

State of the Art in Multilingual and Multicultural Creation of Digital Mathematical Content

Lauri Carlson, Jordi Saludes, and Andreas Strotmann

Deliverable Number:	1.2
Type of Deliverable:	R – Report
Contractual Delivery Date:	04 April 2005
Actual Delivery Date:	21 April 2005
Dissemination Level:	PU – Public
Workpackage:	1
Document Author/s:	Lauri Carlson, Jordi Saludes, Andreas Strotmann
Document Editor:	Lauri Carlson, Andreas Strotmann
Document Status:	Final
Key Words:	



Abstract

The objective of this workpackage of the WebALT eContent project is to explore the state of the art of multilingual authoring and distribution in mathematics, and deduce from it requirements for a set of multilingual and multicultural tools that harness existing technology in this area. The document surveys

- localization issues for mathematical formulae (Section 2)
- authoring and display tools for MathML-Content and OpenMath (Section 2)
- localization issues of natural language (Section 3)
- localization issues for course content formalizations (Section 4)
- issues of natural language and mathematics (Section 5)

The survey concludes with preliminary recommendations as to those available tools that are most likely to be useful for the WebALT project.

The information contained in this report is subject to change without notice and should not be construed as a commitment by any members of the WebALT Consortium. In the event of any software or algorithms being described in this report, the WebALT Consortium assumes no responsibility for the use or inability to use any of its software or algorithms. The information is provided without any warranty of any kind and the WebALT Consortium expressly disclaims all implied warranties, including but not limited to the implied warranties of merchantability and fitness for a particular use. This document may not be copied, reproduced, or modified in whole or in part for any purpose without written permission from the WebALT Consortium. In addition, to such written permission to copy, acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced.
© COPYRIGHT 2005-2006 WebALT Consortium.



Table of Contents

1	INTRODUCTION	4
1.1	CAVEAT	4
1.2	CHARACTERIZATION OF THE FIELD OF INTEREST.....	4
1.3	MATHEMATICS LOCALIZATION ISSUES.....	4
1.3.1	<i>Converting Formal Mathematical Content to Presentation</i>	<i>5</i>
1.3.2	<i>User interfaces for creating language-independent mathematical content .</i>	<i>5</i>
1.3.3	<i>Semantic Markup for Mathematical Content.....</i>	<i>6</i>
1.3.4	<i>Natural Language Generation and Mathematical Vernacular.....</i>	<i>6</i>
1.4	OVERVIEW.....	6
2	LOCALIZATION ISSUES IN MATHEMATICAL FORMULAE	7
2.1	LOCALIZATION NEEDS OF THE WEBALT PROJECT	8
2.1.1	<i>Types of Context Influencing Choice of Presentation</i>	<i>8</i>
2.1.2	<i>Localization in Presentation Markup Languages.....</i>	<i>9</i>
2.2	CONTENT-TO-PRESENTATION MARKUP RENDERING	9
2.2.1	<i>Content Markup Creation.....</i>	<i>10</i>
2.3	CONSEQUENCES FOR WEBALT.....	11
2.3.1	<i>C2P Requirements Analysis.....</i>	<i>11</i>
2.3.2	<i>Choosing Presentation Styles</i>	<i>11</i>
2.3.3	<i>Content-to-presentation renderers available.....</i>	<i>13</i>
3	LOCALIZATION ISSUES OF NATURAL LANGUAGE	14
3.1	STATE OF THE ART IN NATURAL LANGUAGE GENERATION (NLG)	14
3.1.1	<i>What NLG is used for</i>	<i>14</i>
3.1.2	<i>NLG pipeline</i>	<i>19</i>
3.1.3	<i>NLG techniques.....</i>	<i>20</i>
3.1.4	<i>Knowledge sources</i>	<i>23</i>
3.1.5	<i>Quality characteristics.....</i>	<i>26</i>
3.1.6	<i>Technologies.....</i>	<i>26</i>
3.1.7	<i>Tools</i>	<i>28</i>
3.1.8	<i>Fashion items</i>	<i>29</i>
3.2	WEBALT GENERATION TASK	30
3.2.1	<i>The data</i>	<i>30</i>
3.2.2	<i>Text types.....</i>	<i>30</i>
3.2.3	<i>Plan A.....</i>	<i>36</i>
3.2.4	<i>Evaluation of plan A.....</i>	<i>39</i>
3.2.5	<i>Plan B.....</i>	<i>40</i>
4	LOCALIZATION ISSUES FOR CCDS IMPLEMENTATIONS	41
4.1	MOTIVATION	41



4.2	LOCALIZATION AND XML	42
5	NATURAL LANGUAGE AND FORMAL MATHEMATICS.....	43
5.1	COMPREHENSIBLE OUTPUT FROM PROVERS.....	43
5.2	FORMAL MATHEMATICS FROM NATURAL LANGUAGE	44
5.3	THE LANGUAGE OF MATHEMATICS IN TEACHING.....	45
CONCLUSIONS		45



1 Introduction

In this section we will provide an overview of the field we are discussing here.

1.1 Caveat

This document surveys multilingual mathematics. However, since the WebALT project is not being funded as a research project, but as a software development effort, this survey has a specific role to play within a software development process rather than within a research project. In particular, the focus of this survey is much narrower than usual, aiming to inform the selection of software tools for use in the development process of the WebALT demonstrator rather than to open up new avenues for research.

Thus, many interesting research projects that might be given considerable amounts of space in the writing of a research survey will be given short shrift here, although we do endeavour to mention a sufficiently wide range of potentially relevant R&D.

1.2 Characterization of the Field of Interest

The WebALT project has contracted to develop a demonstrator that provides a database of mathematical exercises for undergraduate students across Europe. The entries in the database are to be created in a language-independent manner, using a generalized version of OpenMath, MathML-Content, or a similar formalized content markup for mathematics. From this formalized mathematical content, natural language generation (NLG) mechanisms are used to convert this content so that it can be presented to students and teachers in their own language and in a natural fashion. Languages that are to be targeted for the demonstrator include only European languages; however, there is considerable interest in marketing the resulting software and database worldwide in the long run.

1.3 Mathematics localization issues

While there is a wide range of interesting research being done on cultural and sociological issues in mathematics teaching especially for minorities or developing cultures, a topic that is of particular interest to the WebALT project concerns the ways that mathematical content is presented to, or created by, students in different European cultures and languages.



This includes differences in notations used in mathematical formulae (e.g. the different ways of expressing “gcd” in different languages, or the different notations used for writing out long division in different cultures), different degrees to which formal notations or natural language (NL) are used for expressing such content in different contexts (typically with an increasing shift towards formal presentations as the target audience’s mathematical maturity increases), and especially in non-European cultures, how different writing systems influence the presentation of mathematical formulae and texts.

1.3.1 Converting Formal Mathematical Content to Presentation

Again, this is a complex area of research, but we will need to concentrate on those parts that are of immediate and practical relevance to the WebALT R&D effort.

Considerable research has been done on ways of presenting complex mathematical content to human experts in the field of automatic theorem proving, for example, where long-winded and highly detailed proofs make it hard for the human user to follow the underlying logic. However, the WebALT project is interested in working with much easier material, presumably obviating the need for much of the complexity of the algorithms that are being investigated in this area.

Another field of particular interest to WebALT is that of generating presentation markup from MathML or OpenMath or similar content markup for mathematical formal content. While most of the research and development effort in this area has been in the English language and for Anglo-Saxon cultural environments, a few other contexts have been the subject of research as well.

This is true also for the case of screen-readers for mathematical content, where content-markup has been studied as a source for generating audio – a potential future market for WebALT Inc., but not one that we intend to target within the short time of the project itself.

1.3.2 User interfaces for creating language-independent mathematical content

At first blush, this topic has nothing to do with multi-lingual mathematics as such, but on closer examination, it is very relevant because it considers how people can most naturally create mathematical content of the kind we are interested in, and do so in their own language. Typically, this would involve some



sort of keyword-driven pseudo-natural-language input mode at the simpler end of the spectrum, or it might involve a full-blown NL parser for the limited universe of discourse that we intend to cover.

Again, there has been considerable interest in this area in the Mathematical Knowledge Management field, where the lack of effective tools for converting existing mathematical literature into formalized mathematical content has long posed a formidable barrier to practical research in this area. As before, the WebALT project could hope to avoid some of the particularly hard problems involved in building such general-purpose tools by restricting itself to the immediate necessities of the particularly simple types of content found in its exercise database.

1.3.3 Semantic Markup for Mathematical Content

Closely related to the previous section, there has been quite some research in the Mathematical Knowledge Management area into ways of representing the semantic analysis of a NL mathematical text along with its original presentation layout. This includes, in particular, a considerable amount of research into type theories that are capable of supporting the semantic analysis of the informal language employed in mathematical vernacular.

1.3.4 Natural Language Generation and Mathematical Vernacular

While Natural Language Generation, or NLG, is a well-established field of research in Computational Linguistics, few have studied its application to generating “mathematical vernacular”, or the peculiar mix of natural language text and mathematical formulae that is so characteristic of publications in mathematics, from content-marked-up formal mathematics, and fewer still doing so in multiple linguistic or cultural contexts.

Because of the central role that the idea of using NLG on content-marked-up mathematical content plays in the WebALT project, we provide a thorough survey of the literature on multi-lingual NLG, and explore its use in generating mathematical vernacular.

1.4 Overview

In the following, each of the fields of research briefly sketched above will be surveyed in a separate section. The survey concludes with recommendations for a set of tools to use in the WebALT project, and a longer-term outlook for WebALT Inc.



In Chapter 2, we survey the literature on localization issues for mathematical formulae, while in Chapter 3, we survey the literature on localization issues for natural language text generation. Together, these two aspects compose the main ingredients of core content that the WebALT project's final showcase should handle: a) the introductory text fragments to be described and b) the formulae, for which transformation instructions are to be provided, thereby fully describing a typical mathematical exercise.

In Chapter 4, we survey the literature on localization of those international standards that govern the communication of course contents between e-learning systems, focussing on those that deal with assessments and exercise materials. This chapter also discusses technology that is relevant for localizing the WebALT Course Content Dictionaries (CCDs) that aim to standardize the structure of the courses that the WebALT problem database aims to enhance.

Chapter 5 briefly surveys a range of topics in the area of language and mathematics that are not directly applicable to the WebALT project itself, but which nevertheless deserve to be mentioned for their ability to inform potential future developments.

2 Localization Issues in Mathematical Formulae

As the MathML Recommendation (Carlisle et al. 2003) points out,

there are many to one mappings from presentation to semantics and vice versa. For example the mathematical construct "H multiplied by e" is often encoded using an explicit operator as in $H \times e$. In different presentational contexts, the multiplication operator might be invisible "H e", or rendered as the spoken word "times". Generally, many different presentations are possible depending on the context and style preferences of the author or reader. [...]

Mathematical presentation also changes with culture and time: some expressions in combinatorial mathematics today have one meaning to a Russian mathematician, and quite another to a French mathematician [...].

It is perhaps not a coincidence that this observation is used to explain the intent of MathML Content Markup, and it is the fundamental reason why the WebALT project has opted to attempt to derive localized presentations from language-independent semantic representations of the content of its subject material.



