is gathered and authored collaboratively. Future work has to observe whether and how the author's adaptation approach can be applied to other documents, disciplines, and systems.

11.2. Future Work

Figure 81 illustrates the author's vision of a complete semi-automatic adaptation framework. The *horizontal layers* in Figure 81 present the manual tasks that have to be performed by humans: Users have to write documents. They have to explicate the meaning of their documents so that they can be understood by a machine. For the proposed adaptation services users have to mark the meaning of symbols/formulae and their notation preferences. They also have to modularise their documents into infoms and mark the semantic/narrative transitions as well as variant relations between these infoms. In addition, users have to formalise variant relations and context properties declaratively and have to explicitly provide their user preferences to guide the adaptation workflow.

The *vertical layers* in Figure 81 present enabling technologies that can be provided based on the manual tasks. Information management techniques can process documents to, e.g., index keywords for better search results. Markup of documents facilitates XML techniques to easily capture the meaning of documents and to extract, e.g., notation preferences or document parts. Formalisations of relations and properties support machines to make inferences

along transitive or symmetric relations. A processable representation of user preferences is required for any user-specific adaptation services.

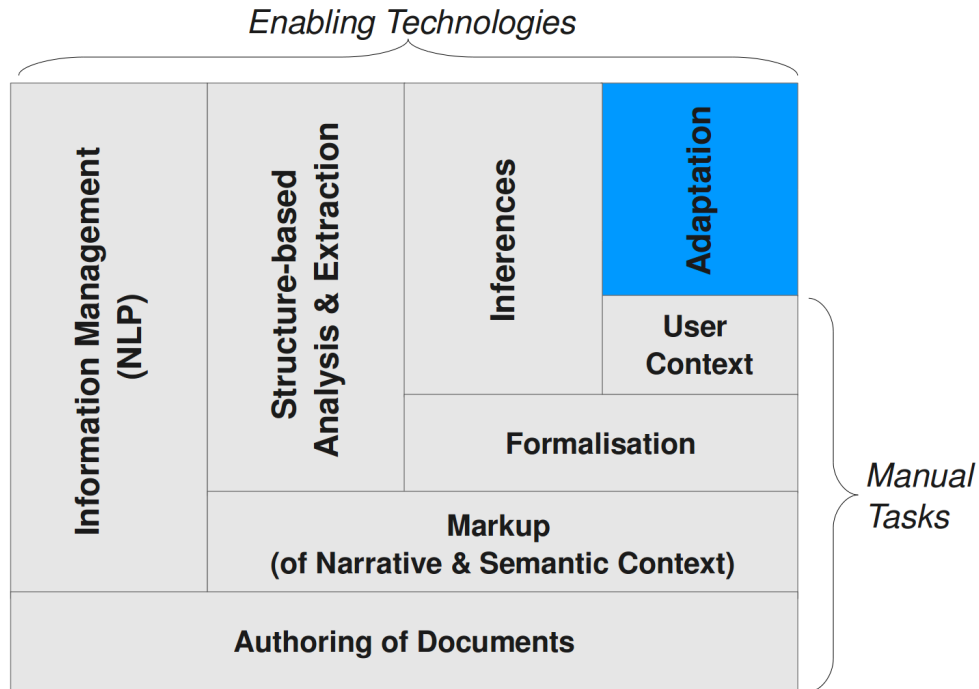


Figure 81.: The road to adaptation

The author has based her work on several assumptions, which leave many important aspects of Figure 81 for further research. These include the support of users in authoring marked-up documents — in particular notation definitions, narrative/semantic transitions, and variant relations. In addition, one needs to address the formalisation of variant relations and other context parameter and the extension of the proposed adaptation approach towards an extensible, declarative one. In a first step, this requires a (standardised) specification of the respective formalisation and document markup approach. In a second step, this requires the implementation of tools that support authoring, maintenance, and sharing of resources. A collaborative authoring environment should integrate change management techniques to resolve conflicts as well as offer means that allow users to share their marked-up materials and formalisations.

Another important future research issue is the empirical evaluation of the acceptance of the adaptation services by users and their impact on the authors' intentions. Other issues include the capturing of user preferences for adaptive services, the extension of abstract documents towards dynamic documents, i.e., documents which include queries that can be processed by an adaptation engine, as well as the application of the transition markup to topic-optimised knowledge bases to support coherent document adaptations in respective systems.

The author believes that the acceptance of the proposed framework and the usefulness of its services can only be achieved in close collaboration with researchers, instructors, and students from mathematics as well as practitioners from industry that aim at applying mathematics for their products. Future work should thus also explore synergies and potentials of future collaborations.

11.3. Epilog

The attentive reader might have realised that the proposed user/group modelling approach in Section 10.3 has been discussed and published at various scientific venues [MK09a, MK09b, Mül09a, MK08a, Mül08b, Mül08a, MK08c, WM07]. Looking at her research proposal [Mül07a], one will notice that this topic was the initial research task of the author. Several talks, discussions, and brainstorming have been necessary, before the author realised that she first needed to understand a central piece of her research puzzle: the semantic markup of documents and its exploitation to model practices and to provide adaptation services. Originally developed to handle a necessary aspect of mathematical documents (the test tube of this thesis), the notation framework has become an essential part of this thesis. The reification of notation preferences

as notation definitions proves that practices are inscribed in documents and can be modelled by semantic technologies. Having gained much better intuitions on the central piece of her research puzzle, the author is now finally prepared and skilled to address the proposed user and group modelling approach. For the author, this task is still one of the most appealing ones, next to the formalisation of context parameters to transform this work into a *declarative approach* and the application of the adaptation framework to other disciplines and other kinds of documents to transform it into a *generic approach*.



