University Study Course
Text and Digital Media
020059 — Spring 2011

Michael Kohlhase & Thomas Rommel
Computer Science, Jacobs University
{m.kohlhase,t.rommel}@jacobs-university.de
MK office: Room 169, Research I, phone: x3140
TR office: Room 105, Research IV, phone: x3331

1
JACOBS UNIVERSITY

1 Preface

This document contains the course notes for the university study course "Text and Digital Media" held at Jacobs University Bremen in the the spring semester 2011 by Profs. Thomas Rommel and Michael Kohlhase.

This Document

1.1 This Document

Contents: The document mixes the slides presented in class with comments of the instructor to give students a more complete background reference.

Caveat: This document is made available for the students of this course only. It is still an early draft, and will develop over the course of the course. If the course is repeated, it will be developed further in coming academic years.

Licensing: Apart from this caveat, the course materials (slides, course notes, and problems) are licensed under a Creative Commons license that requires attribution, forbids commercial use, and allows derivative works as long as these are licensed under the same license.

Knowledge Representation Experiment: This document is also an experiment in knowledge representation. Under the hood, it uses the sTEX package [Koh08, Koh10], a TEX/LATEX extension for semantic markup, which allows to export the contents into the eLearning platform PantaRhei.

Other Resources: [1] [2] EdNote:1

Comments: Comments and extensions are always welcome, please send them to the author. EdNote:2

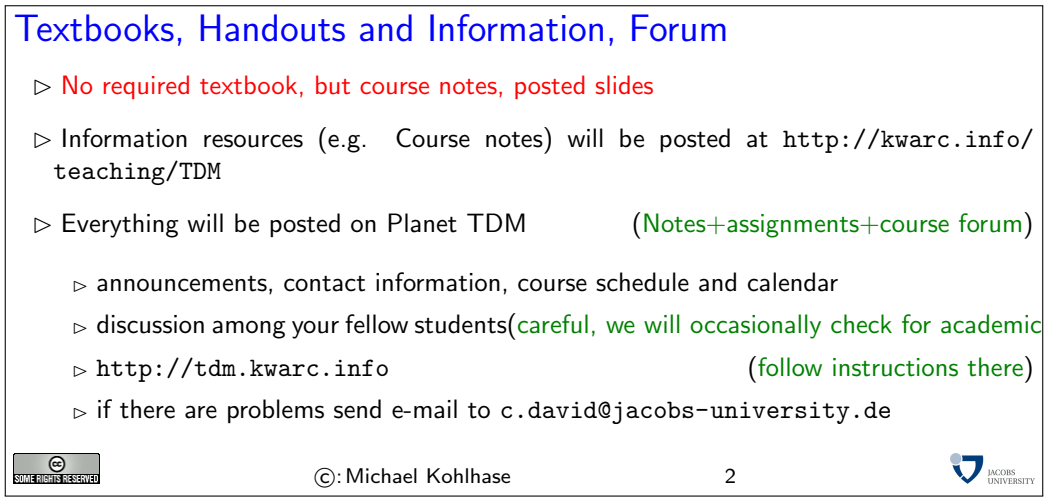
[1] EdNote: describe the discussions in Panta Rhei
[2] EdNote: Say something about the problems

Contents

2 Administrativa

We will now go through the ground rules for the course. This is a kind of a social contract between the instructors and the students. Both have to keep their side of the deal to make the acquaintance with issues about "text and digital media" as efficient and painless as possible.

2.1 Resources



No Textbook: Due to the special circumstances discussed above, there is no single textbook that covers the course. Instead we have a comprehensive set of course notes (this document). They are provided in two forms: as a large PDF that is posted at the course web page and on the Planet TDM system. The latter is actually the preferred method of interaction with the course materials, since it allows to discuss the material in place, to play with notations, to give feedback, etc. The PDF file is for printing and as a fallback, if the Planet TDM system, which is still under development develops problems.

Next we come to a special project that is going on in parallel to teaching the course. I am using the coures materials as a research object as well. This gives you an additional resource, but may affect the shape of the coures materials (which now server double purpose). Of course I can use all the help on the research project I can get.

Experiment: E-Learning with OMDoc/PantaRhei

- ▷ My research area: deep representation formats for (mathematical) knowledge
- ▷ Application: E-learning systems (represent knowledge to transport it)
- ▷ Experiment: Start with this course (Drink my own medicine)
  - ▹ Re-Represent the slide materials in OMDoc (Open Math Documents)
  - ▹ Feed it into the PantaRhei system (`http://trac.mathweb.org/planetary`)
  - ▹ Try it on you all (to get feedback from you)
- ▷ Tasks (Unfortunately, I cannot pay you for this; maybe later)
  - ▹ help me complete the material on the slides (what is missing/would help?)
  - ▹ I need to remember "what I say", examples on the board. (take notes)
- ▷ Benefits for you (so why should you help?)
  - ▹ you will be mentioned in the acknowledgements (for all that is worth)
  - ▹ you will help build better course materials (think of next-year's freshmen)

2.2 Grades

Now we come to a topic that is always interesting to the students: the grading scheme. The grading scheme I am using has changed over time, but I am quite happy with it.

Prerequisites, Requirements, Grades

- ▷ Prerequisites: Motivation, Interest, Curiosity, hard work
  - ▹ you can do this course if you want!
- ▷ Grades: The final grade will entirely be based on weekly homework assignments
- ▷ TDM Teams: Homeworks will be solved and submitted in teams of three (two from SES, one from SHSS), which will be formed for the course in the beginning.
- ▷ Rationale: We want to have knowledge transfer (between the disciplines.)

2.3 Homeworks, Submission, and Cheating

Homework assignments

- ▷ Goal: Reinforce and apply what is taught/discussed in class.
- ▷ homeworks: will be practical writing assgignmenst in a variety of geners and formats (take time to solve)
- ▷ admin: To keep things running smoothly
  - ▹ Homeworks will be posted on PantaRhei
  - ▹ Homeworks are handed in electronically in `grader` (plain text, Postscript, PDF,...)
  - ▹ discuss problems on PantaRhei (Profs/TAs/students can help you!)
- ▷ Homework discipline:
  - ▹ start early! (many assignments need more than one evening's work)
  - ▹ Don't start by sitting at a blank screen
  - ▹ Humans will be trying to understand the text/code/math when grading it.

Homework assignments are a central part of the course, they allow you to review the concepts covered in class, and practice using them.

Homework Submissions, Grading, Tutorials

- ▷ Submissions: We use Heinrich Stamerjohanns' `grader` system
  - ▹ submit all homework assignments electronically to `https://jgrader.de`
  - ▹ you can login with you Jacobs account (should have one!)
  - ▹ feedback/grades to your submissions
  - ▹ get an overview over how you are doing! (do not leave to midterm)
- ▷ Tutorials: select a tutorial group and actually go to it regularly
  - ▹ to discuss the course topics after class (GenCS needs pre/postparation)
  - ▹ to discuss your homework after submission (to see what was the problem)
  - ▹ to find a study group (probably the most determining factor of success)

The next topic is very important, you should take this very seriously, even it you think that this is just a self-serving regulation made by the faculty.

All societies have their rules, written and unwritten ones, which serve as a social contract among its members, protect their interestes, and optimize the functioning of the society as a whole. This is also true for the community of scientists worldwide. This society is special, since it balances intense cooperation on joint issues with fierce competition. Most of the rules are largely unwritten; you are expected to follow them anyway. The code of academic integrity at Jacobs is an attempt to put some of the aspects into writing.

It is an essential part of your academic education that you learn to behave like academics, i.e. to function as a member of the academic community. Even if you do not want to become a scientist in the end, you should be aware that many of the people you are dealing with have

gone through an academic education and expect that you (as a graduate of Jacobs) will behave by these rules.

The Code of Academic Integrity

- ▷ Jacobs has a "Code of Academic Integrity"
  - ▷ this is a document passed by the faculty (our law of the university)
  - ▷ you have signed it last week (we take this seriously)
- ▷ It mandates good behavior and penalizes bad from both faculty and students
  - ▷ honest academic behavior (we don't cheat)
  - ▷ respect and protect the intellectual property of others (no plagiarism)
  - ▷ treat all Jacobs members equally (no favoritism)
- ▷ this is to protect you and build an atmosphere of mutual respect
  - ▷ academic societies thrive on reputation and respect as primary currency
- ▷ The Reasonable Person Principle (one lubricant of academia)
  - ▷ we treat each other as reasonable persons
  - ▷ the other's requests and needs are reasonable until proven otherwise

To understand the rules of academic societies it is central to realize that these communities are driven by economic considerations of their members. However, in academic societies, the the primary good that is produced and consumed consists in ideas and knowledge, and the primary currency involved is academic reputation[1]. Even though academic societies may seem as altruistic — scientists share their knowledge freely, even investing time to help their peers understand the concepts more deeply — it is useful to realize that this behavior is just one half of an economic transaction. By publishing their ideas and results, scientists sell their goods for reputation. Of course, this can only work if ideas and facts are attributed to their original creators (who gain reputation by being cited). You will see that scientists can become quite fierce and downright nasty when confronted with behavior that does not respect other's intellectual property.

One special case of academic rules that affects students is the question of cheating, which we will cover next.

[1] Of course, this is a very simplistic attempt to explain academic societies, and there are many other factors at work there. For instance, it is possible to convert reputation into money: if you are a famous scientist, you may get a well-paying job at a good university,...

Cheating [adapted from CMU:15-211 (P. Lee, 2003)]

- ▷ There is no need to cheat in this course!! (hard work will do)
- ▷ cheating prevents you from learning (you are cutting your own flesh)
- ▷ if you are in trouble, come and talk to me (I am here to help you)
- ▷ We expect you to know what is useful collaboration and what is cheating
  - ▹ you will be required to hand in your own original code/text/math for all assignments
  - ▹ you may discuss your homework assignments with others, but if doing so impairs your ability to write truly original code/text/math, you will be cheating
  - ▹ copying from peers, books or the Internet is plagiarism unless properly attributed (even if you change most of the actual words)
  - ▹ more on this as the semester goes on . . .
- ▷ * There are data mining tools that monitor the originality of text/code. *

We are fully aware that the border between cheating and useful and legitimate collaboration is difficult to find and will depend on the special case. Therefore it is very difficult to put this into firm rules. We expect you to develop a firm intuition about behavior with integrity over the course of stay at Jacobs.

2.4 Resources

Textbooks, Handouts and Information, Forum

- ▷ No required textbook, but course notes, posted slides
- ▷ Information resources (e.g. Course notes) will be posted at `http://kwarc.info/teaching/TDM`
- ▷ Everything will be posted on Planet TDM (Notes+assignments+course forum)
  - ▹ announcements, contact information, course schedule and calendar
  - ▹ discussion among your fellow students(careful, we will occasionally check for academic integrity!)
  - ▹ `http://tdm.kwarc.info` (follow instructions there)
  - ▹ if there are problems send e-mail to `c.david@jacobs-university.de`

No Textbook: Due to the special circumstances discussed above, there is no single textbook that covers the course. Instead we have a comprehensive set of course notes (this document). They are provided in two forms: as a large PDF that is posted at the course web page and on the Planet TDM system. The latter is actually the preferred method of interaction with the course materials, since it allows to discuss the material in place, to play with notations, to give feedback, etc. The PDF file is for printing and as a fallback, if the Planet TDM system, which is still under development develops problems.

Software/Hardware tools

- ▷ You will need computer access for this course (come see me if you do not have a computer of your own)
- ▷ we recommend the use of standard software tools
  - ▷ the `emacs` and `vi` text editor (powerful, flexible, available, free)
  - ▷ `UNIX` (`linux`, `MacOSX`, `cygwin`) (prevalent in CS)
  - ▷ `FireFox` (just a better browser (for Math))
- ▷ learn how to touch-type NOW (reap the benefits earlier, not later)

Touch-typing: You should not underestimate the amount of time you will spend typing during your studies. Even if you consider yourself fluent in two-finger typing, touch-typing will give you a factor two in speed. This ability will save you at least half an hour per day, once you master it. Which can make a crucial difference in your success.

Touch-typing is very easy to learn, if you practice about an hour a day for a week, you will re-gain your two-finger speed and from then on start saving time. There are various free typing tutors on the network. At `http://typingsoft.com/all_typing_tutors.htm` you can find about programs, most for windows, some for linux. I would probably try `Ktouch` or `TuxType`

Darko Pesikan recommends the `TypingMaster` program. You can download a demo version from `http://www.typingmaster.com/index.asp?go=tutordemo`

You can find more information by googling something like "learn to touch-type". (goto `http://www.google.com` and type these search terms).

Next we come to a special project that is going on in parallel to teaching the course. I am using the coures materials as a research object as well. This gives you an additional resource, but may affect the shape of the coures materials (which now server double purpose). Of course I can use all the help on the research project I can get.

Experiment: E-Learning with OMDoc/PantaRhei

- ▷ My research area: deep representation formats for (mathematical) knowledge
- ▷ Application: E-learning systems (represent knowledge to transport it)
- ▷ Experiment: Start with this course (Drink my own medicine)
  - ▷ Re-Represent the slide materials in OMDoc (Open Math Documents)
  - ▷ Feed it into the PantaRhei system (`http://trac.mathweb.org/planetary`)
  - ▷ Try it on you all (to get feedback from you)
- ▷ Tasks (Unfortunately, I cannot pay you for this; maybe later)
  - ▷ help me complete the material on the slides (what is missing/would help?)
  - ▷ I need to remember "what I say", examples on the board. (take notes)
- ▷ Benefits for you (so why should you help?)
  - ▷ you will be mentioned in the acknowledgements (for all that is worth)
  - ▷ you will help build better course materials (think of next-year's freshmen)

3 Documents as Digital Objects

Documents as Digital Objects

- ▷ Question: how do texts get onto the computer? (after all, computers can only do 0/1)
- ▷ Hint: At the most basic level, texts are just sequences of characters.
- ▷ Answer: We have to encode characters as sequences of bits.
- ▷ We will not go into how sequences of bits are stored on a hard disc or in memory of a computer here.