2.1 Localization Needs of the WebALT Project

Briefly, therefore, the basic idea behind the WebALT approach to building a multi-lingual and multi-cultural European database of exercise materials for e-Learning in mathematics, is to store the materials in something as much like MathML-Content as feasible, thus representing the common underlying meaning of those materials within the context of a particular course in mathematics, and to render the material into as appropriate a localized version of that content as feasible, using existing software solutions.

The fundamental question then is just what renderings are appropriate in which context for a given underlying pattern of meanings.

2.1.1 Types of Context Influencing Choice of Presentation

The quote given above touches upon different types of context that may influence the choice of a mathematical notation for a particular mathematical construct:

- The detailed semantics of the mathematical operation (e.g., the kind of multiplication, such as cross-product or scalar product of vectors)
- The level of mathematical sophistication of the target audience (e.g., using “x” notation for simple multiplication in elementary school, but “invisible” multiplication in secondary and higher education)
- Typographic conventions (e.g. using explicit multiplication symbols to distinguish multiplication from other possible readings of an “invisible” operator, e.g. between integers)
- Individual stylistic choices (e.g. the use of a mirrored capital E vs. a “big Or” notation for existential quantification, with corresponding changes in layout positions for the respective parts of the quantified expression)
- Cultural or linguistic distinctions (e.g. “tan” vs. “tg” in different parts of the world, and the different notations for the greatest common divisor in different languages – “gcd” in English, “ggT” in German, “mcd” in Spanish, “MCD” in Italian, and so on in different languages, all abbreviating the respective languages’ translation of that term)
- Choice between formal or informal rendering (e.g. “f where x is a S” vs. “ $\lambda x:S . f$ ”)



- Choice between different rendering modes (e.g. visual vs. aural)

While these problems are mentioned in the MathML literature, for example, there are few if any systematic studies on the correct way to choose good renderings based on all of these kinds of context, although there are a few publications that deal with some of these problems in an ad-hoc fashion. This lack of a general solution to the problem of converting a semantic representation of mathematical content into appropriate renderings may well be one of the reasons that presentation markup, be it MathML-Presentation or LaTeX, is still the most common way to author mathematical content.

2.1.2 Localization in Presentation Markup Languages

Such layout description languages essentially beg the questions that the automatic rendering of content markup brings up. All the choices between presentation variants that need to be made here tend to be made, explicitly, by the author, based both on the authors personal preferences and her awareness of the target audience's requirements. Thus, for example, there does not even appear to be an internationalized TeX macro `\gcd` available that would generate the correct three-letter code given a particular language's "babel" TeX environment.

Nevertheless, research into providing TeX or Presentation MathML support for representing and rendering formulae in Arabic cultures and languages, and the considerable problems associated with this endeavour (Eddahibi et al. 2004), can give an idea of the amount of work that would be involved in an internationalised version of TeX or MathML-Presentation. Thus, it is hardly surprising that the question of internationalisation and localisation was explicitly left for later in the current version 2.0 of MathML. However, it is becoming quite clear that these questions are now scheduled for investigation in the next round of MathML standardization, and the WebALT project may well decide to share the experience that will be gained in this area by the time this topic becomes current in the MathML Working Group.

2.2 Content-to-Presentation Markup Rendering

Thus far, all efforts that we are aware of that endeavour to present mathematical formulae based on a content markup representation, are monolingual in nature, mostly generating formulae for some version of an Anglo-Saxon cultural and linguistic context in the higher education arena. Limited support may be available for authors to make their own purely stylistic choices of a rendering for their



mathematical concepts, e.g. in some of the MathML-Content to Presentation transformation stylesheets available on the web (Smirnova and Watt 2004), and support for aural rendering has been investigated in the literature (Raman 1994).

Nevertheless, the very availability of quality rendering engines for content-marked-up mathematical formulae even for only a single language and culture is very promising, for two reasons. First, it proves the feasibility of a significant component of the WebALT endeavour: if it can be done for one language or culture, it can be done in others, too. Second, the renderings of mathematical formulae in different contexts do not differ except in details across the globe, so that an existing method for rendering content-markup should be adaptable to other contexts fairly easily.

2.2.1 Content Markup Creation

Content-to-presentation rendering is unfortunately of little use if no content-marked-up mathematical content could be made available. Two methods are commonly used to address this issue, and we consider briefly how they might be addressed.

The most common technique for creating content markup versions of mathematical content may well be the use of computer algebra systems or similar symbolic mathematical computing tools to create a content-markup rendering of a formula computed by such a tool. Perhaps surprisingly, such tools do not address localization issues for formulas at all, and may therefore not be good vehicles for the WebALT project, but rather potential future customers for WebALT technology.

Nevertheless, all CA systems do support editing of formulas, some GUI-based, some text-flow based. However, they require familiarity with a formal way of specifying mathematics, and do not support any localization or other type of specialization of such input methods.

For this reason, some projects have begun working on stand-alone editors for content-marked-up mathematical formulae. For the use of teachers creating exercises and students entering answers to exercise questions, WebALT requires an editing interface with a number of features:

- It needs to allow restricting the range of mathematical concepts that are allowed in a mathematical expression based on the requirements of the application.



- It needs to support template-driven editing that is configurable at runtime, in order to allow for multi-lingual input methods.
- It needs to come with configuration tools that allow teachers to adapt the tools to their students' needs by modifying the templates made available to the students.

Some of the partners, namely UPC and Math4More, have been working on such a tool for some time, and have demonstrated the potential of their editor for adaptation to different linguistic contexts.

2.3 Consequences for WebALT

Supposing that at least the simpler exercises can be treated according to Plan A, detailed in section 3.2.3, a few consequences arise from the preceding discussion as well as from the detailed discussion of NLG below.

2.3.1 C2P Requirements Analysis

An important role of WebALT Workpackage 4 (Evaluation and Quality Control) is to "formulate details of localizations and translations for content-to-presentation (C2P) style-sheets covering different languages and countries" targeted in this project, by constantly working to create real-life exercises using appropriate versions of the WebALT system. Thus, a detailed study of appropriate renderings of the material available in the demonstrator in different languages and countries, a study that so far appears to be lacking in the literature, is already slated to be undertaken in WP4.

By sharing this experience with the MathML Working Group, it may be possible to minimize the amount of XSLT programming for C2P stylesheets to be done within the WebALT group, but a certain amount of such programming is probably unavoidable in practice.

In this context, it might be a good idea to study ways for the WebALT project to obtain some kind of official status in the MathML working group when and if it decides to focus on internationalisation issues during the development of an upcoming version 3.0.

2.3.2 Choosing Presentation Styles

First of all, not all the choices available in principle to a C2P rendering engine need to be made at runtime. Provided WebALT uses MathML Content or



OpenMath as a semantic representation of the meaning of a formula, the following choices may be made at different stages:

- The mapping between a detailed semantics of a mathematical operation and its rendering can be made configurable by the author, implementable as something like the MathML “semantics” element. Teachers may need to override this kind of choice (see stylistic choices below).
- The level of mathematical sophistication of the target audience needs to be judged by the author, but may also need to be adjusted by the teacher and/or automatically based on an adaptive student model. Consequently, rendering hints that correspond to such a choice would need to be inserted into the otherwise purely semantic WebALT-internal representation of an exercise.
- Typographic conventions (e.g. using explicit multiplication symbols to distinguish multiplication from other possible readings of an “invisible” operator, e.g. between integers) need to be handled by a generic C2P handler anyway, although the details may vary between different writing systems, for example, and may therefore require localization to some extent. Nevertheless, this is a choice reserved for the software systems involved, and not available to authors, teachers, or students.
- Individual stylistic choices should be available to the teacher rather than the author. The ORCCA C2P stylesheet is being developed to take this into account (Smirnova and Watt 2004).
- Cultural or linguistic distinctions between formalisms have not been added to any existing MathML C2P stylesheets yet, although Mozilla developers have been publishing some thoughts on the issue (Vincent 2004). The WebALT project will probably need to do some of this code development as part of the project, since this type of adjustment should be available through automatic rendering.
- The choice between formal or informal rendering is often part of the decision of the author and/or teacher concerning the level of mathematical sophistication of the target audience, but it may also be a matter of personal stylistic preference in particular circumstances. Thus, the author’s or teacher’s exercise creation tools should include mechanisms for presenting such choices in an intuitive fashion, and use those choices to



add a corresponding rendering hint to the internal semantic representation of an exercise.

- Choice between rendering modes (e.g. visual vs. aural) is not at this point being considered for WebALT, although it might be possible to do this with a reasonable amount of extra work. The reason for this is that the problem in principle reduces partly to the case where the entire formal content is tagged as to be rendered “informally” in the sense introduced above, with aural rendering then performed by the standard screen reader for the NL text thus generated. However, in this case, the text planning phase of NLG becomes considerably more important than in the case of embedding formulae as some sort of two-dimensional layout into “preplanned” text.

2.3.3 Content-to-presentation renderers available

For MathML, XSLT stylesheets are available from two sources, namely from the World Wide Web Consortium (D. Carlisle (2002)’s so-called Universal C2P Stylesheet, which is universal only in the sense of working in all browsers and with all plugins) and from the ORCCA research center, which is a partner of the WebALT project. Both translate to MathML-Presentation with an Anglo-Saxon cultural and linguistic bias.

The ORCCA C2P stylesheet attempts to provide some support for user choice of actual notification used. However, it is somewhat limited as to what it can do to support such choices without reverting to explicitly coding parallel MathML-Presentation markup, since MathML provides only limited support for encoding stylistic choices in MathML-Content. Indeed, adding support for such rendering hints (especially the `xml:lang` attribute) to the MathML language may be one of the areas that the WebALT project may need to try and influence the future development of MathML, in order to stay compatible in the long run.

There does not appear to be standard facilities available for determining the final rendering of a MathML-Content formula in a standard browser, possibly using a special plugin, based on the choice of language or culture in the user’s browser, e.g. using the `xml:lang` attribute. However, as of XSLT version 2.0, Xpath 2.0 expressions are now available for distinguishing between language settings, so that it should be fairly straightforward to adapt any existing MathML C2P XSLT stylesheet to a multicultural context as long as WP4 and/or the MathML Working Group provide the necessary input as to the requirements for such adaptations.



If WebALT chooses to use OpenMath instead of MathML-Content, a MathML “phrasebook” for OpenMath will be necessary for translation in both directions, e.g. one produced by one of the WebALT members (RIACA 2004).

Most Computer Algebra systems provide the means to generate some kind of screen, HTML, MathML-Presentation, and/or TeX rendering of their internal mathematical structures. Since most of them also read and write MathML-Content and/or OpenMath, any such Computer Algebra system could function as a C2P renderer, possibly with a MathML phrasebook added for OpenMath rendering. However, despite having user interfaces for a variety of languages, these maths renderers tend to use some kind of North-American “standard” rendering instead of localizing it to the same linguistic and cultural context as the user interface.

3 Localization Issues of Natural Language

3.1 State of the Art in Natural Language Generation (NLG)

Natural language generation is documented in many web sources, including:

- ACL special interest group for generation SIGGEN www.siggen.org
- Bremen B-Z list of generation projects by John Bateman and Michael Zock <http://www.fb10.uni-bremen.de/anglistik/langpro/NLG-table/NLG-table-root.htm>
- Biannual International NLG conferences (INLG [98,00,02,04](#))

An introductory textbook on NLG is Reiter/Dale 2000. This section builds extensively on a recent NLG state of art survey by Kalina Bontcheva (Bontcheva 2004).