Figure 82.: ENTec [enT]

*Can anything be sadder than work left unfinished?
Yes; work never begun. (Christina G. Rossetti)*

Bibliography

- [AAS08] AAS. Need Assessment Report by the Academic Affairs Committee of the Undergraduate Student Government, 2008. Private communication.
- [ABC⁺08] Ron Ausbrooks, Bert Bos, Olga Caprotti, David Carlisle, Giorgi Chavchanidze, Ananth Coorg, Stéphane Dalmas, Stan Devitt, Sam Dooley, Margaret Hinchcliffe, Patrick Ion, Michael Kohlhase, Azzeddine Lazrek, Dennis Leas, Paul Libbrecht, Manolis Mavrikis, Bruce Miller, Robert Miner, Murray Sargent, Kyle Siegrist, Neil Soiffer, Stephen Watt, and Mohamed Zergaoui. Mathematical Markup Language Version 3.0. W3C Working Draft, World Wide Web Consortium, November 2008.
- [ABD⁺06] Serge Autexier, Christoph Benzmüller, Dominik Dietrich, Andreas Meier, and Claus-Peter Wirth. A Generic Modular Data Structure for Proof Attempts Alternating on Ideas and Granularity. In Jon Borwein and William M. Farmer, editors, *Proceedings of the Conference on Mathematical Knowledge Management*, volume 4108 of *LNAI*, pages 126–142. Springer, 2006.
- [ABMP08] Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton. RDFa in XHTML: Syntax and Processing. W3C Recommendation, World Wide Web Consortium, October 2008.
- [ACG⁺06] Andrea Asperti, Claudio Sacerdoti Coen, Ferruccio Guidi, Enrico Tassi, and Stefano Zacchiroli. Matita V0.1.0 User Manual. User manual, Computer Science Department of the University of Bologna, July 2006.
- [Act] ActiveMath Demo. Retrieved from <http://demo.activemath.org/ActiveMath2/main/menu.cmd> on August 29, 2009.
- [ada09] adaptor – A Java Library for reordering OMDoc documents. Retrieved from <https://trac.kwarc.info/panta-rhei/wiki/adaptor> on February 28, 2010, 2009.
- [AFNW07] Serge Autexier, Armin Fiedler, Thomas Neumann, and Marc Wagner. Supporting User-Defined Notations When Integrating Scientific Text-Editors with Proof Assistance Systems. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Proceedings of the Conference of Mathematical Knowledge Management*, volume 4573 of *LNAI*, pages 176–190. Springer, 2007.
- [Aio08] Andrei Aiordachioaie. Improving Panta Rhei as a Community Tool. Bachelor thesis, Jacobs University Bremen, 2008.
- [AM06] Dominique Archambault and Victor Moco. Canonical MathML to Simplify Conversion of MathML to Braille Mathematical Notations. In *Computers Helping People with Special Needs*, volume 4061 of *LNCS*, pages 1191–1198. Springer, 2006.
- [ArX] arXMLiv: Translating the arXiv to XML+MathML. Retrieved from <http://kwarc.info/projects/arXMLiv> on September 24, 2009.
- [ASC] ASCII MathML.js (ver 2.0): Translating ASCII math notation to MathML and graphics. Retrieved from <http://www1.chapman.edu/~jipsen/mathml/asciimath.html> on September 24, 2009.
- [ASFM07] Dominique Archambault, Bernhard Stöger, Donal Fitzpatrick, and Klaus Miesenberger. Access to Scientific Content by Visually Impaired People. *Upgrade: The European Journal for the Informatics Professional*, VIII(2):29–42, April 2007.
- [Bar] Kyle Barnhart. The Connexions Markup Language (CNXML), Version 0.6. Retrieved from <http://cnx.org/news/connexions-introduces-cnxml-0.6> on July 1, 2009.
- [BB07] Uldis Bojārs and John G. Breslin. SIOC Core Ontology Specification. W3C Member Submission, World Wide Web Consortium, June 2007.
- [BBH⁺02] R.G. Baraniuk, C.S. Burrus, B.M. Hendricks, G.L. Henry, A.O. Hero III, D.H. Johnson, D.L. Jones, J. Kusuma, R.D. Nowak, J.E. Odegard, L.C. Potter, K. Ramchandran, R.J. Reedstrom, P. Schniter, I.W. Selesnick, D.B. Williams, and W.L. Wilson. Connexions: DSP education for a networked world. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing.*, volume 4 of *ICASSP*, pages 4144–4147. IEEE, 2002.
- [BC04] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development – Coq’Art: The Calculus of Inductive Constructions*. Springer, 2004.
- [BCC⁺04] Stephen Buswell, Olga Caprotti, David P. Carlisle, Michael C. Dewar, Marc Gaetano, and Michael Kohlhase. The OPENMATH Standard, Version 2.0. Technical report, The Open Math Society, 2004.

Bibliography

- [BG04] Dan Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, World Wide Web Consortium, February 2004.
- [BKN07] P. Brusilovsky, A. Kobsa, and W. Neidl, editors. *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *LNCS*. Springer, 2007.
- [BL90] Tim Berners-Lee. Information Management: A Proposal. proposal, CERN, March 1989, May 1990.
- [BLFM05] Tim Berners-Lee, Roy Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, Internet Engineering Task Force, 2005.
- [BLLJ08] Bert Bos, Hakon Wium Lie, Chris Lilley, and Ian Jacobs. Cascading Style Sheets. W3C Recommendation, World Wide Web Consortium, May 1998, revised April 2008.
- [Bou68] Nicolas Bourbaki. *Theory of Sets*. Elements of Mathematics. Springer, 1968.
- [BPSM⁺08] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible Markup Language (XML) 1.0 (Five Edition). W3C Recommendation, World Wide Web Consortium, November 2008.
- [BSMG02] David J. Bartholomew, Fiona Steele, Irini Moustaki, and Jane I. Galbraith. *The analysis and interpretation of multivariate data for social scientists*. Chapman & Hall/CRC, 2002.
- [Caj93] Florian Cajori. *A History of Mathematical Notations*. Courier Dover Publications, 1993. Originally published in 1929.
- [CCJS04] A. Cohen, H. Cuypers, D. Jibeteau, and M. Spanbroek. Exercise Language. Deliverable D7, LeActiveMath Consortium, 2004.
- [CCK⁺08] Hans Cuypers, Arjeh M. Cohen, Jan Willem Knopper, Rikko Verrijzer, and Mark Spanbroek. MATHDOX, a system for interactive Mathematics. In J. Luca and E. R. Weippl, editors, *Proceedings of World Conference on Educational Multimedia, Hypermedia, and Telecommunications*, pages 5177–5182. AACE, 2008.
- [CCV10] A.M. Cohen, H. Cuypers, and R. Verrijzer. Mathematical Context in Interactive Documents. *Mathematics in Computer Science*, 3(3):331–347, 2010.
- [CD99] James Clark and Steve DeRose. XML Path Language (XPath) Version 1.0. W3C recommendation, The World Wide Web Consortium, November 1999.
- [Cen] Adobe Flash Development Center. Developer Connection: Learn Flash. Retrieved from <http://www.adobe.com/devnet/flash> on Nov 22, 2009.
- [CF09] Jacques Carette and William M. Farmer. Review of Mathematical Knowledge Management. In Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt, editors, *Conference on Intelligent Computer Mathematics*, number 5625 in *LNAI*, pages 233–246, 2009.
- [Cha87] George J. Chaitin. *Algorithmic Information Theory*. Cambridge University Press, 1987.
- [Chi] Bart Childs. An Introduction to the WEB Style of Literature Programming. Retrieved from <http://www.literateprogramming.com/IntroC.pdf> on July 25, 2009.
- [Cla09] Claudio Sacerdoti Coen and Enrico Tassi. Natural deduction environment for Matita. Technical report ubcs-2009-12, Department of Computer Science, University of Bologna, June 2009.
- [CM09] Cristian Calude and Christine Müller. Formal Proofs: Reconciling Correctness and Understanding. In Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt, editors, *Conference on Intelligent Computer Mathematics*, number 5625 in *LNAI*, pages 217–232. Springer, 2009.
- [CNX] CONNEXIONS. Retrieved from <http://cnx.org> on June 9, 2009.
- [Cod] CodeIgniter - Open source PHP web application framework. Retrieved from <http://codeigniter.com/> on June 9, 2009.
- [Cor] Microsoft Corp. Microsoft Office Suite. Retrieved from <http://office.microsoft.com> on July 7, 2009.
- [CVJ99] Wendy Chisholm, Gregg Vanderheiden, and Ian Jacobs. Web Content Accessibility Guidelines 1.0. W3C recommendation, World Wide Web Consortium, May 1999.
- [Cyc] The Syntax of CycL. Retrieved from <http://www.cyc.com/cycdoc/ref/cycl-syntax.html> on September 9, 2009.
- [Doc] DocFire. Retrieved from <http://www.docfire.com> on September 7, 2009.
- [dox] Doxygen: Source code documentation generator tool. Retrieved from <http://www.stack.nl/~dimitri/doxygen> on July 27, 2009.
- [Dub] Dublin Core Metadata Initiative. Retrieved from <http://www.dublincore.org> on August 25, 2009.