Before we go on, let us first get into some basics: how do we measure information, and how does this relate to units of information we know.

Units of Information

Bit (b)	*binary digit 0/1*
Byte (B)	*8 bit*
2 Bytes	A Unicode character.
10 Bytes	your name.
Kilobyte (KB)	*1,000 bytes OR* 10^3 *bytes*
2 Kilobytes	A Typewritten page.
100 Kilobytes	A low-resolution photograph.
Megabyte (MB)	*1,000,000 bytes OR* 10^6 *bytes*
1 Megabyte	A small novel OR a 3.5 inch floppy disk.
2 Megabytes	A high-resolution photograph.
5 Megabytes	The complete works of Shakespeare.
10 Megabytes	A minute of high-fidelity sound.
100 Megabytes	1 meter of shelved books.
500 Megabytes	A CD-ROM.
Gigabyte (GB)	*1,000,000,000 bytes or* 10^9 *bytes*
1 Gigabyte	a pickup truck filled with books.
20 Gigabytes	A good collection of the works of Beethoven.
100 Gigabytes	A library floor of academic journals.
Terabyte (TB)	*1,000,000,000,000 bytes or* 10^{12} *bytes*
1 Terabyte	50000 trees made into paper and printed.
2 Terabytes	An academic research library.
10 Terabytes	The print collections of the U.S. Library of Congress.
400 Terabytes	National Climactic Data Center (NOAA) database.
Petabyte (PB)	*1,000,000,000,000,000 bytes or* 10^{15} *bytes*
1 Petabyte	3 years of EOS data (2001).
2 Petabytes	All U.S. academic research libraries.
20 Petabytes	Production of hard-disk drives in 1995.
200 Petabytes	All printed material (ever).
Exabyte (EB)	*1,000,000,000,000,000,000 bytes or* 10^{18} *bytes*
2 Exabytes	Total volume of information generated in 1999.
5 Exabytes	All words ever spoken by human beings ever.
300 Exabytes	All data stored digitally in 2007.
Zettabyte (EB)	*1,000,000,000,000,000,000,000 bytes or* 10^{21} *bytes*
2 Zettabytes	Total volume digital data transmitted in 2011
100 Zettabytes	Data equivalent to the human Genome in one body.

The information in this table is compiled from various studies, most recently [HL11].

3.1 Character Encodings

Now we can come back to the question of how characters (and thus texts) can be encoded. Actually, this is a rather interesting story, once we realize the history and scope of such encodings.

The `ASCII` code we will introduce here is one of the first standardized and widely used character encodings for a complete alphabet. It is still widely used today. The code tries to strike a balance between a being able to encode a large set of characters and the representational capabiligies in the time of punch cards (cardboard cards that represented sequences of binary numbers by

rectangular arrays of dots).[3] EdNote:3

The ASCII Character Code

▷ **Definition 1** The American Standard Code for Information Interchange (ASCII) code assigns characters to numbers 0-127

Code	···0	···1	···2	···3	···4	···5	···6	···7	···8	···9	···A	···B	···C	···D	···E	···F
0···	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1···	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2···	␣	!	”	#	$	%	&	’	(	)	*	+	,	−	.	/
3···	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4···	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5···	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6···	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7···	p	q	r	s	t	u	v	w	x	y	z	{	|	}	∼	DEL

The first 32 characters are control characters for ASCII devices like printers

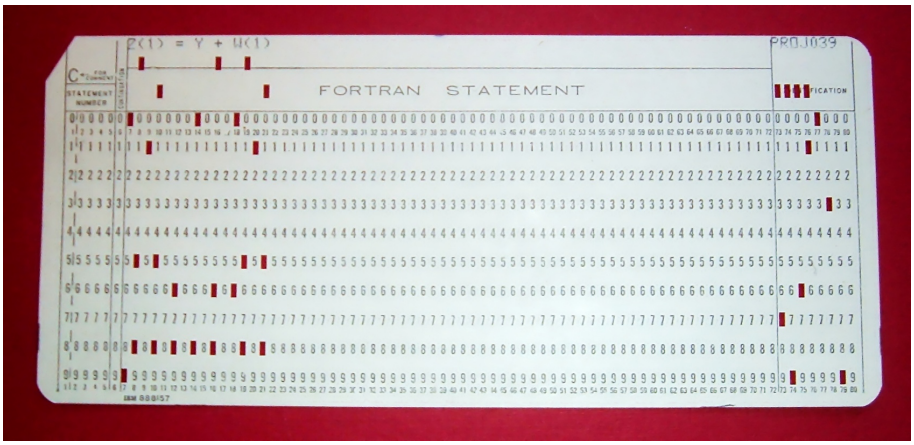
⋙ Motivated by punchcards: The character 0 (binary 000000) carries no information NUL, (used as dividers)
Character 127 (binary 1111111) can be used for deleting (overwriting) last value (cannot delete holes)

▷ The ASCII code was standardized in 1963 and is still prevalent in computers today (but seen as US-centric)

A Punchcard

▷ A punch card is a piece of stiff paper that contains digital information represented by the presence or absence of holes in predefined positions.

▷ **Example 2** This punch card encoded the `Fortran` statement `Z(1) = Y + W(1)`



The `ASCII` code as above has a variety of problems, for instance that the control characters are mostly no longer in use, the code is lacking many characters of languages other than the English

[3] EdNote: is the 7-bit grouping really motivated by the cognitive limit?

language it was developed for, and finally, it only uses seven bits, where a byte (eight bits) is the preferred unit in information technology. Therefore there have been a whole zoo of extensions, which — due to the fact that there were so many of them — never quite solved the encoding problem.

Problems with ASCII encoding

- ▷ Problem: Many of the control characters are obsolete by now (e.g. NUL,BEL, or DEL)
- ▷ Problem: Many European characters are not represented (e.g. è,ñ,ü,ß,...)
- ▷ European ASCII Variants: Exchange less-used characters for national ones
- ▷ **Example 3 (German ASCII)** remap e.g. [↦ Ä,] ↦ Ü in German ASCII ("Apple][" comes out as "Apple ÜÄ")
- ▷ **Definition 4 (ISO-Latin (ISO/IEC 8859))** 16 Extensions of ASCII to 8-bit (256 characters) ISO-Latin 1 ≙ "Western European", ISO-Latin 6 ≙ "Arabic",ISO-Latin 7 ≙ "Greek"...
- ▷ Problem: No cursive Arabic, Asian, African, Old Icelandic Runes, Math,...
- ▷ Idea: Do something totally different to include all the world's scripts: For a scalable architecture, separate
  - ▷ what characters are available from the (character set)
  - ▷ bit string-to-character mapping (character encoding)

The goal of the UniCode standard is to cover all the worlds scripts (past, present, and future) and provide efficient encodings for them. The only scripts in regular use that are currently excluded are fictional scripts like the elvish scripts from the Lord of the Rings or Klingon scripts from the Star Trek series.

An important idea behind UniCode is to separate concerns between standardizing the character set — i.e. the set of encodable characters and the encoding itself.

Unicode and the Universal Character Set

- ▷ **Definition 5 (Twin Standards)** A scalable Architecture for representing all the worlds scripts
  - ▷ The Universal Character Set defined by the ISO/IEC 10646 International Standard, is a standard set of characters upon which many character encodings are based.
  - ▷ The Unicode Standard defines a set of standard character encodings, rules for normalization, decomposition, collation, rendering and bidirectional display order
- ▷ **Definition 6** Each UCS character is identified by an unambiguous name and an integer number called its code point.
- ▷ The UCS has 1.1 million code points and nearly 100 000 characters.
- ▷ **Definition 7** Most (non-Chinese) characters have code points in $[1, 65536]$ (the basic multilingual plane).
- ▷ **Notation 8** For code points in the Basic Multilingual Plane (BMP), four digits are used, e.g. U+0058 for the character `LATIN CAPITAL LETTER X`;

Note that there is indeed an issue with space-efficient encoding here. UniCode reserves space for 2^{32} (more than a million) characters to be able to handle future scripts. But just simply using 32 bits for every UniCode character would be extremely wasteful: UniCode-encoded versions of `ASCII` files would be four times as large.

Therefore UniCode allows multiple encodings. `UTF-32` is a simple 32-bit code that directly uses the code points in binary form. `UTF-8` is optimized for western languages and coincides with the `ASCII` where they overlap. As a consequence, `ASCII` encoded texts can be decoded in `UTF-8` without changes — but in the `UTF-8` encoding, we can also address all other UniCode characters (using multi-byte characters).

Character Encodings in Unicode

▷ **Definition 9** A character encoding is a mapping from bit strings to UCS code points.

▷ Idea: Unicode supports multiple encodings (but not character sets) for efficiency

▷ **Definition 10 (Unicode Transformation Format)** ▷ `UTF-8`, 8-bit, variable-width encoding, which maximizes compatibility with ASCII.

▷ `UTF-16`, 16-bit, variable-width encoding (popular in Asia)

▷ `UTF-32`, a 32-bit, fixed-width encoding (for safety)

▷ **Definition 11** The `UTF-8` encoding follows the following encoding scheme

Unicode	Byte1	Byte2	Byte3	Byte4
U+000000 – U+00007F	0xxxxxxx			
U+000080 – U+0007FF	110xxxxx	10xxxxxx		
U+000800 – U+00$FFFF$	1110xxxx	10xxxxxx	10xxxxxx	
U+010000 – U+10$FFFF$	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

▷ **Example 12** $ = U+0024 is encoded as 00100100 (1 byte)

¢ = U+00$A2$ is encoded as 11000010,10100010 (two bytes)

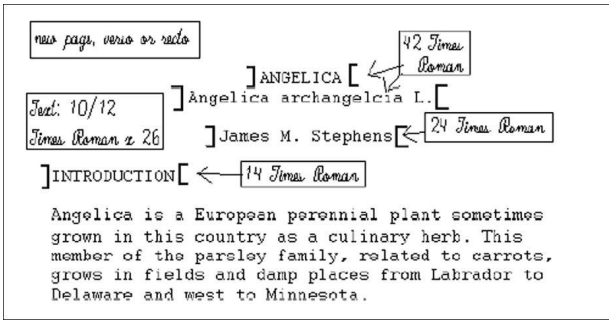
e = U+20AC is encoded as 11100010,10000010,10101100 (three bytes)

Note how the fixed bit prefixes in the encoding are engineered to determine which of the four cases apply, so that `UTF-8` encoded documents can be safely decoded..

3.2 Texts are more than Sequences of Characters

Document Markup

▷ **Definition 13 (Document Markup)** Document markup is the process of adding codes to a document to identify the structure of a document or the format in which it is to appear.



Styles of Markup

▷ **Definition 14 (Presentation Markup)** A presentation markup scheme is one that specifies document structure to aid document processing by humans

▷ **Example 15** e.g. *roff, Postscript, DVI, early MS Word, low-level TEX

\+ simple, context-free, portable (verbatim), easy to implement/transform

– inflexible, possibly verbose,

▷ **Definition 16 (Content Markup)** A content markup scheme is one that specifies document structure to aid document processing by machines or with machine support.

▷ **Example 17** e.g. LATEX (if used correctly), Programming Languages, ATP input

\+ flexible, portable (in spirit), unambiguous, language-independent

– possibly verbose, context dependent, hard to read and write

Content vs. Presentation by Example

Format	Representation	Content?
LATEX	`{\textbf{proof}}:...\hfill\Box`	`\begin{proof}...\end{proof}`
HTML	`<font size=+2><b>...</b></font>`	`<h1>...</h1>`
Lisp	$8 + \sqrt{x^3}$	`(power (plus 8 (sqrt x)) 3)`
TEX	`$\{f\|f(0)> 0{\rm and}f(1)<0\}$`	$\{f\|f(0) > 0 \text{ and } f(1) < 0\}$
TEX	`$\{f\|f(0)> 0$ and $f(1)<0\}$`	$\{f\|f(0) > 0 \text{ and } f(1) < 0\}$

▷ We consider these to be representations of the same content (object)

▷ Problem: Transformations between presentation and content Markup

▷ Content $\rightsquigarrow$ Pres.: usually done by styling (++ user-adaptivity)

▷ Pres. $\rightsquigarrow$ Content: Heuristic Process (e.g. binomials $\binom{n}{k}$ vs. C_k^n vs. C_n^k)

Content vs. Semantics/Formalization

▷ Content: logic-independent infrastructure
Identification of abstract syntax, "semantics" by reference for symbols.

```
<apply>
  <plus/>
  <csymbol definitionURL="mbase://numbers/perfect#the-smallest"/>
  <cn>2</cn>
</apply>
```

▷ Semantics: establishing meaning by fixing consequences
adds formal inference rules and axioms.

▷ Mechanization in a specific system (Thm Prover or Proof Checker)

▷ logical framework (specify the logic in the system itself)

4 On the Meaning of Texts (Natural Language)

Fascination of Language

▷ Even more so than thinking, language is a skill that only humans have.

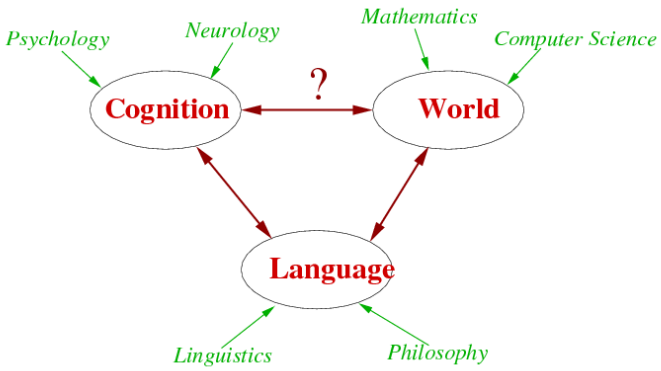
▷ It is a miracle that we can express complex thoughts in a sentence in a matter of seconds.

▷ It is no less miracuous that a child can learn tens of thousands of words and a complex grammar in a matter of a few years.