3.1.1 What NLG is used for

The Bremen list itemizes 360 NLG projects 1961-2004. The selection ranges from the famous pattern matching psychologist ELIZA to the 10-language generation system KPML. The list is not exhaustive; for example, Aarne Ranta’s Grammatical Framework (GF) is not on the list. Not many of the projects have web addresses. Fewer yet have working demos. Of the few demos that work, the commercial CoGenTex (<http://www.cogentex.com/>) system may be representative; it



produces canned-looking text from tabled data. More projects were started in the 1990's than before or after:

2000-4	38
1995-9	103
1990-4	102
1980-9	82
1970-9	26
1960-9	7

3.1.1.1 What from?

The generation task is defined by the input-output mapping. Input to generation is something structured and less natural-language like. Most commonly, it is some tabular information from a database source. In more refined cases, it may be formulae in some semantically-interpreted formal language. A third possibility is parsed natural language, for instance generation in a Machine Translation (MT) system.

- Tables
- Formulae
- Parsed NL

3.1.1.1.1 Tables

Many *template based* generators work from data presented in tabular form. The generation tasks are reminiscent of relational database reporting operations of selecting, sorting and grouping data records and fields, and splicing them into NL text templates.

3.1.1.1.2 Formulae

Generating from formulae is closer to the traditional task of programming language compilers of producing object code from source code. This is the plan A scenario of WebALT.

The OpenMath content dictionaries (CDs) already contain specifications for mathematical concepts and operations, including a small number of entries for mathematical problem solving primitives.

3.1.1.1.3 Parsed NL



Linguistically oriented generators work from linguistically rich input which corresponds to a parse graph or tree of a NL parser. For instance, KPML generates from a linguistically rich logical form given as a LISP expression:

```
(EXAMPLE
  :NAME      EX-SET-6
  :GENERATEDFORM "Do they meet precisely along the edges, and do they
cover most of the graph?"
  :TARGETFORM  "Do they meet precisely along the edges, and do they
cover most of the graph?"
  :LOGICALFORM
((M C)
 (M / NONDIRECTED-ACTION :LEX MEET :SPEECHACT YES-NO-QUESTION :ACTOR
  (O / OBJECT :PRONOUN THEY)
  :PARALLEL-EXTENT
  (E / OBJECT :LEX EDGE :NUMBER PLURAL :DETERMINER THE)
  :MANNER
  (P / QUALITY :LEX PRECISELY))
 (C / ASCRIPTION :LEX COVER :SPEECHACT YES-NO-QUESTION :INCHOATIVE-Q
  INCHOATIVE :DOMAIN
  (O / OBJECT :EXPRESS-TYPE NO :IDENTIFIABILITY-Q IDENTIFIABLE :NUMBER
  PLURAL)
  :RANGE
  (G / OBJECT :LEX GRAPH-NOUN :DETERMINER THE :QUANTITY-SELECTION-Q
  QUANTITY :QUANTITY-SELECTION-ID G :CARDINAL-QUANTITY-Q
  NOTCARDINAL :PROPORTION-QUANTITY-Q PROPORTION :HIGH-QUANTITY-Q
  HIGH)))
 :SET-NAME |nigel-exerciseset|
```

In Plan B, outlined in section 3.2.5, we expect the course creator to use a constrained natural language structure editor (possibly an extension of the mathematical formula editor, possibly some other product) to produce tagged NL text for input to the multilingual generation pipeline.

3.1.1.2 What to?

Earlier on, generation meant text generation. With speech recognition and synthesis, generating utterances for a dialogue system has become a hot topic. Even more recently, generators get to prepare output for several alternative or simultaneous channels, including interactive graphics and video.

- Text
- Speech
- Multimodal

Bontcheva (2004) distinguishes between

- Static systems



- Interactive (or dynamic) systems

Static systems generate whole texts which are read later by users (e.g. technical documentation). These systems do not receive direct feedback from their readers. Dynamic systems generate explanations or dialogue on the fly. These systems typically keep track of the interaction history and can adapt their models and output based on user feedback. The WebALT scenario is static from a student's point of view, but dynamic for a course developer.

3.1.1.3 What for?

The domains for which NL is generated vary. 143 do not specify domain, 12 claim to be domain independent. No domain gets over 5% of the total. 72 domains are mentioned just once, the other 36 are listed below. But clearly, NLG is a domain dependent undertaking.

Total	360
--	143
Medical	17
Independent	12
Weather	12
Instructions	10
Explanation	8
spoken dialogue	6
Stories	6
General	5
Appointments	4
online help	4
stock market	4
formal proofs	3
Geometry	3
Museums	3
News	3
Psychiatry	3
Recipes	3
route descriptions	3
scene description	3
user modeling	3
Arguments	2
business letters	2
Documentation	2
labor market	2
Manuals	2
Music	2
Paraphrase	2



Reservations	2
Soccer	2
Software	2
spatial relationships	2
teaching syntax	2
Telecom	2
temporal connectives	2
Tutoring	2

3.1.1.4 What languages?

251/360 systems generate English, with 205 of these generating English exclusively. KPML is the only truly multilingual sentence generator of the lot with over 10 languages. The ones boasting 3 or more languages are listed below:

KPML	English, German, French, Dutch, Japanese, Czech, Russian, Bulgarian, Spanish, Greek, Chinese	11
Multimeteo	English, French, Spanish, German, Dutch	5
CLE	English, Swedish, French, Spanish	4
KISS-Lexicalizer	Dutch, English, German, French	4
M-PIRO	English, Greek, Italian, Spanish	4
TEMSIS	German, French, English, Portuguese	4
AGILE	Bulgarian, Czech, Russian	3
Amalgam	German, French, English	3
Anthem	English, Dutch, French	3
Drafter-II	English, French, Italian	3
FUF	English, Spanish, Hebrew	3
GIST-tax	English, German, Italian	3
KOMET	English, German, Dutch	3
MDA	English, French, German	3
MIKE	English, French, Japanese	3
ProfGlot	English, French, Dutch	3
Susy	English, French, Dutch	3
VM-GECO-II	German, English, Japanese	3

There are GF resource grammars covering basic structures of seven languages (English, Finnish, French, German, Italian, Russian and Swedish). As a curiosity, GF generates numerals in 80 languages.

3.1.1.5 What coverage?

Unlike parsing, where it is common to aim for general language coverage, most NLG is constrained language and domain specific. This is natural because NLG inputs are by necessity restricted and domain specific, unlike parser inputs. For this reason too, NLG projects tend to be ad hoc and short-lived.



3.1.2 NLG pipeline

Unlike parsing, generation is usually understood to include text generation as well as sentence generation (aka surface realisation). While parsers are satisfied to get some way out of natural language, generators have to get all the way into it from given input. The usual route can be described as a pipeline (the names and subdivisions of the phases may vary):

- Text planning (content selection, rhetorical arrangement)
- Sentence planning (aggregation, lexical choice, referring expressions)
- Sentence generation (constructions, word order, morphology)
- Rendering (orthography, graphics, sound)

3.1.2.1 Text planning (content selection, rhetorical arrangement)

The two most commonly used discourse organisation approaches are text schemas and text planning based on rhetorical structures. For efficiency reasons most applied NLG systems prefer schema-like approaches (Bontcheva 2004:7). Text schemas, introduced by McKeown (1985), are script-like structures which represent discourse patterns. They can be applied recursively to generate coherent multisentential text satisfying a given, high level communicative goal. For example, the PEBA II generation system uses the following schema to generate a comparison of two animals:

```
CompareAndContrast ->
  LinnaeanRelationship CompareProperties
CompareProperties ->
  CompareProperty CompareProperties
CompareProperties -> end
```

(Bontcheva 2004:21).

The extent to which text planning appears in WebALT, will depend on the level of support that we will provide for the creation of the exercise source texts. One scenario is that the generator is in fact a multilingual editor for exercises. In this case, text planning is done interactively as the exercise is created.

In Plan A, we do not expect to have to do content selection or text planning separate from sentence planning (i.e. a text is generated the same way as a sentence, but inserting periods instead of commas).



3.1.2.2 Sentence planning (aggregation, lexical choice, referring expressions)

Sentence planning too, can partly happen interactively at the time of the creation of the original exercise. The exercise creator gives the semantics of the exercise and at the same time gives (perhaps with metadata, perhaps by example) an indication of how that semantics is going to be split into sentences and phrases. The other language generators can then use the instructions they obtain from the source example to produce automatically analogously constructed renderings.

In many cases, sentence planning can be viewed as application of logical or mathematical inference to the formulae to generate from. Aggregation (Reape/Mellish 1999), in particular, refers to processes of expression grouping and removal of redundancy analogous to the algebraic distribution law going from $ab + ac$ to $a(b+c)$. For another example, equivalences between deeply embedded formulae and less embedded ones related by variable sharing can be used to generate alternative natural language sentence plans.

If so, we should consider doing sentence planning by applying a controlled inference mechanism already on the mathematical source formula. The mechanism could be term rewriting, possibly implemented as another GF grammar.

3.1.2.3 Sentence generation (constructions, word order, morphology)

Surface realisation (sentence generation) happens in the generator(s) according to linguistic, partly cross-linguistic, partly language specific rules.

3.1.2.4 Rendering (orthography, graphics, sound)

Rendering (including rendering of math formulae) will happen outside the generator proper on the basis of tags produced by the generator.

3.1.3 NLG techniques

A rough and ready typology of NLG techniques is the following:

- canned text
- templates
- grammar

The difference is in the amount of combinatorics. All generators use some canned text (at least words), most have templates (with slots to insert stuff that does



not need further generating, like numbers, formulae, etc.), and the more grammar-like ones use rules which combine, vary, and arrange templates.

3.1.3.1 Canned text

The simplest solution to generation is to use canned text. The inverse in parsing would be string search. This approach is appropriate only if limited or no reasoning is required about the canned information.

3.1.3.2 Templates

The next level is to use boilerplate text which has slots to fill in by varying input bits. The inverse of this is parsing by pattern matching, as used in information extraction and in chatbots.

Templates are text-production techniques based on manipulating character strings with little or no use of linguistic or domain knowledge. A very simple example is the error messages generated by a computer program. Another example is mail-merge (e.g. Word). Bontcheva (2004:22).

Depending on the variety of the input and the output, writing new templates for each combination may become unfeasible because there will be too many alternatives to cover. The natural way to go then is to start creating new templates by combining old ones by rule (Bontcheva 2004:5).

Examples of recursive template generation languages are Exemplars by CoGenTex, TG/2 (Busemann 1996) and XtraGen (Stenzhorn 2002).

3.1.3.3 Grammar

When the combinatorics of templates calling one another gets out of hand, the need arises to separate the declarative description of the input-output mapping (*grammar*) from the algorithm which applies it (parser or generator). The separate parts are by then easier to maintain and exchange. In general, grammar based generation gives better language independence and better control of language variation. Grammar development is largely analogous to (other) programming. Busemann (2004) calls a grammar with 100 to 200 rules (whatever one counts as one rule) small. A grammar with thousands of rules needs a development environment.

In language technology, the difficult thing to handle is ambiguity (= nondeterminism). While the input is less often ambiguous in generation, natural



language is, so the generator needs to choose what to say. This happens when one generates from anything like a realistic grammar of a natural language.