Bibliography

- [Eck98] Bruce Eckel. *Thinking in Java*. Prentice-Hall International, 1998. available online at <http://www.mindview.net/Books/TIJ4>.
- [Emm09] J. Emmons. Personal communication to M. Kohlhase, C. Müller, and N. Müller, July 9 2009.
- [enT] enTEC IT Communication. Retrieved from <http://www.entecag.ch/services.html> on March 3, 2010.
- [exa] exari: Document Assembly. Retrieved from <http://www.exari.com/document-assembly.html> on September 7, 2009.
- [Fac] Facebook. Retrieved from <http://www.facebook.com> on August 27, 2009.
- [FGT92] William Farmer, Josuah Guttman, and Xavier Thayer. Little Theories. In D. Kapur, editor, *Proceedings of the 11th Conference on Automated Deduction*, volume 607 of *LNCS*, pages 467–581, Saratoga Springs, NY, USA, 1992. Springer.
- [Fie01] Armin Fiedler. *User-adaptive Proof Explanation*. PhD thesis, Universität des Saarlandes, 2001.
- [FOA] Friend of a Friend (FOAF) project. Retrieved from <http://www.foaf-project.org> on June 9, 2009.
- [Gal23] Galileo Galilei. *The Assayer*. Rome, October 1623.
- [Gar05] Jesse James Garrett. Ajax: A New Approach to Web Applications. Essay, Adapive Path, February 18 2005.
- [GLR09] Jana Giceva, Christoph Lange, and Florian Rabe. Integrating Web Services into Active Mathematical Documents. In Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt, editors, *Conference on Intelligent Computer Mathematics*, number 5625 in *LNAI*, pages 279–293. Springer, 2009.
- [GMMW03] Paul Grosso, Eve Maler, Jonathan Marsh, and Norman Walsh. XPointer element() Scheme. W3C recommendation, World Wide Web Consortium, March 2003.
- [Gog09a] George Gogvadze. Representation for Interactive Exercises. In Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt, editors, *Conference on Intelligent Computer Mathematics*, number 5625 in *LNAI*, pages 294–309. Springer, 2009.
- [Gog09b] George Gogvadze. Semantic Evaluation Services for Web-Based Exercises. In Marc Spaniol, Quing Li, Ralf Klamma, and Rynson W.H. Lau, editors, *Advances in Web Based Learning - ICWL 2009*, *LNCS*, pages 172–181, Aachen, Germany, August 19–20 2009. Springer.
- [grg] Graph Rewrite Generator. Retrieved from <http://www.grgen.net> on January 25, 2010.
- [Gro99] The W3C HTML Working Group. HTML 4.01 Specification. W3C recommendation, World Wide Web Consortium, December 1999.
- [Gro02] The W3C HTML Working Group. XHTML 1.0 The Extensible HyperText Markup Language (Second Edition) – A Reformulation of HTML 4 in XML 1.0. W3C recommendation, World Wide Web Consortium, August 2002.
- [Har92] Godfrey H. Hardy. *A Mathematician’s Apology*. Cambridge University Press, 1992.
- [Hen04] G. Henry. Connexions: An Alternative Approach to Publishing. In *Research and Advanced Technology for Digital Libraries*, volume 3232 of *LNCS*, pages 421–431. Springer, 2004.
- [hHELoM] helm Hypertextual Electronic Library of Mathematics. helm. Retrieved from <http://helm.cs.unibo.it> on September 1, 2009.
- [Hib] Hibernate: Relational Persistence for Java and .NET. Retrieved from <https://www.hibernate.org> on September 24, 2009.
- [HKK⁺96] Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Erica Melis, Dan Nesmith, Jörn Richts, and Jörg Siekmann. Die Beweisentwicklungsumgebung Ω mega. *Informatik – Forschung und Entwicklung*, 11(1):20–26, February 1996.
- [Hota] HotDocs. Retrieved from <http://hotdocs.com> on September 7, 2009.
- [Hotb] HotDocs Glossary. Retrieved from <http://bashasys.com/index.php/software/hotdocs/hotdocs-glossary> on September 7, 2009.
- [HP91] Idit Harel and Seymour Papert. *Constructionism (Cognition and Computing)*. Ablex Pubication, 1991.
- [HRB⁺02] Brent Hendricks, Ross Reedstrom, Richard Baraniuk, Don Johnson, Bill Wilson, and Geneva Henry. Connexions: MathML and Collaborative Curriculum Development in Engineering. In *Online Proceedings of the MATHML conference*, 2002.
- [Hus09] M. Husband. Viewing Connexions Content. In *Connexions Web site*. from <http://cnx.org/content/m11837/1.14>, seen August 31, 2009.