The study of natural language (and of course its meaning) is more complex than natural sciences, where we only observe objects that exist independently of ourselves as observers. Language is an inherently human activity, and deeply interdependent with human cognition (it is arguably one of its motors and means of expression). On the other hand, language is used to communicate about phenomena in the world around us, the world in us, and about hypothetical worlds we only imagine.

Therefore, natural language semantics must necessarily be an intersective discipiline and a trans-disciplinary endeavor, combining methods, results and insights from various disciplines.

NL Semantics as an Intersective Discipline

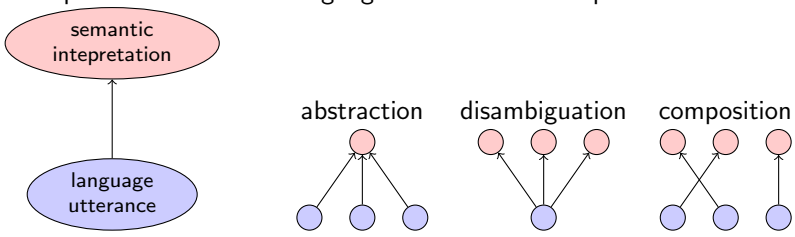


Language Technoloty

- ▷ Language Assistance
  - ▷ written language: Spell-/grammar-/style-checking
  - ▷ spoken language: dictation systems and screen readers
  - ▷ multilingual text: machine-supported text and dialog translation, eLearning
- ▷ Dialog Systems
  - ▷ Information Systems: at airport, tele-banking, e-commerce, call centers
  - ▷ Dialog interfaces for computers, robots, cars
- ▷ Information management:
  - ▷ search and classification of documetns
  - ▷ information extraction, question answering.

Language and Information

- ▷ humans use words (sentences, texts) in natural languages to represent information
- ▷ but:
- ▷ what really counts is not the words themselves, but the meaning information they carry.
- ▷ for questions/answers, it would be very useful to find out what words (sentences/texts) mean.
- ▷ Interpretation of natural language utterances: three problems



Fun with Diamonds (are they real?) [Dav67]

- ▷ *This is a blue diamond* ($\models diamond, \models blue$)
- ▷ *This is a big diamond* ($\models diamond, \not\models big$)
- ▷ *This is a fake diamond* ($\not\models diamond$)
- ▷ *This is a fake blue diamond* ($\models blue?, \models diamond?$)
- ▷ *Mary knows that this is a diamond* ($\models diamond$)
- ▷ *Mary believes that this is a diamond* ($\not\models diamond$)

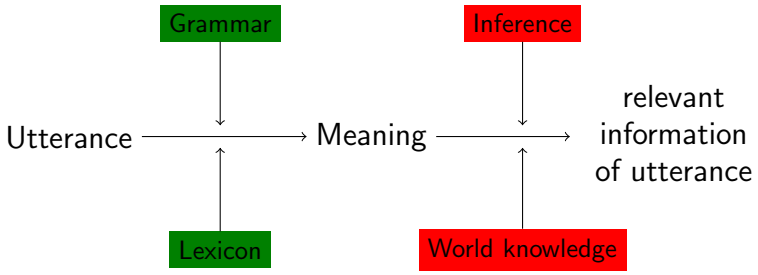
Logical analysis vs. conceptual analysis: These examples — Mostly borrowed from [Dav67] — help us to see the difference between logical analysis and conceptual analysis. We observed that from *This is a big diamond.* we cannot conclude *This is big*. Now consider the sentence *Jane is a beautiful dancer*. Similarly, it does not follow from this that Jane is beautiful, but only that she dances beautifully. Now, what it is to be beautiful or to be a beautiful dancer is a complicated matter. To say what these things are is a problem of conceptual analysis. The job of semantics is to uncover the logical form of these sentences. Semantics should tell us that the two sentences have the same logical forms; and ensure that these logical forms make the right predictions about the entailments and truth conditions of the sentences, specifically, that they don't entail that the object is big or that Jane is beautiful. But our semantics should provide a distinct logical form for sentences of the type: *This is a fake diamond.* From which it follows that the thing is fake, but not that it is a diamond.

Ambiguity (It could mean more than one thing)

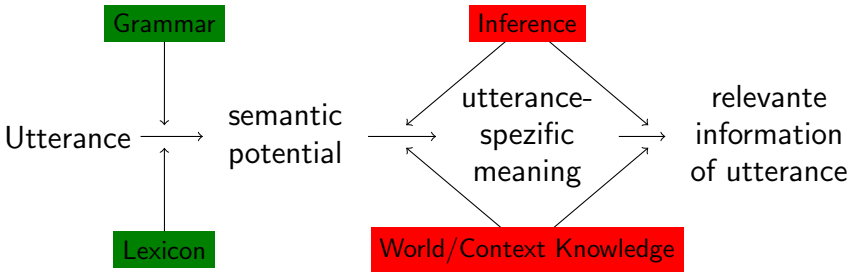
- ▷ *John went to the bank* (river or financial?)
- ▷ *You should have seen the bull we got from the pope* (three-way!)
- ▷ *I saw her duck* (animal or action?)
- ▷ *John chased the gangster in the red sports car* (three-way too!)

One way to think about the examples of ambiguity on the previous slide is that they illustrate a certain kind of indeterminacy in sentence meaning. But really what is indeterminate here is what sentence is represented by the physical realization (the written sentence or the phonetic string). The symbol *duck* just happens to be associated with two different things, the noun and the verb. Figuring out how to interpret the sentence is a matter of deciding which item to select. Similarly for the syntactic ambiguity represented by PP attachment. Once you, as interpreter, have selected one of the options, the interpretation is actually fixed. (This doesn't mean, by the way, that as an interpreter you necessarily do select a particular one of the options, just that you can.)

▷ *The lecture begins at 11:00 am .*



▷ *it starts at eleven.*



Semantics is not a Cure-It-All!

How many animals of each species did Moses take onto the ark?

▷ Actually, it was Noah (But you understood the question anyways)

▷ The only thing that currently really helps is a restricted domain

- ▷ restricted vocabulary
- ▷ restricted world model

▷ Demo: Bahnauskunft unter 0241-604020

▷ Demo: DBPedia `http://wikipedia.aksw.org/`

5 Basics Concepts of the World Wide Web

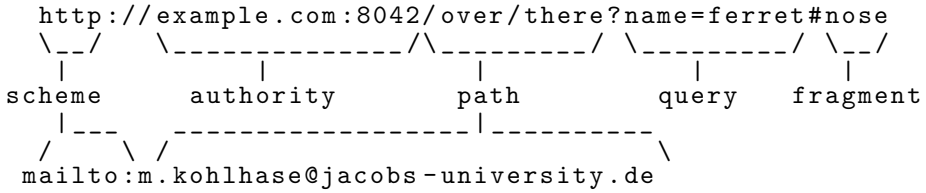
The world wide web is a service on the Internet based on specific protocols and markup formats for documents.

Uniform Resource Identifier (URI), Plumbing of the Web

▷ **Definition 18** A uniform resource identifier is a global identifiers of network-retrievable documents (web resources). URIs adhere a uniform syntax (grammar) defined in RFC-3986 [BLFM05]. Rules contain: URI :== scheme, ':', hierPart, ['?' query], ['#' fragment]
hier − part :== '//' (pathAbempty | pathAbsolute | pathRootless | pathEmpty)

▷ **Example 19** The following are two example URIs and their component parts:

▷



Note: URIs only identify documents, they do not have to be provide access to them (e.g. in a browser).

Uniform Resource Locators and relative URIs

▷ **Definition 20** A uniform resource locator is a URI that that gives access to a web resource via the `http` protocol.

▷ **Example 21** The following URI is a URL (try it in your browser)

`http://kwarc.info/kohlhase/index.html`

▷ Note: URI/URLs are one of the core features of the web infrastructure, they are considered to be the plumbing of the WWWeb. (direct the flow of data)

▷ **Definition 22** URIs can be abbreviated to relative URIs; missing parts are filled in from the context

▷ **Example 23**

relative URI	abbreviates	in context
`#foo`	⟪*current_file*⟫`#foo`	curent file
`../bar.txt`	`file:///home/kohlhase/foo/bar.txt`	file system
`../bar.html`	`http://example.org/foo/bar.html`	on the web

Web Browsers

▷ **Definition 24** A web Browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web, enabling users to view Web pages and to jump from one page to another.

▷ Practical Browser Tools:

- ▷ Status Bar: security info, page load progress
- ▷ Favorites (bookmarks)
- ▷ View Source: view the code of a Web page
- ▷ Tools/Internet Options, history, temporary Internet files, home page, auto complete, security settings, programs, etc.

▷ **Example 25** e.g. IE, Mozilla Firefox, Safari, etc.

▷ **Definition 26** A web page is a document on the Web that can include multimedia data

▷ **Definition 27** A web site is a collection of related Web pages usually designed or controlled by the same individual or company.

▷ a web site generally shares a common domain name.

HTTP: Hypertext Transfer Protocol

▷ **Definition 28** The Hypertext Transfer Protocol (HTTP) is an application layer protocol for distributed, collaborative, hypermedia information systems.

▷ June 1999: HTTP/1.1 is defined in RFC 2616 [FGM+99]

Definition 29 HTTP is used by a client (called user agent) to access web resources (addressed by Uniform Resource Locators (URLs)) via a http request. The web server answers by supplying the resource

▷ Most important HTTP requests (5 more less prominent)

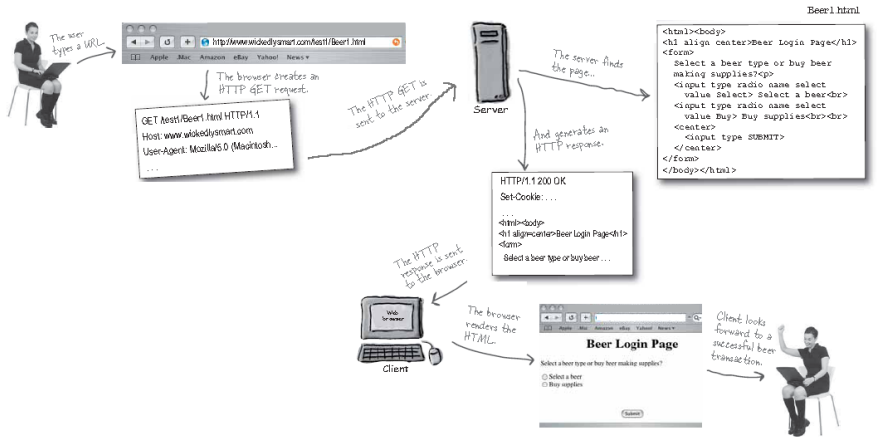
GET	Requests a representation of the specified resource.	safe
PUT	Uploads a representation of the specified resource.	idempotent
DELETE	Deletes the specified resource.	idempotent
POST	Submits data to be processed (e.g., from a web form) to the identified resource.	

▷ **Definition 30** We call a HTTP request safe, iff it does not change the state in the web server. (except for server logs, counters,. . . ; no side effects)

▷ **Definition 31** We call a HTTP request idempotent, iff executing it twice has the same effect as executing it once.

▷ HTTP is a stateless protocol (very memory-efficient for the server.)

Overview: A http request in the browser



Example: An http request in real life

▷ Connect to the web server (port 80) (so that we can see what is happening)

```
telnet www.kwarc.info 80
```

▷ Send off the GET request

```
GET /teaching/GenCS2.html http/1.1
Host: www.kwarc.info
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; rv:1.9.2.4)
  Gecko/20100413 Firefox/3.6.4
```

▷ Response from the server

```
HTTP/1.1 200 OK
Date: Mon, 03 May 2010 06:48:36 GMT
Server: Apache/2.2.9 (Debian) DAV/2 SVN/1.5.1 mod_fastcgi/2.4.6 PHP/5.2.6-1+lenny8 with
    Suhosin-Patch mod_python/3.3.1 Python/2.5.2 mod_ssl/2.2.9 OpenSSL/0.9.8g
Last-Modified: Sun, 02 May 2010 13:09:19 GMT
ETag: "1c78b-db1-4859c2f221dc0"
Accept-Ranges: bytes
Content-Length: 3505
Content-Type: text/html

<!--This file was generated by ws2html.xsl. Do NOT edit manually! -->
<html xmlns="http://www.w3.org/1999/xhtml"><head>...</head></html>
```

HTML: Hypertext Markup Language

▷ **Definition 32** The HyperText Markup Language (HTML), is a representation format for web pages. Current version 4.01 is defined in [RHJ98].

▷ **Definition 33 (Main markup tagsof HTML)** HTML marks up the structure and apearance of text with tags of the form `<el>` (begin) and `</el>` (end), where `el` is one of the following

structure	`html`,`head`, `body`	metadata	`title`, `link`, `meta`
headings	`h1`, `h2`, ..., `h6`	paragraphs	`p`, `br`
lists	`ul`, `ol`, `dl`, ..., `li`	hyperlinks	`a`
images	`img`	tables	`table`, `th`, `tr`, `td`, ...
CSS style	`style`, `div`, `span`	old style	`b`, `u`, `tt`, `i`, ...
interaction	`script`	forms	`form`, `input`, `button`

▷ **Example 34** A (very simple) HTML file.

```
<html>
  <body>
    <p>Hello GenCSII!</p>
  </body>
</html>
```

▷ **Example 35** Forms contain input fields and explanations.

```
<form name="input" action="html_form_submit.asp" method="get">
  Username: <input type="text" name="user" />
  <input type="submit" value="Submit" />
</form>
```

Username: Submit

HTML5: The Next Generation HTML

▷ **Definition 36** The HyperText Markup Language (HTML5), is believed to be the next generation of HTML. It is defined by the W3C and the WhatWG.

▷ HTML5 includes support for video and `MathML` (without namespaces).

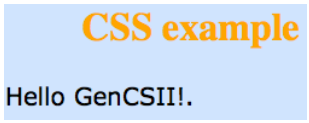
CSS: Cascading Style Sheets

▷ Idea: Separate structure/function from appearance.