This makes generation a search problem: the task is to find a path from an input graph to an output string. In search, the crucial factor is the search branching factor. Different generation methods address this factor.

There are three popular ways to generate with a grammar:

- top down (the ancient one; the downside is too much branching as often with top down natural language processing)
- bottom up; this has been a popular approach combined with minimal recursion semantics (MRS), cf. below
- head driven (combination of both): find head word from goal top down, easy if semantics is shared; then find rule applicable to that word bottom up; then generate sisters of head; then find the next rule up for head phrase, until you are back to goal.

Top down generation works if there is not too much difference between the input and output structures (what is too much is a matter of practice). For instance GF generation involves just different linearizations of an abstract syntax tree into a concrete one (for instance, a string.)

If there is too much choice in the grammar, one can hope to do better by starting at the other end, choosing words first. This happens in bottom up and head driven generation.

These two methods are said to require semantic monotonicity (that words share semantics with the generation goal). Actually, this just makes the methods easier to apply.

Bottom up generation is currently done from input that comes in a flat list or bag (known as minimal recursion semantics), but that too is just an expedient. By normalising input semantics into MRS, language technologists also hope to address the problem of logical equivalence (meaning preserving variation or synonymy in the input) without recourse to a theorem prover.

3.1.3.4 Hybrids

Template-based NLG refers to using NLG techniques for content selection and text planning and template-based techniques for surface realisation.



The main reason for using templates is that surface realisers based on grammars often lack computational efficiency which is crucial for applied NLG systems, especially Web based ones.

Grammar based realisers can be also quite expensive to maintain (requiring a linguist if changes are to be made) and the acquisition of the necessary lexicon, grammar and morphology are quite laborious tasks in their own right (Bontcheva 2004:22). In fact,

There are thousands, if not millions, of application programs in everyday use that automatically generate texts; but probably fewer than ten of these programs use the linguistic and knowledge-based techniques that have been studied by the natural language generation community. (Reiter 1995)

Another motivation for the use of templates and/or canned text in certain circumstances is the lack of sophistication and robustness of most grammar based NLG techniques. Template-based systems can also allow NLG-generated fragments to be inserted into a template slot or canned phrases to be used in generated sentences (one way of achieving this is by using a phrasal lexicon – see below). The main goal is to use generation only in the stages where it adds value at a reasonable price and performance.

From a task-oriented perspective, template-based generators avoid or simplify many tasks involved as part of the surface realisation process, including sentence aggregation, lexicalisation, referring expression generation, syntactic processing. (Bontcheva 2004:22)

The fact is that natural language usage builds on phrases and collocations, which really are a kind of template, a fact to which legacy NLG grammars have given insufficient attention.

In WebALT, Plan A is to use a grammar based generator, while Plan B involves a hybrid.

3.1.4 Knowledge sources

Quoting the enumeration in Bontcheva (2004, section 3.3), a generator's knowledge sources may include:

- Domain Knowledge Base (KB) or database
- Discourse/interaction history



- User model
- Grammar
- Lexicon

Resources worth adding to Bontcheva's list are

- Morphology
- Speech synthesis

Not all NLG systems have or need all of these resources. (e.g., a user model is only needed if adaptivity is required). The most commonly used ones are the domain knowledge/data base, lexicon, and grammar. Some applied systems use templates and canned text to speed up surface realisation and avoid the cost of building grammars and lexicons.

3.1.4.1 Domain Knowledge Base (KB) or database

This contains the domain knowledge which is the basis of the generated texts (e.g., facts about animals or software specifications). Some of the information can be derived from the generator's own resources, e.g., domain-independent ontologies in the Penman Upper Model. This resource can be created by the user, manually acquired or obtained from existing application data. (Bontcheva 2004:8)

In the WebALT case, this is the collection of math problems coming from a database or created by the course creator on the fly.

3.1.4.2 Discourse/interaction history

This refers to information about what has been presented so far. For instance, if a system maintains a list of previous explanations, then it can use this information to avoid repetitions, refer to already presented facts or generate comparisons (Bontcheva 2004:8).

Editor context or editing history may become relevant if we want to support generation of sets of interrelated problems.



3.1.4.3 User model

This is a representation of the users' domain knowledge, plans, goals, beliefs, and other relevant information used by the generator to adapt its output (Bontcheva 2004:8).

User here can mean the course creator or the student. The course creator may have standing preferences to e.g. terminology or grammatical constructions (let/assume, where...). The student's level of math literacy could determine what Jordi Saludes calls formulability factor below.

3.1.4.4 Grammar

This is the target language grammar which is used to generate linguistically correct utterances. Grammar formalisms which have been used in language generation include:

- Systemic grammar, e.g. KPML (Bateman 1997)
- Phrase structure grammar, e.g., Referent Grammar (Sigurd 1987)
- Unification grammar, e.g., Functional Unification Formalism (Elhadad 1996)
- Tree-Adjoining Grammar (Kilger/Poller 2000)

In the classification, GF is another extension of phrase structure grammar (known as multiple parallel CFG, Ljunglöf 2004). Some formalisms allow reversible grammars, i.e., grammars that can be used both for parsing and generation. GF is an example of such a formalism.

3.1.4.5 Lexicon

A lexicon entry for each word typically contains information such as part of speech, inflection class, etc. Despite various initiatives to create reusable Language Technology (LT) lexical resources, most NLG systems need to develop proprietary lexicons (Bontcheva 2004:9).

3.1.4.6 Morphology

For languages with complex morphology like Finnish, a separate morphological parser and generation component is usually needed lest the full form lexicon grows unmanageably big.



3.1.5 Quality characteristics

When comparing NLG solutions, the following criteria can be used (cf. ISO 9126 criteria for software evaluation):

- Maintainability
- Accessibility
- Portability
- Scalability
- Efficiency

3.1.5.1 Maintainability

- how easy or difficult it is for a non-NLG expert to modify the system's behaviour (e.g., provide new lexicalizations of concepts or modify a text plan, Bontcheva 2004:4)

3.1.5.2 Accessibility

- Who owns it (IPR's)
- What does it cost (licencing conditions)

3.1.5.3 Portability

- What platforms are supported
- How easy it is to adapt the generator to another domain, language, etc.

3.1.5.4 Scalability

- How big can it get
- How fast is it when it gets bigger

3.1.5.5 Efficiency

- How fast it runs
- What resources does it require (processor, memory etc.)

In WebALT, the weight is on the earlier quality characteristics.

3.1.6 Technologies

In information technology, (though not alone there) each new fashion in computer language and paradigm calls for reinventing everything programmed in the predecessors.



3.1.6.1 LISP

Natural language generation pretty much started with Lisp in the 1960's, but the language is almost extinct by now. Examples: KPML (Bateman 1997), TG/2 (Busemann 1996), FUF/SURGE (Elhadad 1996).

3.1.6.2 C and C++

C (later C++) got popular for platform dependent programming for its close control of resources. Established solutions still get recoded in them for speed, but they are not a likely choice for prototypes or for platform independent integration work.

3.1.6.3 Prolog

Prolog largely pushed functional language programming to the roadside in the 1980's and 1990's as the lingua franca of symbolic language technology. Herbrand style relational quantifier-free forms of first order logic, Horn clauses, and Robinson unification plus resolution proofs pushed aside lambda calculus, functional programming and term rewriting. By no accident, this happened at the same time as Montague's program began to look too complex to carry through computationally, in the face of the heavy nondeterminism of natural language. However, the fashionable days of Prolog are over by now too, and we are back to deterministic and procedural programming with an object oriented twist. Example: Multimodal Functional Unification Grammar (Reitter 2004).

3.1.6.4 Haskell

Meanwhile, mainly outside LT, functional language diehards went the opposite way to develop more strictly typed functional languages, of which Haskell is one of the best known, but still definitely a minority language. Haskell, like other special purpose languages, is best used offline to produce code for general purpose languages to run. Example: GF (Ranta 2004).

3.1.6.5 Java

Java, originally designed for platform-independent web programming, has become the current general purpose programming language, thanks to its object oriented language design, despite its verbosity and inefficiency. Example: XtraGen (Stenzhorn 2002).



3.1.6.6 XSL(T)

The XML style sheet and transformation language is a recursive tree transformation language originally for rendering XML document trees, but it has since version 2, which allows holding unfinished node sets in variables, become a functional programming language on its own. It can get compiled by XSLC into java translets for better speed. Its main advantage is that it is so close to XML, which is currently the most popular document structuring language. Example: Wilcock (2001).

On the other hand, Stenzhorn (2002) found XSL unsatisfactory because with it he

- was not able to appropriately handle morphology,
- could not parameterize the generation process at the desired level and
- had no possibility to generate alternatives or recover from dead ends during generation since XSL is lacking a backtracking mechanism.

3.1.6.7 RDF, OWL, Jena

As a W3 standard for the 70's AI semantic networks, graph definition language RDF and the ontology language OWL defined on it, are gaining popularity, despite the current shortage of efficient tools to work with the resulting ontologies. The main claim to fame of the OWL/RDF/XML standards family is that it comes from a widely accessible and accredited authority.

3.1.7 Tools

One step up from technologies are general purpose generator tools, like

- Grammar development environments
- Multilingual editors

3.1.7.1 Grammar development environments

Grammar development environments were the vogue in the 1980's and 1990's, and many appeared written in Lisp and Prolog using the then novel graphics GUI's. There are many more parsing environments than generator environments.

The best known NLG environment is no doubt the KPML system (Bateman 1997), which is a further development of the sentence generation component of the Penman text generation system. It is now maintained at the University of



Bremen, Faculty of Linguistics and Literature. The system is a graphically-based development environment for the construction, maintenance, and use of large-scale grammars written with the framework of Systemic-Functional Linguistics (SFL). Grammars have been developed using KPML for a variety of languages including English, German (ongoing), Dutch (not recent), Chinese, Spanish (recent, 2 versions), Russian, Bulgarian, and Czech (but not Finnish). Many of these grammars are freely available for further research and development work within the NLG community. The English grammar is the very large Nigel grammar under development since the early 1980s in the Penman project at USC/ISI, Los Angeles.

KPML is written in ANSI Common Lisp, with the graphical interface in the Common Lisp Interface Manager (CLIM). The source code may be compiled and installed in full using commercial Common Lisp products supporting Common Lisp and CLIM under Unix and Windows. There are also standalone executable images available for Windows systems that may be downloaded and used without compilation and without requiring an additional Lisp software licence.

A more recent generator grammar development environment written in Java is eGram (Busemann 2004). eGram supports the TG/2 and XtraGen generators.

3.1.7.2 Multilingual editors

The idea is that of constrained language structured editing. The generator guides the preparation of the document to make sure it can be localised appropriately. The terms *symbolic authoring* and WYSIWYM (what you see is what you mean) have been coined for this (Scott et al.1998). There is a javascript demo of this on Richard Powers' WYSIWYM home page at Brighton <http://www.itri.bton.ac.uk/projects/WYSIWYM/javademo.html>

There is a GF demonstration gramlet which functions as a structured document editor in which you can edit and linearize the syntax trees described by the grammar. <http://www.cs.chalmers.se/~krijo/gramlets.html>

3.1.8 Fashion items

Language technology funders' interest is attracted more by fashionable novelties than gradual improvement of basic language resources. In generation, the fashionable topics include

- Multilinguality



- Multimodality
- Adaptivity

3.1.8.1 Multilinguality

NLG systems are ideal for multilingual applications, as they tend to be relatively easy to port to a new language. Yet few existing NLG systems are true polyglots. GF is one of them.