Bibliography

- [Hut07] Graham Hutton. *Programming in Haskell*. Cambridge University Press, 2007.
- [IMD] IMDS Group. Retrieved from <http://www.imds-world.com> on September 7, 2009.
- [IS09] Antoine Isaac and Ed Summers. SKOS Simple Knowledge Organisation System. W3C Working Draft, World Wide Web Consortium, June 2009.
- [jan09] Janta: The RESTful Web Service of panta rhei. Source code available at <https://trac.kwarc.info/panta-rhei/wiki/janta>, last updated November 15, 2009, 2009.
- [jav] Javadoc. Retrieved from <http://java.sun.com/j2se/javadoc> on July 27, 2009.
- [JAX] JAX-RS: Java API for RESTful web services. Retrieved from <https://jsr311.dev.java.net> on September 24, 2009.
- [Jel] Jelly: Executable XML. Retrieved from <http://commons.apache.org/jelly> on August 23, 2009.
- [Jer] Jersey: JAX-RS Reference Implementation for building RESTful web services. Retrieved from <https://jsr311.dev.java.net> on September 24, 2009.
- [JOM08a] JOMDoc — a Java Library for OMDoc documents. Retrieved from <http://jomdoc.omdoc.org> on June 9, 2009, 2008.
- [JOM08b] JOMDoc 0.1.4 Release Notes. Retrieved from <https://trac.omdoc.org/jomdoc/wiki/JOMDoc014> on February 10, 2010, 2008.
- [KAB⁺04] E. Klieme, H. Avenarius, W. Blum, P. Döbrich, H. Gruber, M. Prenzel, K. Reiss, K. Riquarts, J. Rost, H. Tenorth, and H. J. Vollmer. The development of national educational standards - an expertise. Technical report, Bundesministerium für Bildung und Forschung, 2004.
- [Kay07] Michael Kay. XSL Transformations (XSLT) Version 2.0. W3C Recommendation, World Wide Web Consortium, January 2007.
- [KBB08] Christopher M. Kelty, C. Sidney Burrus, and Richard G. Baraniuk. Peer Review Anew: Three Principles and a Case Study in Postpublication Quality Assurance. *Special Issue on Educational Technology*, 96(6):1000–1011, 2008.
- [KBLL⁺04] Bernd Krieg-Brückner, Arne Lindow, Christoph Lüth, Achim Mahnke, and George Russell. Semantic Interrelation of Documents via an Ontology. In G. Engels and S. Seehusen, editors, *Proceedings of Tagung der Fachgruppe e-Learning der Gesellschaft für Informatik e.V. (DeLFI)*, volume P-52 of LNI, pages 271–282. Springer, 2004.
- [KGLZ09] Michael Kohlhase, Jana Giceva, Christoph Lange, and Vyacheslav Zholudev. JOBAD – interactive mathematical documents. In Brigitte Endres-Niggemeyer, Valentin Zacharias, and Pascal Hitzler, editors, *AI Mashup Challenge 2009, KI Conference*, September 2009.
- [KK06] Andrea Kohlhase and Michael Kohlhase. Communities of Practice in MKM: An Extensional Model. In Jon Borwein and William M. Farmer, editors, *Proceedings of the Conference on Mathematical Knowledge Management*, volume 4108 of LNAI, pages 179–193. Springer, 2006.
- [KK08] Andrea Kohlhase and Michael Kohlhase. Semantic Knowledge Management for Education. *Proceedings of the IEEE; Special Issue on Educational Technology*, 96(6):970–989, June 2008.
- [KL] Donald E. Knuth and Silvio Levy. The CWEB System of Structured Documentation. Retrieved from <http://www.literateprogramming.com/cweb.pdf> on July 25, 2009.
- [KLM⁺09] Michael Kohlhase, Christoph Lange, Christine Müller, Normen Müller, and Florian Rabe. Notations for Active Mathematical Documents. KWARC Report 2009-1, Jacobs University Bremen, 2009.
- [KLR07] Michael Kohlhase, Christoph Lange, and Florian Rabe. Presenting Mathematical Content With Flexible Elisions. In Olga Caprotti, Michael Kohlhase, and Paul Libbrecht, editors, *Online Proceedings of the OPENMATH/ JEM Workshop 2007*, 2007.
- [KMM07a] Michael Kohlhase, Achim Mahnke, and Christine Müller. Managing Variants in Document Content and Narrative Structures. In Alexander Hinneburg, editor, *LWA Conference Proceedings*, pages 324–229, Halle/Saale, Germany, 2007. Martin-Luther-University Halle-Wittenberg.
- [KMM07b] Michael Kohlhase, Christine Müller, and Normen Müller. Documents with flexible Notation Contexts as Interfaces to Mathematical Knowledge. In Paul Libbrecht, editor, *Mathematical User Interfaces Workshop*, 2007.
- [KMR08] Michael Kohlhase, Christine Müller, and Florian Rabe. Notations for Living Mathematical Documents. In Serge Autexier, John Campbell, J. Rubio, Volker Sorge, Masakazu Suzuki, and Freek Wiedijk, editors, *Proceedings of the Conference on Intelligent Computer Mathematics*, volume 5144 of LNAI, pages 504–519. Springer, 2008.