Definition 37 The Cascading Style Sheets (CSS), is a style sheet language that allows authors and users to attach style (e.g., fonts and spacing) to structured documents. Current version 2.1 is defined in [BCHL09].

▷ **Example 38** Our text file from Example 34 with embedded CSS

```
<html>
  <head>
    <style type="text/css">
       body {background-color:#d0e4fe;}
       h1 {color:orange;
                text-align:center;}
       p {font-family:"Verdana";
                font-size:20px;}
   </style>
  </head>
  <body>
    <h1>CSS example</h1>
    <p>Hello GenCSII!.</p>
  </body>
</html>
```



Dynamic HTML

▷ Idea: generate some of the web page dynamically. (embed interpreter into browser)

Definition 39 JavaScript is an object-oriented scripting language mostly used to enable programmatic access to the document object model in a web browser, providing enhanced user interfaces and dynamic websites. Current version is standardized by ECMA in [ECM09].

▷ **Example 40** We write the some text into a HTML document object (the document API)

```
<html>
  <head>
      <script type="text/javascript">document.write("This␣is␣my␣first␣JavaScript!");</script>
  </head>
  <body>
     <!-- nothing here; will be added by the script later -->
 </body>
</html>
```

Applications and useful tricks in Dynamic HTML

▷ hide document parts by setting CSS `style` attributes to `display:none`

```
<html>
  <head>
     <style type="text/css">#dropper { display: none; }</style>
     <script language="JavaScript" type="text/javascript">
        function toggleDiv(element){
            if(document.getElementById(element).style.display = 'none')
                {document.getElementById(element).style.display = 'block'}
            else if(document.getElementById(element).style.display = 'block')
                {document.getElementById(element).style.display = 'none'}}
      </script>
  </head>
  <body>
     <div onClick="toggleDiv('dropper');">...more </div>
     <div id="dropper">
        <p>Now you see it!</p>
    </div>
  </body>
</html>
```

▷ precompute input fields from browser caches and cookies

▷ write "gmail" or "google docs" in JavaScript web applicaitions.

Cookies

▷ **Definition 41** A cookie is a little text files left on your hard disk by some websites you visit.

▷ cookies are data not programs, they do not generate pop-ups or behave like viruses, but they can include your log-in name and browser preferences

▷ cookies can be convenient, but they can be used to gather information about you and your browsing habits

▷ **Definition 42** third party cookies are used by advertising companies to track users across multiple sites

6 Computing with Documents

Regular Expressions

▷ **Definition 43** A regular expression (also called regexp) is a formal expression that specifies a set of strings.

▷ **Definition 44 (Meta-Characters for Regexps)**

char	denotes
.	any single character
^	beginning of a string
$	end of a string
[...]	any single character in the brackets
[^ ...]	any single character not in the brackets
(...)	marks a group
$\backslash n$	the n^{th} group
|	disjunction
$*$	matches the preceding element zero or more times
$+$	matches the preceding element one or more times
?	matches the preceding element zero or one times
$\{n,m\}$	matches the preceding element between n and m times

▷ **Example 45 (Regular Expressions and their Values)**

regexp	values
car	car
.at	cat, hat, mat, ...
[hc]at	cat, hat, ...
[^ c]at	hat, mat, ...
^ [hc]at	hat, cat, but only at the beginning of the line
$[0-9]$	Digits
$[1-9][0-9]*$	natural numbers
$(.*)\backslash 1$	mama, papa, wakawaka
cat|dog	cat, dog

▷ A regular expression can be interpreted by a regular expression processor (a program that identifies parts that match the provided specification) or a compiled by a parser generator.

Playing with Regular Expressions

▷ If you want to play with regexps, go e.g. to `http://regexpal.com`

The sed Stream Editor

▷ **Definition 46** The sed utility is a stream editor, it takes a stream (think file) and some regexp replacement commands as an input and gives a stream as a output.

A sed command is of the form s/⟪*regexp*⟫/⟪*replacement*⟫/ (replace once) or s/⟪*regexp*⟫/⟪*replacement*⟫/g (replace globally).

To invoke sed in a shell (e.g. on `linux`, `MacOSX`, or `cygwin` on `Windows`)

sed −e 's/oldstuff/newstuff/g' inputFileName > outputFileName

or (if sedfile.**sed** contains many sed commands)

sed −f sedfile.**sed** inputFileName > outputFileName

▷ **Example 47 (Update the Jacobs Web Site)**

sed −e 's/International University/Jacobs University/g;s/IUB/Jacobs/g' index.html > index.html

Example 48 (Stalin eliminates Trotzki) Let cleanse.**sed** be the sed file

```
s/Leon Trotzki//g;s/Trotzki//g
s/Lev Davidovich Bronstein//g;s/Davidovich//g;s/Bronstein//g
```

then Stalin can just use the following shell script to cleanse Kreml documents

find / −name −E ''.*\.html|.*\.txt'' −**exec** '**sed** −f cleanse.**sed** {} > {} \;

The lex/flex Lexer Generator

▷ **Definition 49** The lex is a generator of lexical analyzers (lexers), i.e. a program that reads a lexer specification and outputs C code for a lexer.

A lexer specification is a list of pairs $\langle R, P \rangle$, where R is a regexp and P is C code to be executed when R is matched.

lex is part of UNIX (proprietary), it is extended by the open-source flex.

▷ **Example 50 (Spotting Integers)**

```
−?[1−9][0−9]∗ {printf("Saw␣an␣integer:␣%s\n", yytext)}
.|\n { /∗ Ignore all other characters. ∗/ }
```

If this input is given to flex, it will be converted into a $CLanguage$ file, lex.yy.c. This can be compiled into an executable which matches and outputs strings of integers. For example, given the input abc123z.!& ∗ 2ghj − 6 the program will print:

```
Saw an integer: 123
Saw an integer: 2
Saw an integer: -6
```


lex Example: Tokenizing Artithmetic Expressions

▷ **Example 51** We want to build a simple calculator, so we need a tokenizer for arithmetic expressions. Here is a the flex code for one (see [Vol11] for details):

```
delim          [ \t]
whitesp        {delim}+
digit          [0-9]
number         [-]?{digit}*[.]?{digit}+
%%
{number}   { sscanf(yytext, "%lf", &yylval); return NUMBER;}
"+"        { return PLUS; }
"-"        { return MINUS; }
"/"        { return SLASH; }
"*"        { return ASTERISK; }
"("        { return LPAREN; }
")"        { return RPAREN; }
"\n"       { return NEWLINE; }
{whitesp}  { /* No action and no return */}
```

▷ The declarations before the %% are abbreviations for number (note that they are recursive)

▷ instead of printing notifications we just return token types (values are in yytext)

The yacc/bison Parser Generator

▷ **Definition 52** yacc (Yet Another Compiler Compiler) is a parser generator, i.e. a program that reads a parser specification and outputs C code for a parser. Historically, yacc was used to generate the C parser in UNIX, today, it is superseded by open-source extensions, e.g. bison.

A yacc parser specification consists of three parts divided by %%.

1. token definitions that specify which tokens to expect from flex
2. grammar and the actions: $$ is the constructed result.
3. more C code, including the usual main function.

yacc/bison Example: Building a Calculator

▷ **Example 53** We want to build a simple calculator, so we need a tokenizer for arithmetic expressions. Here is a the flex code for one (see [Vol11] for details):

```
 %token NEWLINE NUMBER PLUS MINUS SLASH ASTERISK LPAREN RPAREN
%%
input:              /* empty string */
     | input line;
 line: NEWLINE
     | expr NEWLINE           { printf("\t%.10g\n",$1); };
expr: expr PLUS term          { $$ = $1 + $3; }
     | expr MINUS term         { $$ = $1 - $3; }
     | term;
term: term ASTERISK factor    { $$ = $1 * $3; }
     | term SLASH factor       { $$ = $1 / $3; }
     | factor;
factor:  LPAREN expr RPAREN   { $$ = $2; }
       | NUMBER;
%%
int main(void) {yyparse();exit(0)}
```

Using this to generate a parser with bison gives a program tcalc which is a simple calculator

```
-1.1 + 2 * ( 4 / 3 )
         1566666667
2+2
         4
```


The perl Programming Language

▷ **Definition 54** perl is a high-level, general-purpose, interpreted, dynamic programming language that makes extensive use of regular expressions.

▷ perl can directly use sed commands (with more regexps and execute subroutines)

▷ instead of specifying the language, let us go through an example!

perl Example: Correcting and Anonymizing Documents

▷ **Example 55** We write an a program that makes simple corrections on documents and also crossres out all names.

- ▷ *The worst president of the US,arguably was George W. Bush. right?*
- ▷ *However,are you famILIar with Paul Erdős or Henri Poincaré?* (Unicode)

Here is the program:

- ▷ we first initialize and load modules

```
#!/usr/bin/perl -w
use warnings;
use utf8;
use Encode;
```

- ▷ then we decode the argument and put it into a variable

```
my $expr = shift;
$expr = decode('utf8',$expr);
```

- ▷ We put put a space after a comma,

```
$expr =~ s/,(\S)/, $1/g;
```

- ▷ next we make abbreviations for regular expressions to save space

```
$c=qr/\p{UpperCase_Letter}/;
$l=qr/\p{Lowercase_Letter}/;
```

- ▷ capitalize the first letter of a new sentence,

```
$expr =~ s/([?.!])\s($l)/$1."␣".uc($2)/eg;
```

- ▷ remove capital letters in the middle of words

```
$expr =~ s/($l)($c+)($l)/$1.lc($2).$3/eg;
```

- ▷ and we cross-out for official public versions of government documents,

```
$expr =~ s/($c$l+ ($c$l*(\.?) )?$c$l+)/'X' x length($1)/eg;
```

- ▷ finally, we print the result

```
print $expr,"\n";
```

The worst president of the US,arguably was George W. Bush. right? becomes
The worst president of the US, arguably was XXXXXX XX XXXX right?

7 Programming Documents

The TEX Typesetting System

▷ **Definition 56** Typesetting is the process of creating the visual appearance of a document by assembling glyphs (visual representations of characters; also called types) on pages.

▷ Since Gutenberg's time (to ca. 1975), typesetting was done by assembling movable types (special metal positives of single letters) into lines and later into pages, which were inked and the printed; or using negatives to form cast-metal positives for printing.

▷ **Definition 57** TEX is a typesetting program designed by Donald Knuth in 1978. It combines movable types (character boxes) with macro programming.

▷ **Definition 58** The `pdftex` program reads a file of text marked up with TEX macros and outputs PDF.

▷ **Example 59 (Hello World in TEX)** `pdftex` typesets the following TEX program

```
Hello, World \bye
```

The command sequence `\bye` stops `pdftex` and is not shown in the output.

TEX Macros for Programming Documents

- ▷ TEX uses command sequences (words starting with "\"; also called macros) for special effects.
- ▷ **Example 60** \bye stops the formatter, \alpha prints α, \int prints $\int$,...
- ▷ Users can also define TEX macros as abbreviations via \def
- ▷ **Example 61** \def\tdm{Text and Digital Media} defines the macro \tdm.
  We love the USC ``\tdm''! expands to
  "We love the USC "Text and Digital Media"!
- ▷ TEX macros can have arguments specify with #1, #2...: delimit with { and }
- ▷ **Example 62** with the macro \def\tnwhat#1{Text and \textbf{#1}}
  \tnwhat{Beer} expands to "Text and **Beer**"
- ▷ TEX has a math mode for formulae: Greeks, sub/superscripts with ^, _
  delimit with $ (inline math) or \[and \] (display style)
- ▷ **Example 63** $\int_0^\infty f(\theta) d\theta$ expands to $\int_0^\infty f(\theta)d\theta$
- ▷ **Example 64** Use macros in math mode as well: \def\frac#1#2{#1\over #2} Then
  \[1+\frac{2}{2+\frac{3}{3+\ldots}}\] expands to

$$1+\frac{2}{2+\frac{3}{3+\ldots}}$$

TEX Counters

- ▷ TEX uses special macros as counters, \newcount, allocates a counter, \advance alters it, and \the references it.
- ▷ **Example 65** We define a sectioning macros

```
\newcount\seccount % allocate a new counter for sections
\newcount\subseccount % allocate a new counter subsections
\seccount0\subseccount0 % initialize both with 0
\def\section#1{ % begin macro definition
\advance\seccount by 1 % step the counter
\subseccount0 % reset the subsection counter
\textbf{\Large\the\seccount. #1} % section number and title
} % end macro definition
\def\subsection#1{\advance\subseccount by 1
\textbf{\large\the\seccount.\the\subseccount. #1}}
```

TEX Conditionals

▷ TEX provides some conditional for your use:
e.g. `\ifx` compares two macros, `\ifnum` compares two number, and `\ifmmode` tells you if you are in math mode.
`\if`⟨cond⟩`...\else...\fi` uses it.

▷ TEX uses special macros for conditionals, `\newif\if`⟨cond⟩, allocates a conditional, `\if`⟨cond⟩`true` and `\if`⟨cond⟩`false` alter it,

▷ **Example 66** sdfsdf

Programming a Chain Letter

▷ **Example 67 (A Parametric Reminder)**

```
\def\reminder#1#2{\hfill Bremen, \today\par\bigskip
\noindent Dear #1,\par\medskip\noindent
please be sure that you will not forget to come to the lecture
today. We are planning big things.\par\medskip\noindent
Sincerely,\par\bigskip\noindent #2\newpage}
```

▷ **Example 68 (Programming a Serial Letter)**
We can use arbitrary characters to delineate arguments in macro definitions.

```
\def\sletter#1,#2;{\def\first{#1}\def\second{#2}\def\empty{}
\ifx\first\empty\else\reminder{#1}{Thomas \& Michael}
\ifx\second\empty\else\sletter#2,;\fi\fi}
\def\serialletter#1{\sletter #1;}
```

Also nothing prevents us from using recursion.