3.1.8.2 Multimodality

Here the idea is to produce output on several channels (text, speech, graphics and touchscreen) in parallel or interleaved. This will not be addressed in WebALT except for formulae, figures and charts included in the exercises. For an example see Reitter (2004).

3.1.8.3 Adaptivity

Adaptivity covers strategies which allow the system behaviour to vary and develop with the user. In WebALT, this might boil down to some degree of user control over the generation process. An example NLG system which provides users with some control over text plans, using a graphical user interface, is ModEx (Lavoie et al. 1996). GF multilingual structure editor is another example.

3.2 WebALT generation task

3.2.1 *The data*

We have gone over two sample sets of problems obtained from the Finnish mathematicians.

3.2.2 *Text types*

Three types of text to handle can be distinguished.

- Short problems
- Long problems
- Instructions



Functionally, typical short problems are *numerical problems*, where NL text (if any) serves to name the mathematical operation to perform on formulae. Long problems include *verbal problems* where part of the exercise is to formulate the question in mathematical terms. Another type of long problem could be called *explorations*. They consist of several steps and may include or come close to the third type of text, *instructions*. An example of each type:

- Short problems

1 23 1 Give the function FIG in the form FIG where FIG is a polynomial and FIG is a rational function with the property FIG.

- Long problems

2 28 16 1 A planned gold mine is estimated to produce gold FIG tons per year, where the positive real number FIG is the number of years passed from the beginning. What is the total amount of gold that will be produced, expressed as an integral?

4 10 25 5 6 1 The function FIG has the zeroes 0, 2 and FIG. Draw, in the same plot, the graphs of the function f and that of its tangent at the average of two of its zeroes. What do you notice? Use Maple for the plots.

- Instructions

4 5 9 2 11 1 The answer was not correct. This is one example for FIG and FIG. FIG. FIG. Now one of the sentences looks quite odd, doesn't it?

The first two text types could be separated operationally by sentence count and/or word count. In principle, natural language should give a choice between making a long sentence or a paragraph of short ones. But the short/long problem distinction seems to correlate better with sentence count.

3.2.2.1 Short problems

- Long sentences from problems-problemtext set:

1 23 1 What lower bound of FIG is obtained using the lower Riemann sum when the interval is decomposed into four subintervals of equal length?

1 23 1 Give the function FIG in the form FIG where FIG is a polynomial and FIG is a rational function with the property FIG.

2 22 3 1 One of these propositional sentences can't be added to the set FIG (with only one sentence) so that FIG would stay consistent. Which one?

1 20 1 Which integral of a rational function will FIG (FIG a rational function) result if you perform the standard substitution FIG?

- Long sentences from problems-kysymykset set:

1 27 1 1 Use the above formula to figure out how large a number n guarantees that the error of Simpson's approximation for the integral FIG is less than 0.0001.



1 27 1 Determine the following derivatives by first computing a few low order derivatives and then finding a general formula for all the derivatives of the function in question.

1 26 1 1 Käytä yllä annettua virhearviota sen tutkimiseksi kuinka suuri luvun n (jakovälien lukumäärän) on oltava jotta Simpsonin kaavan antaman arvion virhe integraalin FIG arvolle on < 0.0001 .

1 25 1 1 How many terms are needed for the partial sums of the following convergent series to get an approximation of the sum with error < 0.0001 ?

1 22 1 Give an example of functions f and g such that FIG is defined, f is not one-to-one (injective) but FIG is one-to-one.

1 22 1 By repeating the above commands "Hint" and "Rule" you may go through all the steps of a manual solution of the problem.

1 21 1 Give an example of functions f and g such that FIG is defined, f is not one-to-one (injective) but FIG is one-to-one.

1 21 1 Anna esimerkki funktioista f ja g siten, että yhdistetty funktio FIG on määritelty, f ei ole injektio, mutta FIG on injektio.

1 21 1 Anna esimerkki funktioista f ja g siten, että yhdistetty funktio FIG on määritelty, f ei ole injektio, mutta FIG on injektio.

- Average sentences from problems-problemtext set:

1 9 1 Which limit does L'Hospital rule give you for FIG?

1 9 1 Mikä seuraavista kaavoista ei ole kaavan FIG looginen seuraus?

1 9 1 Mikä on tulos, jos suoritat standardikorvauksen FIG integraalissa FIG?

1 9 1 GRAPHICS Kuvassa on funktioiden FIG, FIG ja FIG kuvaajat.

1 9 1 Calculate FIG, where FIG is the inverse of FIG.

1 8 1 What results if you substitute FIG in FIG?

1 8 1 What is the correct substitution to compute FIG?

1 8 1 What can you say about the limit FIG?

1 8 1 Vad är korrekt om du vill beräkna FIG?

1 8 1 Vad är korrekt om du vill beräkna FIG?

1 8 1 Mikä seuraavista sijoituksista palauttaa integraalin FIG rationaalifunktion integraaliksi?

1 8 1 Mikä seuraavista on oikea muotoilu osittaisintegroitaessa integraalia FIG?

1 8 1 Mikä seuraavista on korrekti muotoilu osittaisintegroitaessa integraalia FIG?

1 8 1 Mikä on tulos, kun korvataan FIG integraalissa FIG?

1 8 1 Mikä epäyhtälö osoittaa, suppeneeko vai hajaantuuko integraali FIG?

1 8 1 Laske FIG, missä FIG on funktion FIG käänteisfunktio.

1 8 1 How to compute an unbounded integral like FIG?

1 7 2 Mikä seuraavista on ekvivalentti muotoilu seuraavalle ilmaisulle? FIG

1 7 2 Does FIG have a derivative at FIG?

1 7 1 Om FIG, vad är korrekt in FIG?

1 7 1 Miten löydät oikean sijoituksen integraalin FIG laskemiseksi?

1 7 1 Mitä saat, jos korvaat FIG integraalissa FIG?

1 7 1 Minkä raja-arvon L'Hospitalin sääntö antaa lausekkeelle FIG?

1 7 1 Mikä seuraavista on oikea muotoilu integraalille FIG?

- Average sentences from problems-kysymykset set:

1 9 1 Using the definition of the derivative show that FIG.

1 9 1 Use the definition of the integral to compute FIG.

1 9 1 Use Maple to compute FIG for the function FIG.

1 9 1 In Problems 2 -- 4 determine the given limits.

1 9 1 In Problems 1 ja 2 solve the given equations.

1 9 1 If FIG is it true that also FIG converges?

1 9 1 For which values of p , the series FIG converges?

1 9 1 Find a power series representation for the function FIG.

1 9 1 Express the indefinite integral FIG as a power series.

1 8 2 Suppenevatko tehtävissä 5 -- 7 annetut epäoleelliset integraalit?



1 8 2 Osoita, että jos $f(x)$, niin $f(x)$ suppenee.
 1 8 1 Use Mathematical Induction to prove the last estimate.
 1 8 1 Käytä Maplea derivaatan $f(x)$ kuvaajan tutkimiseksi kun $f(x)$.
 1 8 1 Find the derivatives of the following functions: $f(x)$.
 1 8 1 Determine the length of the parametric curve $f(x)$.
 1 8 1 Determine the inverse function of the function $f(x)$.
 1 7 2 Todista tehtävässä 5 johdettu kaava oikeaksi induktiolla.
 1 7 1 Using integration by parts, show that $f(x)$.
 1 7 1 Use this fact in the following problem.
 1 7 1 Tehtävissä 1 ja 2 ratkaise annetut yhtälöt.
 1 7 1 Show that if $f(x)$ then $f(x)$ converges.
 1 7 1 Millä parametrin p arvoilla sarja $f(x)$ suppenee?
 1 7 1 Käytä Maplea funktion $f(x)$ derivaatan $f(x)$ laskemiseksi.
 1 7 1 Interpret the following limits as definite integrals:.
 1 7 1 Find the length of the curve $f(x)$.
 1 7 1 Determine those functions f , for which: $f(x)$.
 1 7 1 Determine the length of the curve $f(x)$.
 1 6 1 Käytä integraalin määritelmää integraalin $f(x)$ laskemiseksi.
 1 6 2 Käyttämällä derivaatan määritelmää osoita, että $f(x)$.
 1 6 1 Käytä sopivaa trigonometristä sijoitusta integraalin laskemiseksi.
 1 6 1 Esitä funktio $f(x)$ suppenevan potenssisarjan summana.

- Short sentences from problems-problemtext set:

1 2 1 Määritä $f(x)$.
 1 2 1 Määrää $f(x)$.
 1 2 1 Laske $f(x)$.
 1 2 1 Integroi $f(x)$.
 1 2 1 Integrate $f(x)$.
 1 2 1 Find $f(x)$.
 1 2 1 $f(x) = ?$
 1 2 1 Evaluate $f(x)$.
 1 2 1 Determine $f(x)$.
 1 2 1 Calculate $f(x)$.
 1 2 1 Beräkna $f(x)$.
 2 1 3 1 $f(x)$. Laske $f(x)$.
 2 1 3 1 $f(x)$. Calculate $f(x)$.

- Short sentences from problems-problemtext set:

1 3 1 $f(x)$ kun $f(x)$.
 1 3 1 $f(x)$ for $f(x)$.
 1 3 1 Derivoi seuraavat funktiot.
 2 2 3 2 Olkoon $f(x)$. Määrää $f(x)$.
 2 2 3 1 Let $f(x)$. Determine $f(x)$.
 1 2 1 Simplify $f(x)$.
 1 2 1 Let $f(x)$.
 1 2 1 Determine $f(x)$.
 1 1 1 $f(x)$.

3.2.2.2 Long problems

- Long texts from problems-problemtext set:

4 7 9 9 4 1 We already know that $f(x)$ is provable. Can you continue the proof to conclude $f(x)$? You may use all tautologies of propositional logic. $f(x)$ abbreviates $f(x)$.
 4 6 9 10 9 1 The axioms are $f(x)$, $f(x)$, $f(x)$. The only rule of proof is Modus Ponens. You may also use Deduction Theorem: $f(x)$ iff $f(x)$. We already know $f(x)$ Can you prove $f(x)$?



4 6 9 10 7 1 The axioms are FIG, FIG, FIG. The only rule of proof is Modus Ponens. You may also use Deduction Theorem: FIG iff FIG. Can you prove FIG and FIG?

4 6 7 7 6 1 Tiedämme että kaava FIG on todistuva. Osaatko jatkaa todistusta ja päätellä FIG? Todistuksessa voit käyttää valitsemiasi propositiologiikan tautologioita. FIG on lyhenne kaavasta FIG.

4 5 5 8 7 1 Aksiomat ovat FIG, FIG, FIG. Päätelystä on Modus Ponens. Voit myös käyttää deduktiolausetta: FIG joss FIG. Oletetaan tunnetuksi FIG Osaatko todistaa FIG?

4 5 5 8 7 1 Aksiomat ovat FIG, FIG, FIG. Päätelystä on Modus Ponens. Voit myös käyttää deduktiolausetta: FIG joss FIG. Can you prove FIG and FIG?