- [KMRW07] Fairouz Kamareddine, Manuel Maarek, Krzysztof Retel, and J.B. Wells. Narrative Structure of Mathematical Texts. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Towards Mechanized Mathematical Assistants. Proceedings of MKM/Calculemus*, LNAI, pages 296–312, Berlin, Heidelberg, 2007. Springer.
- [KMS92] Manfred Kerber, Erica Melis, and Jörg Siekmann. Analogical Reasoning with Typical Examples. SEKI-Report SR-92-13, Universität des Saarlandes, 1992.
- [Knu] Donald E. Knuth. Programs to Read. Retrieved from <http://sunburn.stanford.edu/~knuth/programs.html> on July 25, 2009.
- [Knu92] Donald E. Knuth. *Literate Programming*. The University of Chicago Press, 1992.
- [Koh] Michael Kohlhase. XSLT Stylesheet for converting OMDoc documents into XHTML. Retrieved from <http://kwarc.info/projects/xslt> on June 9, 2009.
- [Koh00] Michael Kohlhase. OMDOC: Towards an OPENMATH representation of mathematical documents. Seki Report SR-00-02, Universität des Saarlandes, 2000.
- [Koh06] Michael Kohlhase. OMDOC – *An open markup format for mathematical documents [Version 1.2]*, volume 4180 of *LNAI*. Springer, 2006.
- [Koh07] Andrea Kohlhase. CPoint — ein invasiver, semantischer Editor für Content in MS PowerPoint. In Veronika Hornung-Prähäuser, editor, *Offene Bildung im/mit dem Web 2.0!?!?*, pages 116–118. Salzburg Research, 2007. 3. Interdisziplinäre EduMedia Tagung, 16.-17. April 2007, Salzburg.
- [Koh08a] Andrea Kohlhase. *Semantic Interaction Design: Composing Knowledge with CPoint*. PhD thesis, Computer Science, Universität Bremen, 04 2008. Dissertation thesis.
- [Koh08b] Michael Kohlhase. Dependency Graph of the course. Retrieved from <http://kwarc.info/teaching/GenCS1/graph.pdf> on June 22, 2009, 2008.
- [Koh08c] Michael Kohlhase. Using L^AT_EX as a Semantic Markup Format. *Mathematics in Computer Science*, 2(2):279–304, December 2008.
- [Koh09] Michael Kohlhase. E-Mail to the KWARC Core mailinglist, June 18 2009.
- [Koi05] Marja-Riitta Koivunen. Annotea Project, October 2005. Retrieved from <http://www.w3.org/2001/Annotea> on June 23, 2009.
- [KW05] Andrea Kienle and Martin Wessner. Principles for Cultivating Scientific Communities of Practice. In Peter van den Besselaar, Giorgio de Michelis, Jenny Preece, and Carla Simone, editors, *Communities and Technologies*, pages 283–299. Springer, 2005.
- [KWZ08] Fairouz Kamareddine, J.B. Wells, and Christoph Zengler. Computerizing Mathematical Text with MathLang. *Electronic Notes in Theoretical Computer Science*, 205:5–30, 2008.
- [Lan06] Christoph Lange. A Semantic Wiki for Mathematical Knowledge Management. Master’s thesis, Universität Trier, 2006.
- [Lan08] Christoph Lange. SWiM – A semantic wiki for mathematical knowledge management. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *Proceedings of the European Semantic Web Conference: The Semantic Web Research and Application*, volume 5021 of *LNCS*, pages 832–837. Springer, 2008.
- [Lan09a] Christoph Lange. E-Mail to the KWARC Core mailinglist, June 17 2009.
- [Lan09b] Christoph Lange. Krextor – An Extensible XML→RDF Extraction Framework. In Chris Bizer, Sören Auer, and Gunnar Aastrand Grimnes, editors, *Scripting and Development for the Semantic Web (SFSW2009)*, 2009. in press.
- [Lan10] Christoph Lange. *A Semantic Wiki for Mathematical Knowledge Management*. PhD thesis, Jacobs University Bremen, 2010. work-in-progress.
- [las] Last.FM. Retrieved from <http://www.last.fm> on August 27, 2009.
- [Lea] Advanced Distributed Learning. SCORM: Shareable Content Object Reference Model. Retrieved from <http://www.adlnet.gov> on June 9, 2009.
- [Lib07] Paul Libbrecht. Content Dictionary Notations. In Olga Caprotti, Michael Kohlhase, and Paul Libbrecht, editors, *Online Proceedings of the OPENMATH/JEM Workshop 2007*, 2007.
- [Liu07] Bing Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, volume XX of *Data-Centric Systems and Applications*. Springer, Berlin, Germany, 2007.
- [LK09] Christoph Lange and Michael Kohlhase. A Mathematical Approach to Ontology Authoring and Documentation. In Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt, editors, *Conference on Intelligent Computer Mathematics*, number 5625 in *LNAI*, pages 389–404. Springer, 2009.