▷ **Example 69 (Making a Serial Letter)**

```
\serialletter{Mati, Anca, Isabel, Calin}
```

TEX Macro Packages

- ▷ Idea: Separate out common macro definitions into a separate file and include that via `\input`. (So we can reuse them over multiple documents)
- ▷ Actually: many people have already done that.
- ▷ The AMS (American Mathematical Society) supplies AMSTEX: TEX macros that make it more convenient to write Math (e.g. the `\frac` macro)
- ▷ Till Tantau supplies tikz (TEX ist kein Zeichenprogram): TEX macros that allow you to draw images.
- ▷ Leslie Lamport supplies LATEX, a set of TEX packages and classes.
- ▷ Michael Kohlhase supplies STEX, a semantic variant of LATEX. classes.
- ▷ The bibTEX package handles bibliographic references.

The Anatomy of a LATEX Document

▷ **Example 70 (A LATEX file)**

```
\documentclass{article} % use the article class (Journal Article)
\title{Anatomy of a {\LaTeX} Document} % specify the title
\author{Michael Kohlhase\\Jacobs University Bremen} % and the author
\date{\today} % and the date
\begin{document} % start the document
\maketitle % make the title
\tableofcontents % make the table of contents
\section{Introduction}\label{sec:intro}
This is really easy, just start writing,
\section{Main Part}\label{sec:main}
We refer the reader to \cite{Lamport:ladps94} for details.
\section{Conclusion}\label{concl:intro}
As we already said in the in Section \ref{sec:intro} this was not so bad was it?
\bibliographystyle{alpha}
\bibliography{kwarc}
\end{document}
```

▷ **Example 71 (and the bibTEX database used in it)**

```
@BOOK{Lamport:ladps94,
  title = {LaTeX: A Document Preparation System, 2/e},
  publisher = {Addison Wesley},
  year = {1994},
  author = {Leslie Lamport}}
```


8 Copyright and Licensing

Copyright

▷ **Definition 72** Copyright is a set of exclusive rights granted to the author or creator of an original work, including the right to copy, distribute and adapt the work.

▷ Clarification: Copyright does not protect ideas, only their expression. ($\neq$ patents)

▷ Registration: In most jurisdictions copyright arises upon fixation and does not need to be registered.

▷ Control: Copyright owners have the exclusive statutory right to exercise control over copying and other exploitation of the works

▷ Expiration: After a specific period of time, the work is said to enter the public domain.

▷ Exceptions: Some jurisdictions state exceptions(e.g. documents funded by US government are copyright-exempt)

▷ Permission: Uses covered under limitations and exceptions to copyright, such as fair use, do not require permission from the copyright owner. All other uses require permission.

▷ In particular: If you write a text, then you have copyright (any original text)

▷ and: nobody else but you has any right to copy, distribute, or adapt your text

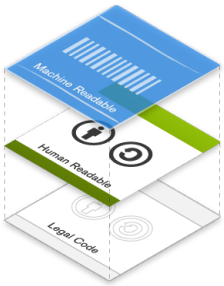
▷ so: if you want to allow them to copy, distribute, or adapt your text, you have to explicitly give them the right to do so (licensing)

▷ Licensing: Copyright owners can license or permanently transfer or assign their exclusive rights to others.

Open Content via Open Content Licenses

▷ **Definition 73** The Creative Commons licenses are

- ▷ a common legal vocabulary for sharing content
- ▷ to create a kind of "public domain" using licensing



▷ Creative Commons license provisions (`http://www.creativecommons.org`)

- ▷ author retains copyright on each module/course
- ▷ author licenses material to the world with requirements
  - +/- attribuition (must reference the author)
  - +/- commercial use (can be restricted)
  - +/- derivative works (can allow modification)
  - +/- share alike ("copyleft") (modifications must be donated back)

9 An Overview over XML Technologies

Excursion: XML (EXtensible Markup Language)

▷ XML is language family for the Web

- ▷ tree representation language (begin/end brackets)
- ▷ restrict instances by *Doc. Type Def. (DTD)* or *Schema* (Grammar)
- ▷ Presentation markup by *style files* (XSL: XML Style Language)

▷ XML is extensible HTML & simplified SGML

▷ logic annotation (*markup*) instead of presentation!

▷ many tools available: parsers, compression, data bases, ...

▷ conceptually: transfer of directed graphs instead of strings.

▷ details at `http://www.w3c.org`

XML is Everywhere (E.g. document metadata)

▷ **Example 74** Open a PDF file in AcrobatReader, then cklick on $File \searrow DocumentProperties \searrow DocumentMetadata \searrow ViewSource$, you get the following text: (showing only a small part)

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
         xmlns:iX='http://ns.adobe.com/iX/1.0/'>
  <rdf:Description xmlns:pdf='http://ns.adobe.com/pdf/1.3/'>
    <pdf:CreationDate>2004-09-08T16:14:07Z</pdf:CreationDate>
    <pdf:ModDate>2004-09-08T16:14:07Z</pdf:ModDate>
    <pdf:Producer>Acrobat Distiller 5.0 (Windows)</pdf:Producer>
    <pdf:Author>Herbert Jaeger</pdf:Author>
    <pdf:Creator>Acrobat PDFMaker 5.0 for Word</pdf:Creator>
    <pdf:Title>Exercises for ACS 1, Fall 2003</pdf:Title>
  </rdf:Description>
  ...
  <rdf:Description xmlns:dc='http://purl.org/dc/elements/1.1/'>
    <dc:creator>Herbert Jaeger</dc:creator>
    <dc:title>Exercises for ACS 1, Fall 2003</dc:title>
  </rdf:Description>
</rdf:RDF>
```


This is an excerpt from the document metadata which AcrobatDistiller saves along with each PDF document it creates. It contains various kinds of information about the creator of the document, its title, the software version used in creating it and much more. Document metadata is useful for libraries, bookselling companies, all kind of text databases, book search engines, and generally all institutions or persons or programs that wish to get an overview of some set of books, documents, texts. The important thing about this document metadata text is that it is not written in an arbitrary, PDF-proprietary format. Document metadata only make sense if these metadata are independent of the specific format of the text. The metadata that MSWord saves with each Word document should be in the same format as the metadata that Amazon saves with each of its book records, and again the same that the British library uses, etc.

XML is Everywhere (E.g. Web Pages)

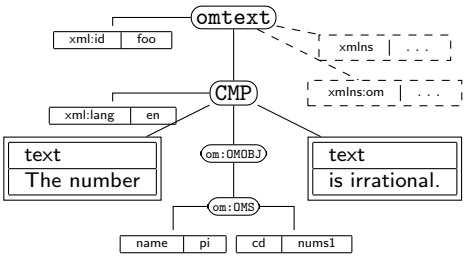
▷ **Example 75** Open web page file in FireFox, then click on $View \searrow PageSource$, you get the following text: (showing only a small part and reformatting)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD␣XHTML␣1.0␣Transitional//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Michael Kohlhase</title>
    <meta name="generator"
          content="Page␣generated␣from␣XML␣sources␣with␣the␣WSML␣package"/>
  </head>
  <body>...
    <p>
      <i>Professor of Computer Science</i><br/>
      Jacobs University<br/><br/>
      <strong>Mailing address - Jacobs (except Thursdays)</strong><br/>
      <a href="http://www.jacobs-university.de/schools/ses">
        School of Engineering &amp;amp; Science
       </a><br/>...
    </p>...
  </body>
</html>
```

XML Documents as Trees

▷ Idea: An XML Document is a Tree

```
<omtext xml:id="foo"
   xmlns="..."
   xmlns:om="...">
  <CMP xml:lang='en'>
   The number
   <om:OMOBJ>
     <om:OMS cd="nums1"
             name="pi"/>
   <om:OMOBJ>
   is irrational.
  </CMP>
</omtext>
```



▷

▷ **Definition 76** The XML document tree is made up of element nodes, attribute nodes, text nodes (and namespace declarations, comments,...)

▷ **Definition 77** For communication this tree is serialized into a balanced bracketing structure, where

- ▷ an element `el` is represented by the brackets `<el>` (called the opening tag) and `</el>` (called the closing tag).
- ▷ The leaves of the tree are represented by empty elements (serialized as `<el></el>`, which can be abbreviated as `<el/>`
- ▷ and text nodes (serialized as a sequence of UniCode characters).
- ▷ An element node can be annotated by further information using attribute nodes — serialized as an attribute in its opening tag

Note: As a document is a tree, the XML specification mandates that there must be a unique document root.

The Dual Role of Grammar in XML (I)

▷ The XML specification [XML] contains a large character-level grammar.(81 productions)

NameChar :== Letter | Digit | '.' | '−' | '_' | ':' | CombiningChar | Extender

Name :== (Letter | '_' | ':') (NameChar)*

element :== EmptyElementTag | STag content ETag

STag :== '<' $(S)^*$ Name $(S)^*$ attribute $(S)^*$ '>'

ETag :== '</' $(S)^*$ Name $(S)^*$ '>'

EmptyElementTag :== '<' $(S)^*$ Name $(S)^*$ attribute $(S)^*$ '/>'

▷ use these to parse well-formed XML document into a tree data structure

▷ use these to serialize a tree data structure into a well-formed XML document

▷ Idea: Integrate XML parsers/serializers into all programming languages to communicate trees instead of strings. (more structure $\hat{=}$ better CS)

The Dual Role of Grammar in XML (II)

▷ Idea: We can define our own XML language by defining our own elements and attributes.

▷ Validation: Specify your language with a tree grammar (works like a charm)

▷ **Definition 78** Document Type Definitions (DTDs) are grammars that are built into the XML framework.

Put `<!DOCTYPE foo PUBLIC "foo.dtd">` into the second line of the document to validate.

▷ **Definition 79** RelaxNG is a modern XML grammar/schema framework on top of the XML framework.

RelaxNG, A tree Grammar for XML

▷ **Definition 80** Relax NG (RelaxNG: Regular Language for XML Next Generation) is a tree grammar framework for XML documents.

A RelaxNG schema is itself an XML document; however, RelaxNG also offers a popular, non-XML compact syntax.

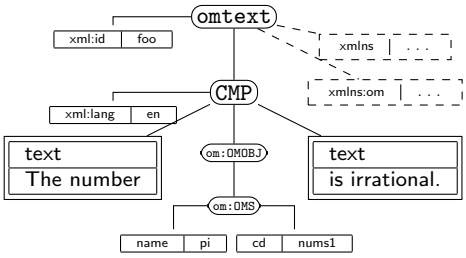
▷ **Example 81** The RelaxNG grammars validate the left document

document	RelaxNG in XML	RelaxNG compact
<lecture> <slide id="foo"> first slide </slide> <slide id="bar"> second one </slide> </lecture>	<grammar> <start> <element name="lecture"> <oneOrMore> <ref name="slide"/> </oneOrMore> </element> </start> <define name="slide"> <element name="slide"> <text/> </element> <attribute name="id"> <text/> </attribute> </define> </grammar>	start = element lecture {slide+} slide = element slide {attribute id {text} text}

XPath, A Language for talking about XML Tree Fragments

▷ **Definition 82** The XML path language (XPath) is a language framework for specifying fragments of XML trees.

▷ **Example 83**



XPath exp.	fragment
/	root
omtext/CMP/∗	all CMP children
//@name	the name attribute on the om:OMS element
//CMP/∗[1]	the first child of all OMS elements
//∗[@cd =′ nums1′]	all elements whose cd has value nums1

XSLT, A tree Transformer for XML

▷ **Definition 84** XSLT (Extensible Stylesheet Language Transformations) is a declarative, XML-based language used for the transformation of XML documents. It is standardized by the W3C.

▷ **Definition 85** XSLT stylesheets consist of a set of templates which match a XML elements via an `XPath` expression and create a result tree.

▷ **Definition 86** An XSLT processor is a program that takes an XSLT stylesheet S and an XML file X as input and transforms X as specified by the templates in S.

▷ **Example 87** There are various open source or free XSLT processors

- ▷ `xsltproc` [Vei] is very fast, but only supports XSLT version 1.
- ▷ `saxon` [Kay08] supports XSLT version 2, but is slower.