- Long texts from problems-kysymykset set:

15 1 5 4 15 13 9 7 25 10 9 9 7 1 1 7 1 FIG. Determine FIG and FIG. Justify your answer. Hint: use Maple or a calculator to compute few points of the set FIG. Observe that points of the set FIG are either positive or negative. Consider the positive and the negative points separately. Use Maple to study their properties. If you do not have Maple at your disposal, the justification part of this problem becomes too difficult and must probably be left undone. In this case a heuristical argument suffices for justification. In the computations, use the Maple command `assume(n, posint)`. That will allow Maple to perform helpful simplifications. The sample solved problem in <http://etcetera.pdf> is useful when solving this problem.

14 1 5 3 10 11 6 8 16 5 3 1 1 6 1 FIG. Määrää FIG ja FIG. Perustelee vastauksesi. Vihje: laske Maplilla tai laskimella joitakin joukon FIG pisteitä. Joukon FIG pisteet kannattaa jakaa kahteen osaan: positiivisiin ja negatiivisiin. Tarkastele näitä osia Maplilla erikseen. Käytä laskuissa Maple komentoa `assume(n, posint)`. Tällöin Maple osaa sieventää esiintyviä lausekkeita paremmin. Jos Maple ei ole käytettävissä tämän tehtävän perusteluosa muuttuu vaikeaksi ja on luultavasti jätettävä tekemättä. Tällöin heuristinen perustelu riittää. Esimerkkitehtävästä "<http://etcetera.pdf>" on hyötyä tätä tehtävää ratkaistaessa.

6 9 23 21 9 17 16 1 In the following you need Maple 8 or newer. Below is the beginning of a Maple worksheet which can be used to find out how to perform the integration in problem. Observe that some of the hints that Maple gives lead to an integration involving the "sec" or the "cosec" functions. We have not considered these functions in class. For example, the first substitution given as a hint may not be very useful for us. Version 8 and newer versions of Maple contain the teaching module in the library `Student[Calculus1]`.

5 8 15 17 8 13 1 The function S is defined by setting FIG. Here the value of the function FIG at FIG is set to be 1. This makes the function to be integrated everywhere continuous and the function S is well defined. Sketch the graph of the function f. For which values of x the function S has local maximum points?

5 6 14 10 18 12 1 Seuraavaan tarvitaan Maple 8 tai uudempi. Alla on alku Maple työkirjasta, jolla voidaan tutkia miten tehtävän 5 integraali lasketaan. Huomaa, että jotkut Maplen ohjeista johtavat funktion "sec" integrointiin. Tätä funktiota ei ole käsitelty erikseen kurssilla, joten esimerkiksi alla annettua ensimmäistä sijoitusta ei välttämättä kannata käyttää. Maple 8 ja uudemmat Maplen versiot sisältävät integroinnin yksityisopetuspaketin (kirjastossa `Student[Calculus1]`).

5 4 4 8 10 7 2 Voit käyttää Maplea ym. tehtävien ratkaisujen löytämiseen. Ratkaisut on kuitenkin harjoituksissa esitettävä ilman Maplea.

Virtuaalisissa harjoituksissa laskut on esitettävä Word+MathType dokumentissa askel askeleelta. Pelkkä Maple työkirjan kopiointi ei riitä.

4 9 10 8 7 1 Function f is even if FIG for all FIG. Function f is odd, if FIG for all FIG. Let FIG be a differentiable odd function. Show that its derivative is even.



3.2.2.3 Instructions

- Long texts from exercises-messages.txt:

5 17 8 13 27 7 1 Notice that all except one left parts of the implications are always false in a nonempty universe. An implication is true in that case. (Formulae of the form FIG are always true in an empty universe.) The left part of the implication FIG can also be true (if the structure has more than one element), but the right part is then false. The second formula is not valid.

5 15 10 9 21 6 1 Huomaa, että kaikkien paitsi yhden implikaation etujäsenet ovat aina epätosia, jos avaruus ei ole tyhjä. Implikaatio on tosi, jos sen vasen puoli on epätosi. (Tyhjässä avaruudessa kaavat muotoa FIG ovat aina tosia.) Implikaation FIG vasen puoli voi olla myös tosi (jos nimittäin struktuurissa on useampia alkioita), mutta silloin oikea puoli on epätosi. Kakkoskaava ei siis ole validi.

4 5 9 2 11 1 The answer was not correct. This is one example for FIG and FIG . FIG . Now one of the sentences looks quite odd, doesn't it?

4 3 3 2 6 1 Valinta oli väärä. Tässä erikoistapaus. FIG . Yksi lauseista näyttää aika hassulta.

- Average texts from exercises-messages.txt:

2 9 14 1 Good! Notice that we can denote FIG and FIG . Now FIG is an instance of axiom (I) and FIG of axiom (II).

2 8 9 1 Note that our assumption FIG actually abbreviates FIG . Using two times Modus Ponens finishes the proof.

2 8 34 1 Your task is to write down the proof. A proof consists of a list of formulae which either are in the assumptions, are tautologies, identity or quantifier axioms or are formed from the previous formulae by Modus Ponens or Generalization rule.

2 8 34 1 Your task is to write down the proof. A proof consists of a list of formulae which either are in the assumptions, are tautologies, identity or quantifier axioms or are formed from the previous formulae by Modus Ponens or generalization rule.

2 8 32 1 The last sentence FIG does not always hold. If FIG is a linear order, then always either FIG or FIG , but the interpretation of FIG does not have to be either the least or the greatest element of FIG .

2 8 17 3 You will have to review basic integration formulae. Please review the lecture notes in the areas shown below and try the problem again later. Good luck!

- Short texts from exercises-messages.txt:

1 14 1 Good work! Now try to use this method to solve the original problem again.

1 13 1 Yksi lause ei ollut validi erikoistapauksessa, joten se ei voi olla yleisemmässäkään tapauksessa.

1 13 1 Todistuksessa olisi tarvittu aksioomaa (III) muodossa FIG , oletuksia ja Modus Ponens -päätelyä kahdesti.

1 13 1 The formula, which was not valid in the special case, can't be valid.

1 13 1 Oletusta voi käyttää muodossa FIG ja ja päätellä aksiooma (II):lla ja Modus Ponensilla.

1 13 1 Now we notice which sentence can't be valid in this more general case.

1 13 1 Muuttuja v_n ei esiinny vapaana etujäsenessä, eikä oletuksia ole, joten voimme käyttää yleistysääntöä.



1 12 1 Todistuksessa voisi esimerkiksi käyttää tautologiaa muotoa FIG, kvanttoriaksioomaa FIG ja tautologiaa FIG.
1 12 1 Alkuperäisen ongelman ratkaisua varten tarvitsemme äskeisen tuloksen ja sopivat FIG:n ja FIG:n.
1 11 3 Sorry! Please review the relevant lecture notes and try again later. Good luck!
1 10 1 Todistuksessa voi käyttää kvanttoriaksioomaa FIG, tautologiaa muotoa FIG ja yleistyssääntöä.
1 10 1 The same method can be used to solve this case.
1 10 1 Nyt varmaan osaat kirjoittaa aikaisemman todistuksen käyttämällä myös aksioomaa (I).

3.2.3 Plan A

As outlined at the WebALT kickoff meeting in Luxembourg, Plan A concentrates on short problems. The idea is to do semantically transparent, reversible constrained language parsing and generation from mathematically well-formed and interpreted input. Currently, we are studying the option that Plan A uses the Grammatical Framework (GF) parsing and generation formalism by Aarne Ranta (Ranta 2004).

The input would be OpenMath (Caprotti/Cohen (eds.) 2000), extended with problem-related vocabulary and possibly OpenMath attributes for NL grammar hints. Different generation variants would be generated from equivalent but different OpenMath formulae. OpenMath already has a small extension CCD for computer algebra and problem solving related vocabulary, which defines the verbs

decide, disprove, evaluate, evaluate_to_type, explore, find, look_up, prove, prove_in_theory, response.

See <http://www.riaca.win.tue.nl/cds/directives/directives1.ocd>.

There are typed versions of OpenMath which may be useful in this connection (Caprotti 1998, Strotmann 2004).

One alternative is that grammar hints would direct an inference engine to create an equivalent formula on the level of abstract syntax, from which the generator then generates the alternative output. The remaining NL generation would be basically syntax directed translation.

Here is an excellent illustration of the idea by Jordi Saludes, solving a sample generation task described by Andreas Strotmann (quoted with only minor editing).



[...] I've written a minimal grammar to deal with the example using the gf tool of Arne:

```
cat
  P ;          -- problem
  Expr ;      -- math expression
  Func ;      -- math function
  Symbol ;    -- math symbol
fun
  Let : (Symbol -> Expr) -> P -> P ;
  Calc : Expr -> P ;
  Simpl : Expr -> P ;
  Solve : Symbol -> Expr -> P ;
  Eval :  Func -> Expr -> Expr ;
  Deriv:  Func -> Func ;
  Inverse: Func -> Func ;
  Lambda: (Symbol -> Expr) -> Func ;

  f,g,h,x,y : Symbol ;
  one, x3 : Expr ;
```

Then we can code Andreas' example in its original form using the tree:

```
Calc (Eval (Deriv (Inverse g)) one)
```

and get an English linearization (gf output):

```
> l -lang=Eng Calc (Eval (Deriv (Inverse g)) one)
Calculate the value of the derivative of the inverse of g at 1
```

or in Catalan:

```
> l -lang=Cat Calc (Eval (Deriv (Inverse g)) one)
Calculeu el valor de la derivada de la inversa de g a 1
```

The same tree can also be linearized as a formula:

```
> l -lang=Math Calc (Eval (Deriv (Inverse g)) one)
Calculate ( ( g )^(-1) )' ( 1 )
```

1. Formulability

I find the NL ones more readable than the last one (at least for a student) and this raises the question: Should we stick to a tree encoding which determines the math/textual aspect of each part or (as in the tree above) or have it decided in a later stage of the processing?

Let's call a tree "formulable" iff it can be linearized as a math formula. A tree is formulable if its head and arguments are formulable. Then, following the grammar above, a tree with head "Calc" is not formulable, but



those with head "Deriv" or "Eval" are. Thus, some parse trees should have a "mode" for stating how they must be rendered: As NL or as math formula. The linearizations above are extreme cases of these two modes: the 1st and 2nd are NL renderings, but the 3rd is formula ("Calc" is not formulable, so the formula mode begins with the "Eval" subtree). Mixed renderings are possible (and perhaps more readable)

Calculate $((g)^{-1})'$ at 1

Calculate the derivative of g^{-1} at 1

A way to decide which to take may be to give each head a "formulability factor" and force subtrees to be rendered NL if the formulability factor of the head is below a given threshold. Then a high threshold will produce compact formula-like renderings and a low threshold will favor textual descriptions.

Regarding the example I think "Inverse" has a low formulability factor, because I find the $^{-1}$ notation cumbersome for the student. In fact, we could make the threshold increase as the student progresses.

What is at stake is problem reusability: The ability to make the system adaptable, not only to the language, but of teaching level.