Bibliography

- [loc] locutor: a library for management of change on semi-structured documents. Retrieved from <http://code.google.com/p/locutor> on January 25, 2010.
- [LP08] Christoph Lange and Alberto González Palomo. Easily Editing and Browsing Complex OpenMath Markup with SWiM. In Paul Libbrecht, editor, *Online Proceedings of the Mathematical User Interfaces Workshop*, 2008.
- [LSY03] Greg Linden, Brent Smith, and Jeremy York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.
- [Łuk67] Jan Łukasiewicz. *Philosophische Bemerkungen zu mehrwertigen Systemen des Aussagenkalküls, Comptes rendus des séances de la Société des Sciences et des Lettres de Varsovie 23:51-77 (1930)*. Translated by H. Weber as *Philosophical Remarks on Many-Valued Systems of Propositional Logics*. Oxford Clarendon Press, 1967. Originally published in 1930.
- [LVon] Ming Li and Paul Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*, volume XXIV of *Texts in Computer Science*. Springer, New York, NY, USA, 2008, third edition.
- [LW91] Jean Lave and Etienne Wenger. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, 1991.
- [LW06] Paul Libbrecht and Stefan Winterstein. Internationalizing LeActiveMath. Deliverable D13, LeActiveMath Consortium, 2006.
- [MAF⁺01] E. Melis, E. Andres, A. Franke, G. Goguadse, P. Libbrecht, M. Pollet, and C. Ullrich. ACTIVE MATH system description. In Johanna D. Moore, Carol Luckhard Redfield, and W. Lewis Johnson, editors, *Artificial Intelligence in Education*, volume 68 of *Frontiers in Artificial Intelligence and Applications*, pages 580–582. IOS Press, 2001.
- [Mata] SCORM package manager for MATHDOX. Retrieved from <http://dam02.win.tue.nl/spg-new> on August 31, 2009.
- [Matb] The MATHDOX Editor. Retrieved from <http://mathdox.org/MathDoxEditor> on August 24, 2009.
- [Matc] The MATHDOX Manual. Retrieved from <http://mathdox.org/player> on August 24, 2009.
- [Matd] MathTran. Retrieved from <http://www.mathtran.org> on September 24, 2009.
- [Mate] Math Web Search. Retrieved from <http://kwarc.info/projects/mws> on June 9, 2009.
- [MBG⁺03] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. WonderWeb Deliverable D18. Deliverable, Laboratory For Applied Ontology ISTC-CNR, December 2003.
- [MEK] MEKON: Document Generation. Retrieved from http://www.4adobe.com/Server_Products/Document_Generation on September 6, 2009.
- [Mel01] Erica Melis. User Model Description. DFKI Report, DFKI, 2001.
- [MFEN08] E. Melis, A. Faulhaber, A. Eichelmann, and S. Narciss. Interoperable Competencies Characterizing Learning Objects in Mathematics. In *Intelligent Tutoring Systems*, volume 5091 of *LNCS*, pages 416–425. Springer, Berlin/ Heidelberg, Germany, 2008.
- [Mil] Bruce Miller. LaTeXML: A LaTeX to xml converter. Web Manual at <http://dlmf.nist.gov/LaTeXML>, retrieved on June 22, 2009.
- [MIT] MIT Open Courseware: 6.042J / 18.062J Mathematics for Computer Science. Retrieved from <http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-042JSpring-2005/CourseHome/index.htm> on June 9, 2009.
- [MK07] Christine Müller and Michael Kohlhase. panta rhei. In Alexander Hinneburg, editor, *LWA Conference Proceedings*, pages 318–323, Hallo/ Saale, Germany, 2007. Martin-Luther-University Halle-Wittenberg.
- [MK08a] Christine Müller and Michael Kohlhase. Communities of Practice in Mathematical E-Learning. Research report, Centre for Discrete Mathematics and Theoretical Computer Science, University of Auckland, Nov 2008.
- [MK08b] Christine Müller and Michael Kohlhase. Communities of Practice in Mathematical eLearning. In *Online Proceedings of the Workshop on Mathematical and Scientific eContent*, pages 34–35, 2008.
- [MK08c] Christine Müller and Michael Kohlhase. Towards A Community of Practice Toolkit Based On Semantically Marked Up Artifacts. In M.D. Lytras, John M. Carroll, Ernesto Damiani, and Robert D. Tennysonet, editors, *Proceedings of the 1st World Summit of the Knowledge Society: Emerging Technologies and Information Systems for the Knowledge Society*, volume 5288 of *LNAI*, pages 41–50, Berlin/ Heidelberg, Germany, 2008. Springer.

Bibliography

- [MK09a] Christine Müller and Michael Kohlhase. Context Aware Adaptation: A Case Study on Mathematical Notations. *Information Systems Management*, 26(3):215–230, 2009.
- [MK09b] Christine Müller and Michael Kohlhase. Semantic Technologies for Mathematical eLearning. In *Proceedings of the Final JEM Workshop*, RWTH Aachen University, Aachen (Germany), 2009.
- [MLUM05] Shahid Manzoor, Paul Libbrecht, Carsten Ullrich, and Erica Melis. Authoring Presentation for OPEN-MATH. In Michael Kohlhase, editor, *Proceedings of the Mathematical Knowledge Management Conference*, volume 3863 of *LNAI*, pages 33–48. Springer, 2005.
- [MM04] Frank Manola and Eric Miller. RDF Primer. W3C Recommendation, World Wide Web Consortium, February 2004.
- [MMT] The MMT Language and System. Retrieved from <https://svn.kwarc.info/repos/kwarc/rabe/Scala/doc/mmt.pdf> on Nov 22, 2009.
- [MOV06] Jonathan Marsh, David Orchard, and Daniel Veillard. XML Inclusions (XInclude) Version 1.0 (Second Edition). W3C Recommendation, World Wide Web Consortium, November 2006.
- [MS04] Erica Melis and Jörg Siekmann. ActiveMath: An Intelligent Tutoring System for Mathematics. In Leszek Rutkowski, Jörg Siekmann, Ryszard Tadeusiewicz, and Lotfi A. Zadeh, editors, *Artificial Intelligence and Soft Computing - ICAISC 2004*, volume 3070 of *LNAI*, pages 91–101, Berlin, Germany, 2004. Springer.
- [MUGL09] Erica Melis, Carsten Ullrich, Giorgi Gogvadze, and Paul Libbrecht. Culturally Aware Mathematics Education Technology. IGI-Global, May 4 2009. in press.
- [Mül06] Normen Müller. An Ontology-Driven Management of Change. In Alexander Hanft, editor, *LWA Conference Proceedings*, pages 186–193, Hildesheim, Germany, 2006. Universität Hildesheim.
- [Mül07a] Christine Müller. *Lectora: Towards an Interactive, Collaborative Reader for Mathematical Documents*. PhD thesis, March 14 2007. Research proposal.
- [Mül07b] Christine Müller. Panta Rhei: Case Study Fall2007. Retrieved from http://kwarc.info/projects/panta-rhei/papers/cs_Fall2007.pdf on June 22, 2009, December 2007.
- [Mül08a] Christine Müller. Towards CoPing with Information Overload. In Joachim Baumeister and Martin Atzmüller, editors, *LWA Conference Proceedings (FGWM)*, pages 33–40, Würzburg, Germany, 2008. Universität Würzburg.
- [Mül08b] Christine Müller. Towards the Adaptation of Scientific Course Material powered by Community of Practice. In Joachim Baumeister and Martin Atzmüller, editors, *LWA Conference Proceedings (ABIS)*, pages 41–43, Würzburg, Germany, 2008. Universität Würzburg.
- [Mül09a] Christine Müller. Communities of Practice & Semantic Web: Stimulating Collaboration by Document Markup. In Jianhua Yang, Athula Ginige, Heinrich C. Mayr, and Ralf-D. Kutsche, editors, *Information Systems: Modeling, Development, and Integration: Third International United Information Systems Conference*, volume 20 of *LNBIP*, pages 432–437. Springer, 2009.
- [Mül09b] Christine Müller. General Computer Science Lens. In *Connexions Web site*. from <http://cnx.org/lenses/cmuller/general-computer-science-lens>, seen August 31, 2009.
- [Mül10] Normen Müller. *Change Management on Semi-Structured Documents*. PhD thesis, Jacobs University Bremen, 05 2010. Dissertation thesis, submitted for defense.
- [MVW05] Jonathan Marsh, Daniel Veillard, and Norman Walsh. *xml:ind* Version 1.0. W3C recommendation, World Wide Web Consortium, September 2005.
- [MW07] Normen Müller and Marc Wagner. Towards Improving Interactive Mathematical Authoring by Ontology-driven Management of Change. In Alexander Hinneburg, editor, *LWA Conference Proceedings*, pages 289–295, Halle/Saale, Germany, 2007. Martin-Luther-University Halle-Wittenberg.
- [Nat09] Natural Numbers (2009, June 14). In *Wikipedia, the free encyclopedia*. from http://en.wikipedia.org/w/index.php?title=Natural_number&oldid=296390357, seen June 15, 2009.
- [Neta] Netflix. Retrieved from <http://www.netflix.com> on December 11, 2009.
- [Netb] Netflix Prize. Retrieved from <http://www.netflixprize.com> on December 11, 2009.
- [Nie09] André Nies. *Computability and Randomness*. Oxford University Press, 2009.
- [NK07] Immanuel Normann and Michael Kohlhase. Extended Formula Normalization for ϵ -Retrieval and Sharing of Mathematical Knowledge. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Towards Mechanized Mathematical Assistants. Proceedings of MKM/Cal-culemus*, number 4573 in *LNAI*, pages 266–279. Springer, 2007.