▷ **Example 88** Use this stylesheet to extract a numbered table of contents from an HTML document

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html><body><xsl:apply-templates select="//h1"/></body></html>
  </xsl:template>

  <xsl:template match="*"/>

  <xsl:template match="h1">
    <p style="font-size:large">
      <xsl:value-of select="preceeding-sibling::h1"/>
      <xsl:copy-of select="*|text()"/>
    </p>
  </xsl:template>
</xsl:stylesheet>
```


10 Converting the arXiv

The arXMLiv Project: arXiv to semantic XML

▷ Idea: Develop a large corpus of knowledge in OMDoc/PhysML

- ▷ to get around the chicken-and-egg problem of MKM
- ▷ corpus-linguistic methods for semantics recovery (linguists interested)

▷ **Definition 89 (The Cornell Preprint arXiv)** (http://www.arxiv.org)

Open access to ca. 600.000 e-prints in Physics, Mathematics, Computer Science and Quantitative Biology.

▷ **Definition 90 (The arXMLiv Project)** (http://arxmliv.kwarc.info)

- ▷ use Bruce Miller's LaTeXML to transform to XHTML+MathML
- ▷ we have an automated, distributed build system (ca. 1 CPU-year)
- ▷ create ca. 8000 LaTeXML binding files (8 Jacobs students help)
- ▷ use MathWebSearch to index XML version (realistic search corpus)

▷ More semantic information will enable more added-value services

- ▷ e.g. filter papers by model assumptions(expanding, stationary, or contracting universe)
- ▷ use linguistic techniques to add the necessary semantics

Why reimplement the TEX parser?

▷ Problem: The TEX parser can change the tokenizer while at runtime (\catcode)

▷ **Example 91 (Obfuscated TEX)** David Carlisle posted the following, when someone claimed that word counting is simple in TEX/LATEX

```
\let~\catcode~‘76~‘A13~‘F1~‘j00~‘P2jdefA71F~‘7113jdefPALLF
PA’’FwPA;;FPAZZFLaLPA//71F71iPAHHFLPAzzFenPASSFthP;A$$FevP
A@@FfPARR717273F737271P;ADDFRgniPAWW71FPATTFvePA**FstRsamP
AGGFRruoPAqq71.72.F717271PAYY7172F727171PA??Fi*LmPA&&71jfi
Fjfi71PAVVFjbigskipRPWGAUU71727374 75,76Fjpar71727375Djifx
:76jelse&U76jfiPLAKK7172F7117271PAXX71FVLnOSeL71SLRyadR@oL
RrhC?yLRurtKFeLPFovPgaTLtReRomL;PABB71 72,73:Fjif.73.jelse
B73:jfiXF71PU71 72,73:PWs;AMM71F71diPAJJFRdriPAQQFRsreLPAI
I71Fo71dPA!!FRgiePBt’el@ lTLqdrYmu.Q.,Ke;vz vzLqpip.Q.,tz;
;Lql.IrsZ.eap,qn.i. i.eLlMaesLdRcna,;!;h htLqm.MRasZ.ilk,%
s$;z zLqs’.ansZ.Ymi,/sx ;LYegseZRyal,@i;@ TLRlogdLrDsW,@;G
LcYlaDLbJsW,SWXJW ree @rzchLhzsW,;WERcesInW qt.’oL.Rtrul;e
doTsW,Wk;Rri@stW aHAHHFndZPpqar.tridgeLinZpe.LtYer.W,:jbye
```

When formatted by TeX, this leads to the full lyrics of “The twelve days of christmas”. When formattet by LATEXML, it gives

```
<song>
  <verse>
    <line>On the first day of Christmas my true love gave to me</line>
    <line>a partridge in a pear tree.</line>
  </verse>
  <verse>
    <line>On the second day of Christmas my true love gave to me</line>
    <line>two turtle doves</line>
    <line>and a partridge in a pear tree.</line>
  </verse>
  <verse>
    <line>On the third day of Christmas my true love gave to me</line>
    <line>three french hens</line>
    <line>two turtle doves</line>
    <line>and a partridge in a pear tree.</line>
  </verse>
  <verse>
    <line>On the fourth day of Christmas my true love gave to me</line>
    <line>four calling birds</line>
    <line>three french hens</line>
    <line>two turtle doves</line>
    <line>and a partridge in a pear tree.</line>
  </verse>
  ...
```

▷

▷ But the real reason is: that we can take advantage of the semantics in the LATEX.

▷ LATEXML does not need to expand macros, we can tell it about XML equivalents.

▷ **Example 92 (Recovering the Semantics of Proofs)**
Add the following magic incantation to amsthm.sty.ltxml (LATEXML binding)

```
DefEnvironment('{proof}',"<xhtml:div class='proof'>#body</xhtml:div>");
```

The arXMLiv approach: Try to cover most packages and classes in the arXiv (Jacobs undergrads' intro to research)

Future Plans for arXMLiv

- ▷ State: LaTeX-to-XHTML+MathML Format Conversion works (65% success)
- ▷ Over the summer: Bump up success rate to 75%, daily downloads, web site, instrumentation,...
- ▷ Soon: Integrate user-level quality control (integrate JS feedback into html)
- ▷ starting Fall: Extend post-processing by linguistic methods for semantic analysis
  - ▷ build semantics blackboard/database for linguistic information (rdf triples)
  - ▷ extend build system for arbitrary XML2BB processes
  - ▷ invite the linguists over (they leave semantics results in BB)
  - ▷ harvest the semantics BB to get OMDoc representations

Current and Possible Applications

- ▷ the arxmliv build system `http://arxmliv.kwarc.info`
- ▷ the transformation web service `http://tex2xml.kwarc.info`
- ▷ LaTeXML daemon to avoid `perl` and LaTeX startup times (Deyan Ginev)
  - ▷ keep LaTeXML alive as a daemon that can process multiple files/fragments (patch memory leaks)
  - ▷ a LaTeXML client just passes files/fragments along ($\frac{10}{s}$ to $\frac{100}{s}$)
- ▷ embedding/editing LaTeX in web pages `http://tex2xml.kwarc.info/test`
- ▷ a MathML version of the arXiv allows vision-impared readers to understand the texts
- ▷ generalization search (need to know sentence structure for detecting universal variables)
- ▷ semantic search by academic discipline or theory assumption (need discourse structure)
- ▷ development of scientific vocabularies (over the past 18 years; drink from the source)

11 Electronic Books and their Formats

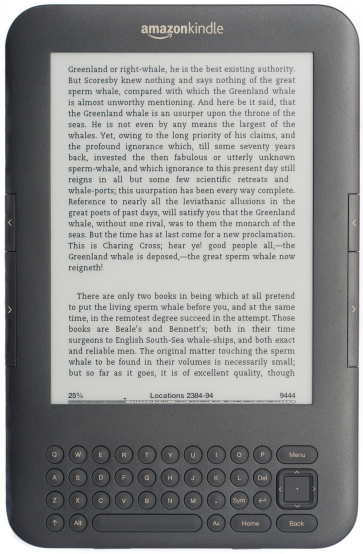
Electronic Books

▷ **Definition 93** An electronic book (eBook) is a publication in electronic form that can be read on digital devices.

▷ **Example 94** Arguably the first eBooks were the texts provided by Project Gutenberg in 1971.

▷ **Definition 95** An electronic book reader (eReader) is a hardware or software devide for reading electronic books.

▷ **Example 96** Popular hardware-based eReaders are Kindle (Amazon.com), the iPad (Apple), and the Nook (Barnes&Noble), but sofware readers also abound.



EPUB: A Standard for Electronic Publishing [Wik11]

▷ **Definition 97** EPUB is a free and open standard for electronic books provided by the International Digital Publishing Forum (IDPF).
It consists of three specifications:

Open Publication Structure (OPS), essentially XHTML and CSS for the document contents

Open Packaging Format (OPF), which describes the structure of the EPUB file in XML.

Open Container Format (Ocf), which collects all files as a ZIP archive.

▷ EPUB files usually have the extension `.epub`.

▷ EPUB does not specify a format for digital rights management (DRM), which makes it less attractive for the big publishers.

▷ EPUB is supported by almost all eReaders and publishing software

EPUB: Open Packaging Format & Navigation Control

▷ **Definition 98** The Open Packaging Format (OPF) is a standard for specifying giving additional structure and coherence to an electronic book in EPUB. It specifies the

- ▷ contents (what files) in the `manifest` element
- ▷ metadata (author, date, etc) in the `metadata` element
- ▷ linear reading order in the `spine` element, and
- ▷ (optionally) important structural components in the `guide` element.

of the package in a OPF file with the extension `.opf`.

▷ **Definition 99** The navigation control of the an EPUB gives a machine-readable table of contents of the book in XML.

An Example OPF file

```
<?xml version="1.0"?>
<package version="2.0" xmlns="http://www.idpf.org/2007/opf" unique-identifier="BookId">

  <metadata xmlns:dc="http://purl.org/dc/elements/1.1/"
                xmlns:opf="http://www.idpf.org/2007/opf">
    <dc:title>Pride and Prejudice</dc:title>
    <dc:language>en</dc:language>
    <dc:identifier id="BookId" opf:scheme="ISBN">123456789X</dc:identifier>
    <dc:creator opf:file-as="Austen,␣Jane" opf:role="aut">Jane Austen</dc:creator>
  </metadata>

  <manifest>
    <item id="chapter1" href="chapter1.xhtml" media-type="application/xhtml+xml"/>
    <item id="stylesheet" href="style.css" media-type="text/css"/>
    <item id="ch1-pic" href="ch1-pic.png" media-type="image/png"/>
    <item id="myfont" href="css/myfont.otf" media-type="application/x-font-opentype"/>
    <item id="ncx" href="book.ncx" media-type="application/x-dtbncx+xml"/>
  </manifest>

  <spine toc="ncx">
    <itemref idref="chapter1" />
  </spine>

  <guide>
    <reference type="loi" title="List␣Of␣Illustrations" href="appendix.html#figures" />
  </guide>

</package>
```


An Example NCX file

```
<?xml version="1.0" encoding="UTF-8"?>
<ncx version="2005-1" xml:lang="en" xmlns="http://www.daisy.org/z3986/2005/ncx/">

  <head>
   <meta name="dtb:uid" content="123456789X"/> <!-- same as in .opf -->
   <meta name="dtb:depth" content="1"/> <!-- 1 or higher -->
   <meta name="dtb:totalPageCount" content="0"/> <!-- must be 0 -->
   <meta name="dtb:maxPageNumber" content="0"/> <!-- must be 0 -->
 </head>

  <docTitle>
    <text>Pride and Prejudice</text>
  </docTitle>

  <docAuthor>
    <text>Austen, Jane</text>
  </docAuthor>

  <navMap>
    <navPoint class="chapter" id="chapter1" playOrder="1">
      <navLabel><text>Chapter 1</text></navLabel>
      <content src="chapter1.xhtml"/>
    </navPoint>
  </navMap>

</ncx>
```

EPUB: Open Container Format

▷ **Definition 100** An EPUB file is a group of files conforming to the OPS/OPF standards that is wrapped in a ZIP file. The Open Container Format (OCF) specifies how these files should be organized in the ZIP archive, and defines two additional files that must be included.

▷ The `mimetype` file must be a text document in ASCII and must contain the string `application/epub+zip`. It must also be uncompressed, unencrypted, and the first file in the ZIP archive.

▷ The purpose of this file is to provide a more reliable way for applications to identify the mimetype of the file than just the `.epub` extension.

▷ Also, there must be a folder named `META-INF` which contains the required file `container.xml`. This XML file points to the file defining the contents of the book. This will be the `.opf` file.

An Example Container

ZIP Container

```
mimetype
META-INF/
  container.xml
OPS/
  book.opf
  book.ncx
  chapter1.xhtml
  ch1-pic.png
  css/
    style.css
    myfont.otf
```

`container.xml`

```
<?xml version="1.0" encoding="UTF-8" ?>
<container version="1.0"
    xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
  <rootfiles>
    <rootfile full-path="OPS/book.opf"
       media-type="application/oebps-package+xml"/>
    <rootfile full-path="OPS/book.ncx"
       media-type="application/x-dtbncx+xml"/>
  </rootfiles>
</container>
```


12 Centralized Version Control

Computing and Managing Differences with `diff` & `patch`

▷ **Definition 101** `diff` is a file comparison utility that computes differences between two files f_1 and f_2. Differences are output linewise in a "patch", which can be applied to f_1 to obtain f_2 via the `patch` utility.

▷ **Example 102**

`The quick brown fox jumps over the lazy dog`	`The quack brown fox jumps over the loozy dog`	`1c1`
		`< The quick brown`
		`---`
		`> The quack brown`
		`3c3`
		`< the lazy dog`
		`---`
		`> the loozy dog`

Merging Differences with `merge3`

▷ There are basically two ways of merging the differences of files into one.

▷ **Definition 103** In two-way merge, an automated procedure tries to combine two different files by copying over differences by guessing or asking the user.

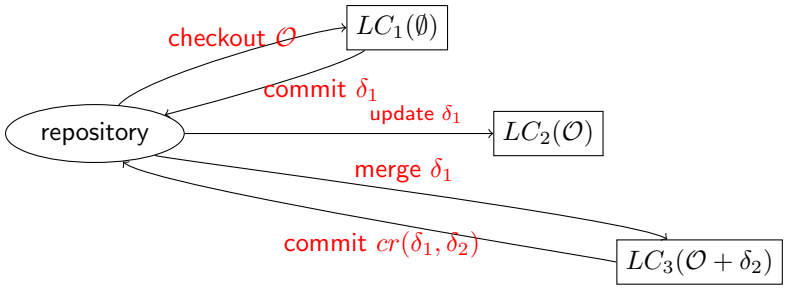
▷ **Definition 104** In three-way merge the files are assumed to be created by changing a joint original (the parent) by editing. The `merge3` tool examines the differences and patterns appearing in the changes between both files as well as the parent, building a relationship model to generate a new revision. Usually, non-conflicting differences (affecting only one of the files) can directly be copied over.

Version Control with Subversion

▷ **Definition 105** Subversion is a centralized version control system that features

- ▷ Central repository (for current revision and reverse diffs)
- ▷ Local working copies (asynchronous checkouts, updates, commits)

They are kept synchronized by passing around `diff` differences and patching the repository and working copies. Conflicts are resolved by (three-way) merge.



13 Writing Technical Documentation and Manuals

13.1 Technical Documentation in DocBook

DocBook

▷ **Definition 106** DocBook is a content markup language for technical documentation based on SGML or XML. It supplies elements/tags for the logical of book-like documents.

▷ DocBook was originally intended for writing technical documents related to computer hardware and software but it can be used for any other sort of documentation.

▷ DocBook content is presentation-neutral and can be published in a variety of formats, including HTML, XHTML, EPUB, PDF, man pages and HTML Help, without requiring users to make any changes to the source.

▷ DocBook began in 1991 as a joint project of HAL Computer Systems and O'Reilly & Associates. Since 1998 it is maintained by a Technical Committee at OASIS.

DocBook Elements

▷ DocBook provides about 400 content markup tags

▷ Structural Elements: specify broad characteristics of their contents, e.g. book, part, article, chapter, appendix, dedication

▷ Block-level Elements: specify structured blocks of text (usually starting and ending with new "lines"). e.g. paragraphs, lists, definitions, etc. They usually have a fixed content model; some can contain text.

▷ Inline-level Elements: wrap text within a block-level element (usually without breaking "lines"), e.g. for emphasis, hyperlinks, definienda,. They typically cause the document processor to apply some kind of distinct typographical treatment to the enclosed text.

DocBook Example

▷ A "Hello World" document in DocBook