2. Variants

Now for the 2 ways of generating NL in Andreas' example:

```
> 1 -lang=Eng Calc (Eval (Deriv (Inverse g)) one)
Calculate the value of the derivative of the inverse of g at 1      -- [1]
3 msec
> 1 -lang=Eng Let (\f->(Inverse g)) (Calc (Eval (Deriv f) one))
Calculate the value of the derivative of f at 1 , where f is the inverse of
g                                                                    -- [2]
```

This time a binding between f and (Inverse g) has been introduced in a "Let" tree.

Again I find the form [2] more readable. My hypothesis is that readability depends on the depth of the parse tree. The form [1] has a tree depth of 5, the second one has two trees of depth 3 and 4.

```
Bind f (Inverse g) = \f -> (Inverse g) . Depth = 3
Calc (Eval (Deriv f) one)) . Depth = 4
```

(I consider the "Let" root of the gf expression a kind of syntactic glue for building a single tree out of several, with no cognitive load on the reader)

Then the proposal will be to store form [1] as the canonical form and pre-process it before going to NLG:

(i) Flatten the tree by extraposition of high depth branches into a sequence of binding trees

We can combine this idea with the "formulability" idea and choose low formulable nodes as splitting points.



```
Calc (Eval (Deriv (Inverse g)) one) ; =>
```

```
{   Bind f (Inverse g) ;  
    Calc (Eval (Deriv f) one)) ;  
}
```

(ii). Reassemble the sequence into a parse tree using "Let" and pass it to the NLG

```
{   Bind f (Inverse g) ;  
    Calc (Eval (Deriv f) one)) ;  
}   =>
```

```
Let (\f -> (Inverse g)) (Calc (Eval (Deriv f) one)))
```

3.2.4 Evaluation of plan A

3.2.4.1 Maintainability

This covers maintainability of the WebALT technologies and tools and that of the WebALT components written with them.

GF itself is currently maintained by Aarne Ranta and his students as part of his and their research work. It has been or is currently being used in a number of research projects (most recently, the EU TALK dialogue management project). The newest version 2.1 was released on November 8, 2004, version 2.2 is announced as coming soon.

In WebALT we want to provide tools and techniques that are extendable and maintainable by non-language experts (the majority of existing NLG applications can only be modified and extended by language engineers). GF addresses this with its separation of application grammars and resource grammars (discussed below).

3.2.4.2 Accessibility

GF is open source under the GPL licence. Aarne Ranta has indicated interest in supporting the WebALT project in some capacity (though not as a paid associate).



3.2.4.3 Portability

The GF pipeline currently consists of an offline grammar compiler `gf` written in Haskell. The compiler produces a compiled grammar format. Such embedded grammar documents can be loaded and interpreted by runtime generators written in Haskell or Java (Bringert 2005).

A GF gramlet is a program automatically produced from a GF (Grammatical Framework) grammar. Given a GF grammar, the corresponding gramlet provides some of the functionality of GF — specialized for that grammar. There is a newer Java interpreter for compiled GF grammars (called embedded grammars) by Bringert (2005).

3.2.4.4 Scalability

Recently, GF grammar development has been split into *application grammars* and *resource grammars*. They relate to one another as abstract and concrete grammar, respectively. The resource grammar spells out domain independent language specific realisations. The domain grammar describes the language of the domain. The idea is that the application developer does not need to know about the linguistic detail of the resource grammars. The interface from the application grammar to different resource grammars can be shared.

3.2.4.5 Efficiency

The GF compiler takes as input a GF source grammar `*.gf` and produces offline a compiled grammar `*.gfc`. Depending on the size of the grammar, the compilation takes time from seconds to minutes. The size of the compiled grammar can be big, up to 10MB, so its initial load to the online generator takes appreciable time. This needs to be done once at the start of a generating session.

Since generation does largely deterministic syntax directed translation (called linearization in GF), it is fast (of the order of 10ms per text). Parsing is slower, currently of the order of 10-100 ms/sentence. [Aarne Ranta, p.c.]

3.2.5 Plan B

This plan is primarily intended to cope with text which is not fully interpreted mathematically, like some long problems and instructions. It does generation from annotated natural language input using XSL(C).

In this scenario, the input to generation would be tagged (partially parsed and disambiguated) NL input. Generation happens with a largely deterministic



pipeline using canned text and templates where possible. Such a generator is being developed at the University of Helsinki in a national dialogue management project (Fenix 4M) on the basis of work done by Graham Wilcock (Wilcock 2001,2002,2003,2003b).

Wilcock has developed general purpose ontology verbalisers for RDF and DAML+OIL and OWL. They are template-based and use a pipeline of XSLT transformations in order to produce text. The text structure follows closely the ontology constructs, e.g., "This is a description of John Smith identified by <http://...>His given name is John...". It is produced by performing sentence aggregation to connect sentences with the same subject and also referring expressions like "his" are used instead of repeating the person's name. The approach is a form of shallow generation, which is based on domain- and task-specific modules. (Bontcheva 2004:19).

It is too early to evaluate Plan B in detail by the WebALT quality criteria at this point in time.

4 Localization Issues for CCDs Implementations

WebALT Project Deliverable 1.1, which surveys e-learning technology, includes a thorough survey of standards for communicating course contents between such systems. We therefore limit our discussion in this chapter, to problems involved in localizing those international standards, focussing in particular on issues that deal with assessments and exercise materials. We also discuss technology that may be relevant for localizing the WebALT CCDs that aim to standardize the structure of the courses that the WebALT problem database aims to enhance.

4.1 Motivation

The CCD is meant to encode a structured table of contents of a typical course in mathematics, and would reasonably include terminology for concepts like "learning unit on the chain rule of differentiation". Clearly, to be useful for authors, teachers, and students alike, such terms will need to be rendered differently in different locales. This is a special case of the more general problem of maintaining multilingual classification hierarchies that is studied in the field of Library and Information Studies, with the Dewey Decimal Classification (DDC) as one such prime example of a successful standard. In addition to the localization issues for mathematical information that we have mentioned before, and in contrast to the way that multilingual classification hierarchies are maintained, a



CCD may be translated in the form of the concrete table of contents of whichever textbook is used in a particular course.

Another important class of concepts that will need to be rendered into different cultures rather than languages is may be exemplified by the case of partial-credit grading (perhaps using an A-D/F or 100-0 scale in the US, a 1-6 scale in Germany, or a 10-1 scale in Finland, for example). Translations of such culture-specific notions could be extremely tricky, but perhaps by using a standardized point scheme internally and offering the teacher the ability to assign a mapping from points to grades might help considerably when it comes to such translations.

Concepts that are provided with a standardized XML-based representation in e-learning communication standards, namely concepts such as "course", "exercise", "question", or "answer", will require localization in the WebALT system's user interfaces. A lot of work in this sense has probably been done already by those who have implemented the standards in e-learning platforms that are available in several languages already, and depending on the extent to which WebALT allows some of its components to be provided by such e-learning platforms, such localization issues may already have been taken care of.

4.2 Localization and XML

These problems are special cases of the general problem of localising content that is coded according to some XML application.

In principle, XML has been designed to enable localization, using three mechanisms.

- Defining Unicode as the fundamental XML character set, localization to any locale becomes possible within any XML document.
- The `xml:lang` attribute is globally available for any XML element, with the intended meaning that the contents of an XML element are written in the language specified as the value of this attribute. As an aside, no similar method is available for culture-specific localization, except as a sub-specifier of a language (e.g. en-US for US English).
- Using language-independent XML element names that can in principle, be rendered to any natural language on presentation to a user.



By providing a mechanism to branch code based on the language in which a particular XML element has been written, XSLT, the designated method for transforming XML application instances, now provides support for all three of these mechanisms. It is therefore possible to enhance the existing “universal MathML Content to Presentation stylesheets” that are already available in XSLT for use in web browsers that support MathML-Presentation, by adding multi-lingual and multi-cultural support to them.

5 Natural Language and Formal Mathematics

While not all of these topics are directly relevant to the WebALT project, it may be useful to provide an overview of some of the research that may have an impact on the project nevertheless.

5.1 Comprehensible Output from Provers

One of the most interesting areas of research tangential to the WebALT project, from a practical point of view, is being conducted in the automatic theorem proving area of symbolic mathematics.

Automatically generated proofs generally have the problem that they are extremely tedious, long-winded, and purely formal, making it hard to comprehend even for people closely involved in the underlying material. Incomprehensibility of the output produced by ATP systems severely limits its usefulness, and consequently, systems have been devised to help alleviate this problem.

The problems addressed here are quite similar to those generally found in NLG, with some extra twists: proofs need to be analysed for structure in order to avoid repetitions -- text planning is thus a major issue in this area; formal mathematics is partly rendered as (canned) “natural” text in order to improve legibility; and most of all, “irrelevant” details of the proof may be suppressed based on a (usually simple) model of user competence (e.g., no sense in proving that $5+3=8$).

As pointed out in previous chapters, there are severe limits as to the usability of research in this area within WebALT: WebALT does not need much text planning, because text structure is most usefully tagged in the semantic representation of an exercise during the editing process; unlike the pseudo-NLG in ATP systems, WebALT users (the students) will need good quality NL as output; and in



WebALT, exercises are meant to be encoded with exactly the level of detail that is to appear in NL form, thus requiring no abstracting facilities.

5.2 Formal Mathematics from Natural Language

In another research area that is closely related to ATP, the question is how to take a “natural language” proof (actually often called “mathematical vernacular”, meaning the typical mix of highly stylised natural language text fragments and two-dimensional mathematical formulae to be found in mathematical journals) and extract its semantics in enough detail that the proofs contained in such “natural” mathematical texts can be checked automatically.

The 2004 conference on Mathematical Knowledge Management, for example, had several papers on this problem, but this area has long been a focus of research in related fields. Nevertheless, as one particular paper title in that most recent conference in the field indicates quite clearly (“A path to faithful formalizations of mathematics”, Jojgov and Nederpelt 2004), this area is still very much at the research stage, with no practical solutions foreseeably available within the time frame of the project.

Several approaches have been documented in the literature:

- By devising a formal language with a surface appearance as close to mathematical vernacular as possible, content can be authored directly in pseudo-NL. This is the basic idea in the Mizar project, for example.
- Full-blown parsers for text fragments in mathematical vernacular are being attempted (Coen and Zacchiroli 2004), typically based on ideas in the area of “weak type theory”, an attempt at defining a simple categorial type theory for mathematical vernacular, based on which parsers attempt to disambiguate both the natural language and the presentation formula contents of a piece of mathematical vernacular. Since there appear to be strong ties of this approach to theories in formal semantics of natural language, this approach is quite promising, even if at such an early stage of research that it cannot possibly be used within the WebALT project.
- Somewhat in between these two approaches in terms of complexity and naturalness, another idea is to provide users with editing tools for tagging natural mathematical vernacular with enough semantic (weak) type information to allow a backend semantic analyser to produce a useful formal mathematical representation of its content, while leaving the



underlying NL text entirely untouched. This is the approach taken in the MathLang project, for example (Kamareddine et al. 2004).

All of these are based on generating formal mathematics in some kind of content markup combined with simple linguistic structure markup. Again, MathLang is such a markup language, and OMDoc (Kohlhase 2000) might qualify here, too. The WebALT project's Workpackage 2 needs to work out a formal language of this type, albeit probably considerably simpler on the one hand, and a lot more detailed on the other.