Bibliography

- [Nor09a] Bryce Nordgren. E-Mail to the OPENMATH mailinglist. Retrieved from <http://openmath.org/pipermail/om/2009-February/001203.html> on June 22, 2009, February 28 2009.
- [Nor09b] Bryce Nordgren. Workspace to Develop a Math Domain for DITA (2009, February 21). In DITA *wiki*. from <http://dita.xml.org/wiki/workspace-to-develop-a-math-domain-for-dita>, seen June 9, 2009.
- [NPW02] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.
- [NW03] Bill Naylor and Stephen M. Watt. Meta-Stylesheets for the Conversion of Mathematical Documents into Multiple Forms. *Annals of Mathematics and Artificial Intelligence*, 38(1-3):3–25, 2003.
- [NWBKR08] Andreas Nauerz, Marting Welsch, Fedor Bakalov, and König-Ries. Adaptive Portals: Adapting and Recommending Content and Expertise. In Joachim Baumeister and Martin Atzmüller, editors, *LWA Conference Proceedings (ABIS)*, pages 44–50, Würzburg, Germany, 2008. Universität Würzburg.
- [OM-] OPENMATH community. Retrieved from <http://www.openmath.org> on February 4, 2010.
- [OMC] OPENMATH content dictionaries. Retrieved from <http://www.openmath.org/cd> on June 9, 2009.
- [Ope] OpenID: Shared Identity Service. Retrieved from <http://openid.net> on June 9, 2009.
- [O’R05] Tim O’Reilly. What is Web 2.0. *O’Reilly*, September 30 2005.
- [OU] The Open University. Retrieved from <http://www.open.ac.uk> on September 5, 2009.
- [Pad04] Luca Padovani. A Math Canvas for the GNOME Desktop. In *Proceedings of the 5th Annual GNOME User and Developer European Conference (GUADEC’04)*, Agder University College, Kristiansand, Norway, 2004.
- [PAH07] Michael Priestley, Robert D. Anderson, and JoAnn Hackos. OASIS DITA Version 1.1. Standard specification, OASIS, August 2007.
- [Pal06] Alberto González Palomo. Sentido: an Authoring Environment for OMDoc. In *OMDOC – An open markup format for mathematical documents [Version 1.2]* [Koh06], chapter 26.3.
- [pana] Panta: The PHP Frontend of panta rhei. Source code available at <https://trac.kwarc.info/panta-rhei/wiki/panta>, last updated November 15, 2009.
- [panb] Panta rhei: The interactive, collaborative Document Viewer. Source code available at <https://svn.kwarc.info/repos/pantarhei/src/pantarhei/branches/pantarhei-platform>, last updated September 9, 2009.
- [Pau05] Lawrence C. Paulson. Isabelle Reference Manual. Technical report, Computer Laboratory, University of Cambridge, October 2005.
- [PBM09] Elvira Popescu, Costin Badica, and Lucian Moraret. WELSA: An Intelligent and Adaptive Web-based Educational System. In *Proceedings IDC 2009*. Springer, 2009. in press.
- [PBT08] Elvira Popescu, Costin Badica, and Philippe Trigano. Learning Objects’ Architecture and Indexing in WELSA Adaptive Educational System. *Scalable Computing: Practice and Experience*, 9(1):11–20, 2008.
- [PNL02] Adam Pease, Ian Niles, and John Li. The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Applications. In *Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web*, Edmonton, Canada, July 28–August 1 2002.
- [Pop09a] Elvira Popescu. Addressing Learning Style Criticism: The Unified Learning Style Model Revisited. In Marc Spaniol, Qing Li, Ralf Klamma, and Rynson W.H. Lau, editors, *Advances in Web Based Learning - ICWL 2009*, LNCS, pages 332–342, Aachen, Germany, August 19–20 2009. Springer.
- [Pop09b] Elvira Popescu. Diagnosing Students’ Learning Style in an Educational Hypermedia System. In Nikos Tsianos Constantinos Mourlas and Panagiotis Germanakos, editors, *Cognitive and Emotional Processes in Web-based Education: Integrating Human Factors and Personalization*, pages 187–208. IGI Global, 2009.
- [Pop09c] Elvira Popescu. Evaluating the Impact of Adaptation to Learning Styles in a Web-Based Educational System. In Marc Spaniol, Qing Li, Ralf Klamma, and Rynson W.H. Lau, editors, *Advances in Web Based Learning - ICWL 2009*, LNCS, pages 343–352, Aachen, Germany, August 19–20 2009. Springer.
- [Pro] The Mozilla Project. XUL Programmer’s Reference Manual. Retrieved from https://developer.mozilla.org/en/XUL_Reference on June 23, 2009.