```
<?xml version="1.0" encoding="UTF-8"?>
<book xml:id="simple_book" xmlns="http://docbook.org/ns/docbook" version="5.0">
  <title>Very simple book</title>
  <chapter xml:id="chapter_1">
    <title>Chapter 1</title>
    <para>Hello world!</para>
    <para>
      I hope that your day is proceeding
      <emphasis>splendidly</emphasis>!
    </para>
  </chapter>
  <chapter xml:id="chapter_2">
    <title>Chapter 2</title>
    <para>Hello again, world!</para>
  </chapter>
</book>
```

13.2 Topic-Oriented Documentation with DITA

DITA the "Darwin Information Typing Architecture"

▷ **Definition 107** DITA is a topic-oriented content markup language for technical documentation based on XML. It supports a topic-oriented documentation style.

▷ **Definition 108** The basic unit of information in DITA is a topic, i.e. a discrete piece of content that is about a specific subject, has an identifiable purpose, and can stand alone (does not need to be presented in context for the end-user to make sense of the content).

▷ Topics can be reused in any context; DITA makes use of this.

▷ **Definition 109** DITA combines topics into documents via DITA maps.

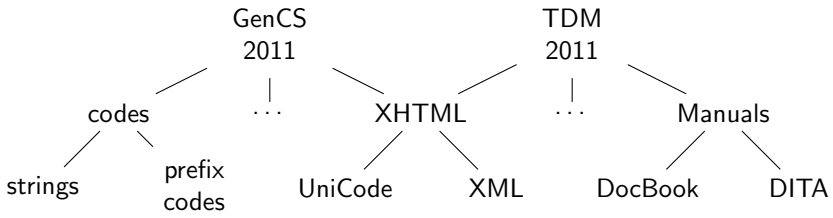
▷ Consequence: A DITA topic (and DITA map) can be referenced in multiple DITA maps.

▷ Extension: Conditional text allows filtering or styling content based on attributes for audience, platform, product, and other properties. (the DITA processor filters text)

Using DITA Maps for Reuse

▷ Idea: Concepts can be reused in more than one DITA map

▷ **Example 110** For instance a module on HTML/XML in the courses "General Computer Science" and "Text and Digital Media".



A DITA Concept File

▷ **Definition 111** A DITA concept is a special DITA topic that describes an abstract idea or a named unit of knowledge.

▷ **Example 112** A concept for "academic conference" (note the conditional text)

```
<concept id="A.dita">
  <title>Academic Conference</title>
  <conbody>
    <p audience="students">
     An <term>academic conference</term> is a gathering of scientists
     who discuss <term>scientific papers</term>.
    </p>
    <p audience="professors">
      An <term>academic conference</term> is a pretense to travel to
      nice locations on university money and drink loads of beer.
    </p>
    <para conref="#topic/p2"/>
  </conbody>
  <related-links>
    <linkpool type="concept">
      <link audience="students" href="http://easychair.org"/>
      <link audience="professors" href="http://acapulco.mx"/>
    </linkpool>
  </related-links>
</concept>
```

We can generate two versions from this content markup format. For instance, with the following DITA value specification:

```
<!-- this file specifies the actions for students -->
<val>
  <prop action="exclude" att="audience" val="professors"/>
  <prop action="include" att="audience" val="students"/>
</val>
```


A DITA Task File

▷ **Definition 113** A DITA task is a special DITA topic that describes a process.

▷ **Example 114** DITA task markup for assignment 8 of the TDM course

```
<task id="TDMassignment8">
  <title>Assignment 8: Reviewing Papers</title>
  <taskbody>
    <prereq>You have to be a registered TDM student.</prereq>
   <steps>
      <step>
        <cmd>accept the PC invitation, log into easychair</cmd>
        <info>You should have been given the information in the invitation e-mail</info>
      </step>
      <step>
        <cmd>indicate your conflicts of interest</cmd>
        <info>you have a conflict with anybody you have a relationship that
          would keep you from being objective (yourself, your family members,
          loved/hated ones, group members,... be honorable)
        </info>
        <stepresult>
          <p>The system records a list of conflicted paper and will not show you anything about them.</p>
        </stepresult>
      </step>
    </steps>
  </taskbody>
</task>
```

A DITA Map File

▷ **Definition 115** A DITA map combines DITA topics and maps into a document by transclusion.

▷ **Example 116** `<map>`

```
<map>
  <title>Life as an Academic</title>
  <topicmeta>...</topicmeta>
  <topicref href="introduction.dita" collection-type="sequence">
    <topicref href="conference.dita"/>
    <topicref href="TDMassignment8.dita"/>
  </topicref>
  <reltable>
    <relcell>conference.dita</relcell>
    <relcell>TDMassignment8.dita</relcell>
  </reltable>
</map>
```


14 The Semantic Web

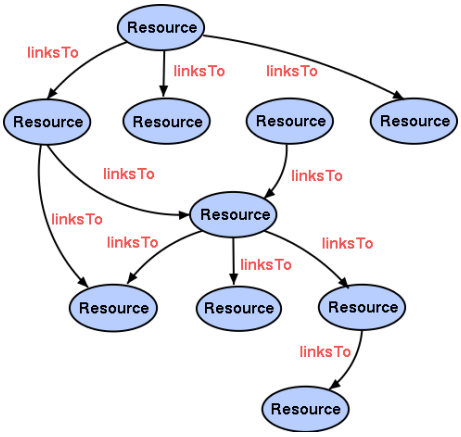
The Current Web

▷ Resources: identified by URI's, untyped

▷ Links: `href`, `src`, ... limited, non-descriptive

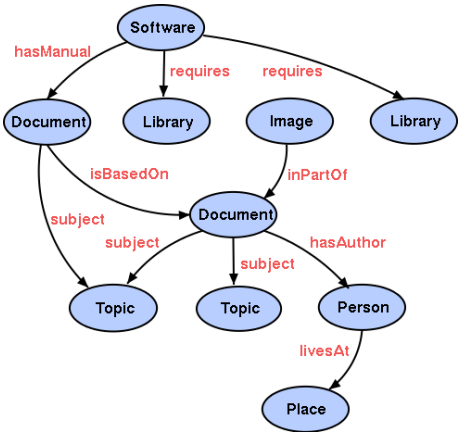
▷ User: Exciting world - semantics of the resource, however, gleaned from content

▷ Machine: Very little information available - significance of the links only evident from the context around the anchor.



The Semantic Web

- ▷ Resources: Globally Identified by URI's or Locally scoped (Blank), Extensible, Relational
- ▷ Links: Identified by URI's, Extensible, Relational
- ▷ User: Even more exciting world, richer user experience
- ▷ Machine: More processable information is available (Data Web)
- ▷ Computers and people: Work, learn and exchange knowledge effectively



What is the Information a User sees?

WWW2002
The eleventh international world wide web conference
Sheraton waikiki hotel
Honolulu, hawaii, USA
7-11 may 2002
1 location 5 days learn interact

Registered participants coming from
australia, canada, chile denmark, france, germany, ghana, hong kong, india, ireland, italy, japan, malta, new zealand, the netherlands, norway, singapore, switzerland, the united kingdom, the united states, vietnam, zaire

On the 7th May Honolulu will provide the backdrop of the eleventh international world wide web conference. This prestigious event ?
Speakers confirmed
Tim Berners-Lee: Tim is the well known inventor of the Web, ?
Ian Foster: Ian is the pioneer of the Grid, the next generation internet ?

What the machine sees

$\mathcal{WWW}\in\mathit{II}\in$
$\mathcal{T}\langle\rceil\rceil\updownarrow\rceil\sqsubseteq\rceil\backslash\sqcup\langle\rangle\backslash\sqcup\rceil\nabla\backslash\dashv\sqcup\rangle\imath\backslash\dashv\updownarrow\sqsupseteq\imath\nabla\updownarrow\lceil\supseteq\rangle\lceil\sqcap\rceil\sqsupseteq\rceil\bigsqcup\imath\backslash\{\rceil\nabla\rceil\backslash\rfloor\rceil$
$\mathcal{S}\langle\rceil\nabla\dashv\sqcup\imath\backslash\sqsupseteq\dashv\rangle\|\|\rangle\langle\imath\sqcup\rceil\updownarrow$
$\mathcal{H}\imath\backslash\imath\updownarrow\sqcap\updownarrow\sqcap\leftrightarrow\langle\dashv\sqsupseteq\dashv\rangle\rangle\Leftrightarrow\mathcal{USA}$
$\nwarrow\infty\infty\Updownarrow\dashv\dagger\in\mathit{II}\in$

$\mathcal{R}\rceil\}\rangle\int\sqcup\rceil\nabla\rceil\lceil\sqrt{}\dashv\nabla\sqcup\rangle\rfloor\rangle\sqrt{}\dashv\backslash\sqcup\int\rfloor\imath\Updownarrow\backslash\}\{\nabla\imath\Updownarrow$
$\dashv\sqcap\int\sqcup\nabla\dashv\updownarrow\rangle\dashv\Leftrightarrow\rfloor\dashv\backslash\dashv\lceil\dashv\Leftrightarrow\rfloor\langle\rangle\updownarrow\rceil\bigsqcup\backslash\Updownarrow\dashv\nabla\|\Leftrightarrow\{\nabla\dashv\backslash\rfloor\rceil\Leftrightarrow\}\rceil\nabla\Updownarrow\dashv\backslash\dagger\Leftrightarrow\}\langle\dashv\backslash\dashv\Leftrightarrow\langle\imath\backslash\}\|\imath\backslash\}\Leftrightarrow\rangle\backslash\lfloor\rangle\dashv\Leftrightarrow$
$\rangle\nabla\rceil\updownarrow\dashv\backslash\lceil\Leftrightarrow\rangle\sqcup\dashv\updownarrow\dagger\Leftrightarrow|\dashv\sqrt{}\dashv\backslash\Leftrightarrow\Updownarrow\dashv\updownarrow\sqcup\dashv\Leftrightarrow\backslash\rfloor\sqsupseteq\rceil\dashv\updownarrow\dashv\backslash\lceil\Leftrightarrow\backslash\imath\nabla\sqsupseteq\dashv\dagger\Leftrightarrow$
$\int\rangle\backslash\}\dashv\sqrt{}\imath\nabla\rceil\Leftrightarrow\int\sqsupseteq\rangle\sqcup\ddagger\rceil\nabla\updownarrow\dashv\backslash\lceil\Leftrightarrow\sqcup\langle\rceil\sqcap\backslash\rangle\sqcup\rceil\lceil\|\|\rangle\backslash\}\lceil\imath\Updownarrow\Leftrightarrow\sqcup\langle\rceil\sqcap\backslash\rangle\sqcup\rceil\lceil\int\sqcup\dashv\sqcup\rceil\int\Leftrightarrow\sqsubseteq\rangle\rceil\sqcup\backslash\dashv\Updownarrow\Leftrightarrow\ddagger\dashv\rangle\nabla\rceil$

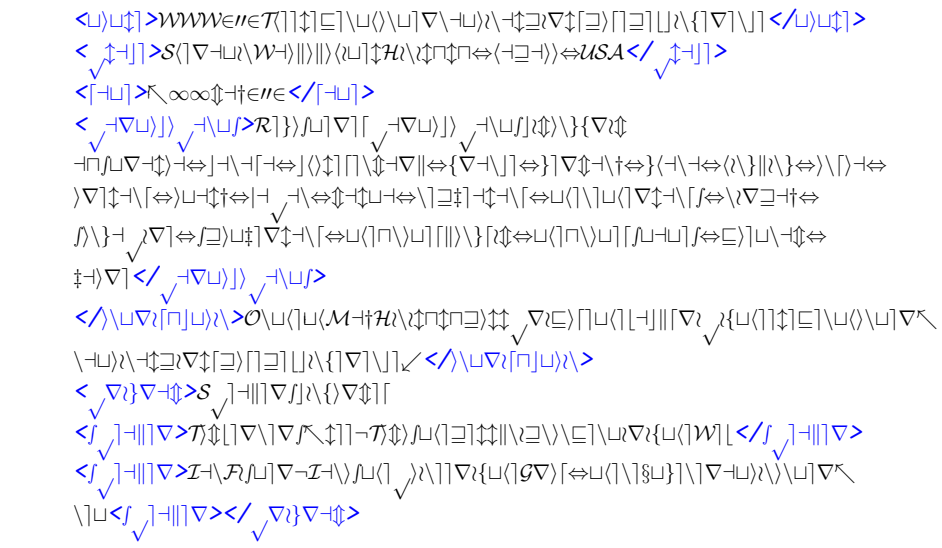
$\mathcal{O}\backslash\sqcup\langle\rceil\sqcup\langle\mathcal{M}\dashv\dagger\mathcal{H}\imath\backslash\imath\updownarrow\sqcap\updownarrow\sqcap\sqsupseteq\rangle\updownarrow\updownarrow\sqrt{}\nabla\imath\sqsubseteq\rangle\bigsqcup\sqcup\langle\rceil\lceil\dashv\rfloor\|\|\nabla\imath\sqrt{}\imath\{\sqcup\langle\rceil\updownarrow\rceil\sqsubseteq\rceil\backslash\sqcup\langle$
$\rangle\backslash\sqcup\rceil\nabla\backslash\dashv\sqcup\rangle\imath\backslash\dashv\updownarrow\sqsupseteq\imath\nabla\updownarrow\lceil\sqsupseteq\rangle\bigsqcup\rceil\sqsupseteq\rceil\bigsqcup\rfloor\imath\backslash\{\rceil\nabla\rceil\backslash\rfloor\rceil\swarrow\mathcal{T}\rangle\int\sqrt{}\nabla\rceil\int\sqcup\rangle\}\rangle\imath\sqcap\int\sqsubseteq\rceil\backslash\sqcup\perp$

$\mathcal{S}\sqrt{}\rceil\dashv\|\|\rceil\nabla\int\rfloor\imath\backslash\{\rangle\nabla\updownarrow\rceil\lceil$
$\mathcal{T}\rangle\updownarrow\lfloor\rceil\nabla\backslash\rceil\nabla\int\nwarrow\updownarrow\rceil\rceil\neg\mathcal{T}\rangle\updownarrow\int\sqcup\langle\rceil\sqsupseteq\rceil\updownarrow\updownarrow\|\backslash\imath\sqsupseteq\backslash\rangle\backslash\sqsubseteq\rceil\backslash\sqcup\imath\nabla\imath\{\sqcup\langle\rceil\mathcal{W}\rceil\lfloor\Leftrightarrow\perp$
$\mathcal{I}\dashv\backslash\mathcal{F}\int\sqcup\rceil\nabla\neg\mathcal{I}\dashv\backslash\rangle\int\sqcup\langle\rceil\sqrt{}\imath\backslash\rceil\rceil\nabla\imath\{\sqcup\langle\rceil\mathcal{G}\nabla\rangle\lceil\Leftrightarrow\sqcup\langle\rceil\backslash\rceil\S\sqcup\}\rceil\backslash\rceil\nabla\dashv\sqcup\rangle\imath\backslash\rangle\backslash\sqcup\rceil\nabla\backslash\rceil\sqcup\perp$

Solution: XML markup with "meaningful" Tags