5.3 The Language of Mathematics in Teaching

In the field of education research, some groups have started studying mathematical formalisms that need to be taught to students as a form of almost-natural language. While this is not directly relevant to the WebALT project, this type of research may help to understand better the requirements for multilingual rendering of the kind of mathematical content we are interested in.

This type of research may also help to understand the role that WebALT technology may be able to play in alleviating problems in mathematics teaching in general (e.g. by providing tools for presenting mathematical content with varying degrees of formal mathematics) and in bilingual or multilingual mathematics teaching in particular (e.g. by providing access to key materials in a student's own language rather than that of the teacher).

As such, a survey of this area may help the project by improving its understanding of the requirements for its products, and to better disseminate the final product by identifying a wider market.

Conclusions

Although there is a wealth of previous work on NLG, there are few truly multilingual systems on the market ready to be directly exploited for multilingual generation of mathematics problems in WebALT.

By dividing the task into Plans A and B according to the character of the inputs we hope to best exploit the tools and expertise accessible to WebALT. For Plan A, semantics driven constrained language generation using GF seems a promising alternative. For Plan B, more surface oriented generation from marked up NL using XML based tools appears to be a good solution.



For presenting mathematical formulas that appear as fragments of generated text in a form that is appropriate to the same linguistic and cultural environment as the matrix sentences, it would be appropriate to both utilize and contribute to efforts by the MathML group to study the requirements for internationalising mathematical formulas. Since a browser-universal set of stylesheets already exist, and since ways of allowing individual stylistic choices within the output generated from such stylesheets are being studied, adding language-specific choices to these pieces of software may well be the most efficient way to solve this aspect of the WebALT project's task of generating multi-lingual and multi-cultural presentations from language-independent content.

One of the project partners, TUE, has developed a MathML Phrasebook that translates between OpenMath and MathML. This piece of software can be extended easily to cover any semantic concepts required within the project, if necessary.

Another partner group, UPC and Math4More, have developed an editor for OpenMath content that has a number of important features for this project:

- It allows restricting the range of mathematical concepts available based on the requirements of the application.
- It supports template-driven editing that is configurable at runtime, a convenient feature for multi-lingual input methods.

It also comes with configuration tools that are very useful as a basis for allowing teachers to adapt the tools to their students' needs by modifying the templates made available to the students.

REFERENCES

- Ayoubi, I.S. 2004. Writing Mathematics in Arabic. Rubicon Whitepaper, <http://www.rubicon.com.jo/papers/MathML-Arabic.pdf>.
- Bateman, J. 1997. Enabling technology for multilingual natural language generation: the KPML development environment. *Journal of Natural Language Engineering*, 3(1):15--55.
- Bontcheva, K. , A. Kiryakov, H. Cunningham, B. Popov, and M. Dimitrov 2003. Semantic web enabled, open source language technology. In *EACL workshop on Language Technology and the Semantic Web: NLP and XML*, Budapest, Hungary, 2003.
- Bontcheva, K. 2004. Natural Language Generation for Intelligent Knowledge Access: State of the Art Review. Technical Report CS-04-05, June 2004. Department of



- Computer Science, University of Sheffield, UK
<http://www.dcs.shef.ac.uk/research/resmes/papers/CS0405.pdf>
- Bontcheva, K. 2004b. Open-source Tools for Creation, Maintenance, and Storage of Lexical Resources for Language Generation from Ontologies. In Proceedings of 4th Language Resources and Evaluation Conference (LREC'04), 2004.
- Bringert, B. 2005. Embedded grammars. MSc Thesis, Chalmers, Göteborg, February 2005.
- Brun, C., Dymetman, M., Lux, V. 2000. Document structure and multilingual authoring . International Natural Language Generation Conference 12-16 June 2000, Mitzpe Ramon, Israel
- Busemann, S. 1996. Best-first surface realization. In Donia Scott, editor, Eighth International Natural Language Generation Workshop. Proceedings, pages 101–110, Herstmonceux, Univ. of Brighton, England. Also available at the Computation and Language Archive at <http://xxx.lanl.gov/abs/cmplg/9605010>.
- Busemann, S. 2004. eGram- a grammar development environment and its usage for language generation. in: *Proc. Fourth International Conference on Language Resources and Evaluation (LREC)*, Lisbon, Portugal, 2004.
- Caprotti, Olga, David Carlisle, Arjeh Cohen (ed.) 2000. The OpenMath Standard - Version 1.0. Report of the Esprit Project OpenMath, Feb. 2000.
<www.nag.co.uk/projects/OpenMath/omstd/>
- Caprotti, Olga, Arjeh M. Cohen 1998. A Type System for OpenMath. Report of the Esprit Project OpenMath, 1998. <www.nag.co.uk/projects/OpenMath/omstd/>
- Carlisle, D. 2002. MathML on the Web: Using XSLT to Enable Cross-Platform Support for XHTML and MathML in Current Browser
- Carlisle, D., Ion, P., Miner, R., and Poppelier, N. 2003. Mathematical Markup Language (MathML) Version 2.0 (Second Edition). W3C Recommendation 21 October 2003. Available at <http://www.w3.org/TR/2003/REC-MathML2-20031021/>.
- Cawsey, A. 2000. Presenting tailored resource descriptions: will XSLT do the job? In Proceedings of the Ninth World-Wide Web conference (WWW9).
- Coen, C.S., and Zacchiroli 2004. Efficient Ambiguous Parsing of Mathematical Formulae. Proceedings of the 3rd International Conference on Mathematical Knowledge Management MKM'2004, September 2004, Bialowieza, Poland.
- Eddahibi, M., Lazrek, A., and Sami, K. 2004. Arabic Mathematical e-Documents. In Apostolos Syropoulos, Karl Berry, Yannis Haralambous, et al., editors, International Conference on TeX, XML, and Digital Typography, Held Jointly with the 25th Annual Meeting of the TeX Users Group, TUG 2004. Proceedings, pages 158ff. Xanthi, Greece, August 30 - September 3, 2004.
- Evans. R., P. Piwek and L.J. Cahill, 2002. What is NLG? *Proceedings of INLG 2002*, New York, USA. (Also available as ITRI technical report ITRI-02-05)
- Elhadad, M. and Robin, J. 1996. An overview of SURGE: a reusable comprehensive syntactic realization component. Technical Report 96-03. Department of Computer Science, Ben Gurion University, Beer Sheva, Israel.



- Jojgov, G., and Nederpelt, R. 2004. A Path to Faithful Formalizations of Mathematics. Proceedings of the 3rd International Conference on Mathematical Knowledge Management MKM'2004, September 2004, Bialowieza, Poland.
- Kamareddine, F., Maarek, M., and Wells, J.B. 2004. Flexible Encoding of Mathematics on the Computer. Proceedings of the 3rd International Conference on Mathematical Knowledge Management MKM'2004, September 2004, Bialowieza, Poland.
- Kilger, Anne, Peter Poller, 2000. CDL-TAG: A grammar formalism for flexible and efficient syntactic generation. Proceedings of TAG+5, May 2000, Paris, France. Kohlhase, M. 2000. OMDoc: An infrastructure for OpenMath content dictionary information. SIGSAM Bulletin 34(2):43–48, June 2000.
- Lavoie, B., O. Rambow, and E. Reiter. 1996. The model-explainer. In Proceedings of the Eighth International Workshop on Natural Language Generation, pages 9-12, Herstmonceux, Sussex, UK.
- Ljunglöf, P. 2004. Expressivity and complexity of the Grammatical Framework. Ph.D.Thesis. Göteborg University, Göteborg, Sweden, November 2004.
- McKeown, K. 1985. Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text. Cambridge University Press, 1985.
- Mellish, C., D. Scott, L. Cahill, R. Evans, D. Paiva, and M. Reape. 2004. A Reference Architecture for Generation Systems. Natural Language Engineering, 2004.
- Naciri, H., and Rideau, L. 2002. Formal Mathematical Proof Explanations in Natural Language Using MathML: An Application to Proofs in Arabic. In Proceedings of MathML International Conference 2002. Available at <http://www.mathmlconference.org/2002/presentations/naciri/>.
- Raman, T.V. 1994. Audio System for Technical Readings, (PhD. Thesis, Cornell University; 1994; <http://www.cs.cornell.edu/home/raman>)
- Ranta, A. 1994. Type-Theoretical Grammar. Oxford University Press/Clarendon Press, 1994.
- Ranta, A. 2004. Grammatical Framework. Journal of Functional Programming Volume 14 , Issue 2 (March 2004) , 145 - 189
- RIACA 2004. MathML Phrasebook. Software available at <http://www.riaca.win.tue.nl/products/mathml/>
- Reape, M., Mellish, C. 1999. Just what is aggregation anyway? In Proceedings of the European Workshop on Natural Language Generation (EWNLG'99), pages 20 – 29, Toulouse, France, May 1999.
- Reiter, E., Dale, R. 2000. Building Natural Language Generation Systems. Cambridge University Press.
- Reitter, D. 2004. A development environment for multimodal functional unification generation grammars. In Proc. Third International Conference on Natural Language Generation (INLG04), 2004. <http://www.reitter-it-media.de//compling/mug/index.html>
- Scott, D., R. Power, Evans. 1998. Generation as a solution to its own problem. In Proceedings of the Ninth International Workshop on Natural Language Generation, pages 256--265, Niagara-on-the-Lake, Ontario, 1998.



- Sigurd, B. 1987. Referent Grammar. A Generalized Phrase Structure Grammar with built-in referents. *Studia Linguistica* 41:2,, 115-135
- Smirnova, E., and Watt, S.M. 2004. An Approach to Mathematical Notation Selection. In *Proceedings of the Second North American Workshop on Mathematical Knowledge Management, Phoenix, Arizona, USA, January 6, 2004*. Available at <http://imps.mcmaster.ca/na-mkm-2004/program.html>.
- Stenzhorn, H. 2003. XtraGen. A natural language generation system using Java and XML technologies. Master's thesis, Department for Computational Linguistics, University of the Saarland.
- Strotmann, A. 2004. The Categorial Type of OpenMath Objects. *Proceedings of the 3rd International Conference on Mathematical Knowledge Management MKM'2004, September 2004, Bialowieza, Poland*.
- Vincent, A. 2004. Internationalisation and Content MathML. In *Burning Chrome: a Mozillazine.org Weblog*. Available at <http://weblogs.mozillazine.org/weirdal/archives/005670.html>.
- Wilcock, G. 2001. Pipelines, Templates and Transformations: XML for Natural Language Generation. In *Proceedings of the First NLP and XML Workshop, NLPRS-2001, Tokyo*, pp. 1-8.
- Wilcock, G. et al. 2002. The Roles of Natural Language and XML in the Semantic Web. In Huang and Lenders (eds.) *Computational Linguistics and Beyond, Language and Linguistics Monograph Series I, Institute of Linguistics, Academia Sinica 2002*, 139-180. <http://www.ling.helsinki.fi/~gwilcock/Pubs/COLING-02-book.pdf>
- Wilcock, G. 2003. Integrating Natural Language Generation with XML Web Technology. 10th Conference of the European Chapter of the Association for Computational Linguistics, Budapest. In *EACL-2003, Proceedings of the Demo Sessions*, pp. 247-250.
- Wilcock, G. 2003b. XML-based Natural Language Generation (Course notes) <http://www.ling.helsinki.fi/~gwilcock/Tartu-2003/>