Bibliography

- [PZ06] Luca Padovani and Stefano Zacchiroli. From Notation to Semantics: There and Back Again. In Jon Borwein and William M. Farmer, editors, *Proceedings of the Conference on Mathematical Knowledge Management*, volume 4108 of *LNAI*, pages 194–207. Springer, 2006.
- [Rab08] Florian Rabe. *Representing Logics and Logic Translations*. PhD thesis, Jacobs University Bremen, 2008.
- [RDF08] RDFa Primer. W3c working group note, World Wide Web Consortium, October 2008.
- [RK08] Florian Rabe and Michael Kohlhase. An Exchange Format for Modular Knowledge. In Piotr Rudnicki, Geoff Sutcliffe, Boris Konev, Renate A. Schmidt, and Stephan Schulz, editors, *Proceedings of the LPAR Workshops*, volume 418 of *CEUR Workshop Proceedings*, pages 50–48. CEUR-WS.org, 2008.
- [Rul] The Rule Markup Initiative. Retrieved from <http://ruleml.org> on September 9, 2009.
- [sla] slashdot. Retrieved from <http://slashdot.org/> on June 16, 2009.
- [Sta07] Bob Stayton. DocBook XSL: The Complete Guide. Online documentation, 4th edition, Sagehill Enterprises, September 2007.
- [SW06] Elena Smirnova and Stephen M. Watt. Notation Selection in Mathematical Computing Environments. In Jean-Guillaume Dumas, editor, *Proceedings Transgressive Computing: A conference in honor of Jean Della Dora*, pages 339–355, Granada, Spain, 2006.
- [SWM04] Michael K. Smith, Chris Welty, and Deborah L. McGuinness. OWL Web Ontology Language Guide. W3C recommendation, World Wide Web Consortium, February 2004.
- [Tea04] The Coq Development Team. Coq Reference Manual. Technical report, Computer Laboratory, University of Cambridge, June 2004.
- [TeX05] Gnu TEXMACS. <http://www.texmacs.org>, July 2005.
- [Thu] Thunderhead NOW Enterprise Communication Platform. Retrieved from <http://www.thunderhead.com> on June 9, 2009.
- [Tin] TinyMCE – a platform independent web based JavaScript HTML WYSIWYG editor. Retrieved from <http://tinymce.moxiecode.com> on August 31, 2009.
- [twi] Twitter. Retrieved from <http://twitter.com> on August 27, 2009.
- [Ull08] Carsten Ullrich. *Pedagogically Founded Courseware Generation for Web-Based Learning*, volume 5260 of *LNCS*. Springer, Berlin, Germany, 2008.
- [ULWM04] C. Ullrich, P. Libbrecht, S. Winterstein, and M. Mhlenbrock. A Flexible and Efficient Presentation-Architecture for Adaptive Hypermedia: Description and Technical Evaluation. In Kinshuk, C.-K. Looi, E. Sutinen, D. Sampson, I. Aedo, L. Uden, and E. Kähkönen, editors, *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies*, pages 21–25, Joensuu, Finland, 2004.
- [Uni10] Jacobs University. Jacobs Student Body Statistics, 2010. Private communication.
- [UOC] Universitat Oberta de Catalunya. Retrieved from <http://www.uoc.edu> on September 5, 2009.
- [Vel] Velocity. Retrieved from <http://jakarta.apache.org/velocity> on August 23, 2009.
- [Vie] GTK MathML View0.8x. helm. Retrieved from <http://helm.cs.unibo.it/mml-widget> on September 1, 2009.
- [vK09] Anne van Kesteren. XMLHttpRequest. W3C Working Draft, World Wide Web Consortium, August 2009.
- [W3C] The World Wide Web Consortium. Retrieved from <http://www.w3c.org> on June 17, 2009.
- [Wal09a] Norman Walsh. DITA 2006 (2007, July 17). In *Norm’s musings. Make of them what you will*. from <http://norman.walsh.name/2006/03/24/dita2006>, seen September 22, 2009.
- [Wal09b] Norman Walsh. DocBook XSL Stylesheets: Reference Documentation. Online documentation, The DocBook Project, March 2009.
- [Wal09c] Norman Walsh. Implementing the Darwin Information Typing Architecture for DocBook (2005, October 21). In *Norm’s musings. Make of them what you will*. from <http://norman.walsh.name/2005/10/21/dita>, seen September 22, 2009.
- [Wal09d] Norman Walsh. Topic-oriented authoring (2007, February 5). In *Norm’s musings. Make of them what you will*. from <http://norman.walsh.name/2007/02/05/painting>, seen September 22, 2009.
- [WEL] WELSA. Retrieved from <http://software.ucv.ro/~epopescu/welsa> on August 27, 2009.

Bibliography

- [Wen05a] Etienne Wenger. Communities of Practice in 21st-century organization, 2005.
- [Wen05b] Etienne Wenger. *Communities of Practice: Learning, Meaning, and Identity*. Cambridge University Press, 2005.
- [WG102] IEEE Learning Technology Standards WG12. IEEE 1484.12.1.2002 Standard for Learning Object Metadata. Retrieved from <http://ltsc.ieee.org/wg12/par1484-12-1.html> on August 27, 2009, 2002.
- [Wie09] Jan Wielemaker. SWI-Prolog 5.6.60 reference manual. Manual, November 11 2009.
- [wik] Wikipedia – The Free Encyclopedia. Retrieved from <http://en.wikipedia.org> on March 9, 2010.
- [WM99] Norman Walsh and Leonard Mueller. *DocBook: The Definitive Guide*. O’Reilly, 1999.
- [WM07] Marc Wagner and Christine Müller. Towards Community of Practice Support for Interactive Mathematical Authoring. In Christine Müller, editor, *Online Proceedings of the 1st Workshop on Scientific Communities Of Practice*, 2007.
- [Wol00] Stephen Wolfram. Mathematical Notation: Past and Future. In Neil Soiffer, Patrick Ion, and Angel Diaz, editors, *Online Proceedings of MathML and Math on the Web: MathML International Conference*. Wolfram Research, 2000.
- [WS03] Christoph Walther and Stephan Schweitzer. About Verifun. In Franz Baader, editor, *Proceedings of the 19th International Conference on Automated Deduction (CADE-19)*, LNCS. Springer, 2003.
- [You] You Tube. Retrieved from <http://www.youtube.com> on August 27, 2009.
- [ZK09] Vyacheslav Zholudev and Michael Kohlhase. TNTBase: a Versioned Storage for XML. In *Proceedings of Balisage: The Markup Conference*, Montréal, Canada, 2009. available at <http://www.balisage.net/Proceedings/vol3/html/Zholudev01/BalisageVol3-Zholudev01.html>.
- [ZKR10] Vyacheslav Zholudev, Michael Kohlhase, and Florian Rabe. A [insert xml format] database for [insert cool application]. In *Proceedings of XMLPrague*, 2010. in print.