```
<title>WWW∈II∈T⟨]]‡]⊑]\⊔⟨⟩\⊔]∇\⊣⊔⟩ı\⊣‡⊒ı∇‡⌈⊒⟩⌋⊓⊒]⌊⟩ı\{]∇]\⌋]</title>
<place>S⟨]∇⊣⊔ı\W⊣⟩‖⟩⟨ı⊔]‡Hı\ı‡⊓‡⊓⇔⟨⊣⊒⊣⟩⟩⇔USA</place>
<date>↖∞∞⇕⊣†∈II∈</date>
<participants>R]}⟩∫⊔]∇]⌈√⊣∇⊔⟩⌋⟩√⊣\⊔∫⌋ı⇕⟩\}{∇ı⇕
⊣⊓∫⊔∇⊣‡⟩⊣⇔⌋⊣\⊣⌈⊣⇔⌋⟨⟩‡]⊓⌋\⇕⊣∇‖⇔{∇⊣\⌋]⇔}]∇⇕⊣\†⇔}⟨⊣\⊣⇔⟨ı\}‖ı\}⇔⟩\⌋⊣⇔
⟩∇]‡⊣\⌈⇔⟩⊔⊣‡†⇔|⊣√⊣\⇔⇕⊣‡⊔⊣⇔\⌋⊒‡]⊣‡⊣\⌈⇔⟨⌋⊓⌋⟨]∇‡⊣\⌈∫⇔ı∇⊒⊣†⇔
∫⟩\}⊣√ı∇]⇔∫⊒⟩⊔‡]∇‡⊣\⌈⇔⊔⟨]⊓\⟩⊔]⌈‖‖⟩\}⌈ı⇕⇔⊔⟨]⊓\⟩⊔]⌈∫⊔⊣⊔]∫⇔⊑⟩]⊔\⊣⇕⇔‡⊣⟩∇]
⊔]∇\⊣⊔⟩ı\⊣‡⊒ı∇‡⌈⊒⟩⌋⊓⊒]⌊⟩ı\{]∇]\⌋]↙</introduction>
<program>S√]⊣‖]∇∫⌋ı\{⟩∇⇕⌈
<speaker>T⟩⇕⌊]∇\⌋∇∫↖‡]]¬T⟩⇕⟩∫⊔⟨]⊒]‡‡‖\ı⊒\⟩\⊑]\⊔ı∇ı{⊔⟨]W]⌊</speaker>
<speaker>I⊣\F⟩∫⊔]∇¬I⊣\⟩∫⊔⟨]√⟩ı\]]∇ı{⊔⟨]G∇⟩⌈⇔⊔⟨]\]§⊔}]\]∇⊣⊔⟩ı\⟩\⊔]∇↖
\]⊔</speaker></program>
```

What the machine sees of the XML



Need to add "Semantics"

- ▷ External agreement on meaning of annotations E.g., Dublin Core
  - ▷ Agree on the meaning of a set of annotation tags
  - ▷ Problems with this approach: Inflexible, Limited number of things can be expressed
- ▷ Use Ontologies to specify meaning of annotations
  - ▷ Ontologies provide a vocabulary of terms
  - ▷ New terms can be formed by combining existing ones
  - ▷ Meaning (semantics) of such terms is formally specified
  - ▷ Can also specify relationships between terms in multiple ontologies

‘

15 Introduction to Knowledge Representation

What is knowledge? Why Representation?

▷ For the purposes of this course: Knowledge is the information necessary to support intelligent reasoning (during NLP)

representation	can be used to determine
set of words	whether a word is admissible
list of words	the rank of a word
a lexicon	translation or grammatical function
structure	function

▷ Representation as structure and function.

- ▷ the representation determines the content theory (what is the data?)
- ▷ the function determines the process model (what do we do with the data?)

Knowledge Representation vs. Data Structures

▷ Why do we use the term "knowledge representation" rather than

- ▷ data structures? (sets, lists, ... above)
- ▷ information representation? (it is information)

▷ no good reason other than AI practice, with the intuition that

- ▷ data is simple and general (supports many algorithms)
- ▷ knowledge is complex (has distinguished process model)

Some Paradigms for AI/NLP

▷ GOFAI (good old-fashioned AI)

- ▷ symbolic knowledge representation, process model based on heuristic search

▷ statistical, corpus-based approaches.

- ▷ symbolic representation, process model based on machine learning
- ▷ knowledge is divided into symbolic- and statistical (search) knowledge

▷ connectionist approach (not in this course)

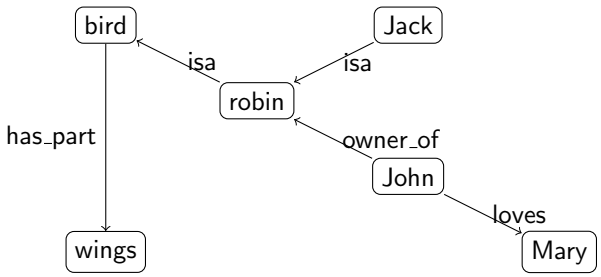
- ▷ sub-symbolic representation, process model based on primitive processing elements (nodes) and weighted links
- ▷ knowledge is only present in activation patters, etc.

KR Approaches/Evaluation Criteria

- ▷ Expressive Adequacy: What can be represented, what distinctions are supported.
- ▷ Reasoning Efficiency: can the representation support processing that generates results in acceptable speed?
- ▷ Primitives: what are the primitive elements of representation, are they intuitive, cognitively adequate?
- ▷ Meta-representation: knowledge about knowledge
- ▷ Incompleteness: the problems of reasoning with knowledge that is known to be incomplete.

Semantic Networks [e.g. Collins and Quillian '69]

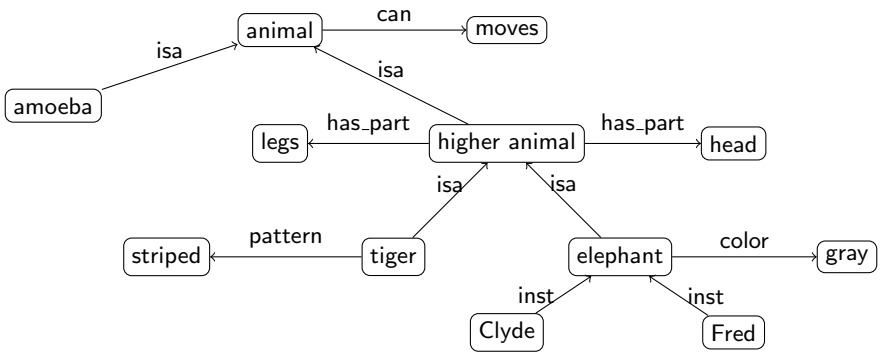
- ▷ Graph structure with for representing knowledge
  - ▷ nodes represent concepts (e.g. bird, John, robin)
  - ▷ links represent relations between these (isa, father_of, belongs_to)



The famous "Isa-Hierarchy"

▷ Idea: encode taxonomic information about concepts and individuals

- ▷ in "isa" links (inclusion of concepts)
- ▷ in "inst" links (concept memberships)
- ▷ use property inheritance in the process model

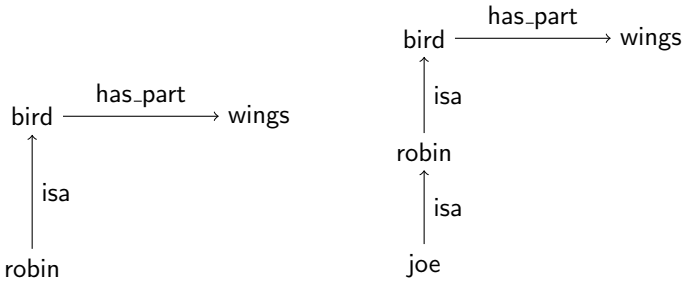


Limitations of Semantic Networks

▷ What is the meaning of a link?

- ▷ link names are very suggestive (misleading for humans)
- ▷ meaning of link types defined in the process model (no denotational semantics)

▷ No division of optional and defining arguments



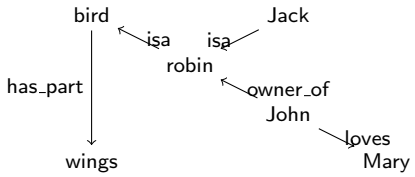
4

[d] EdNote: with a cancel link link to the has link

Another Notation for Semantic Networks

▷ use function/argument notation

- ▷ Interpret nodes as arguments (reification to individuals)
- ▷ Interpret links as functions (logical relations)



isa(robin,bird)
haspart(bird,wings)
isa(Jack,robin)
owner_of(John, robin)
loves(John,Mary)

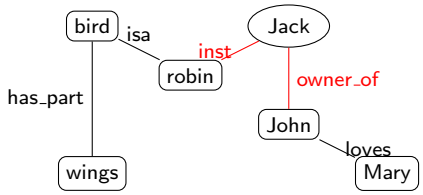
\+ linear notation (equivalent, but better to implement on a computer)

\+ easy to give process model by deduction (e.g. PROLOG)

– worse locality properties (networks are associative)

A Denotational Semantics for Semantic Networks

▷ take isa/inst concept/individual distinction into account



$robin \subseteq bird$
haspart(bird,wings)
$Jack \in robin$
owner_of(John, Jack)
loves(John,Mary)

▷ looks like first-order logic, if we take

- ▷ $A \subseteq B$ to mean $\forall X.A(X) \Rightarrow B(X)$
- ▷ $a \in S$ to mean $S(a)$
- ▷ $haspart(A,B)$ to mean $\forall X.A(X) \Rightarrow (\exists Y.B(Y) \wedge part\_of(X,Y))$

▷ Take first-order deduction as process model (gives inheritance for free)

Frame Notation as Logic with Locality

▷ Predicate Logic: (where is the locality?)

$catch\_22 \in catch\_object$	There is an instance of catching
$catcher(catch\_22, jack\_2)$	Jack did the catching
$caught(catch\_22, ball\_5)$	He caught a certain ball

▷ Frame Notation (group everything around the object)

```
(catch_object  catch_22
               (catcher jack_2)
               (caught ball_5))
```

+ Once you have decided on a frame, all the information is local

+ easy to define schemes for concepts (aka. types in feature structures)

– how to determine frame, when to choose frame (log/chair)

KR involving Time (Scripts [Shank '77])

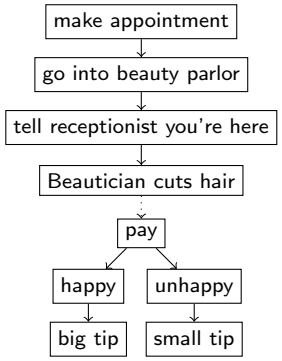
▷ Idea: organize typical event sequences, actors and props into representation structure

▷ **Example 117** getting your hair cut (at a beauty parlor)

- ▷ props, actors as "script variables"
- ▷ events in a (generalized) sequence

▷ use script material for

- ▷ anaphors, bridging references
- ▷ default common ground
- ▷ to fill in missing material into situations



Other Representation Formats (not covered)

▷ Procedural Representations (production systems)

▷ analogical representations (interesting but not here)

▷ iconic representations (interesting but very difficult to formalize)

▷ If you are interested, come see me off-line

16 Description Logics and the Semantic Web

Resource Description Framework

▷ **Definition 118** The Resource Description Framework (RDF) is a framework for describing resources on the web. It is a XML vocabulary developed by the W3C.

▷ Note: RDF is designed to be read and understood by computers, not to be being displayed to people

▷ **Example 119** RDF can be used for describing

- ▷ properties for shopping items, such as price and availability
- ▷ time schedules for web events
- ▷ information about web pages (content, author, created and modified date)
- ▷ content and rating for web pictures
- ▷ content for search engines
- ▷ electronic libraries

Resources and URIs

▷ RDF describes resources with properties and property values.

▷ RDF uses Web identifiers (URIs) to identify resources.

▷ **Definition 120** A resource is anything that can have a URI, such as `http://www.jacobs-university.de`

▷ **Definition 121** A property is a resource that has a name, such as *author* or *homepage*, and a property value is the value of a property, such as *Michael Kohlhase* or `http://kwarc.info/kohlhase` (a property value can be another resource)

▷ **Definition 122** The combination of a resource, a property, and a property value forms a statement (known as the subject, predicate and object of a statement).

▷ **Example 123** Statement: *The [author]^{pred} of [this slide]^{subj} is [Michael Kohlhase]^{obj}*

XML Syntax for RDF

▷ RDF is a concrete XML vocabulary for writing statements

▷ **Example 124** The following RDF document could describe the slides as a resource

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc= "http://purl.org/dc/elements/1.1/">
  <rdf:Description about="https://svn.kwarc.info/.../slides/kr/en/rdf.tex">
    <dc:creator>Michael Kohlhase</dc:creator>
    <dc:source>http://www.w3schools.com/rdf</dc:source>
  </rdf:Description>
</rdf:RDF>
```

▷

This RDF document makes two statements:

- ▷ The subject of both is given in the about attribute of the `rdf:Description` element
- ▷ The predicates are given by the element names of its children
- ▷ The objects are given in the elements as URIs or literal content.

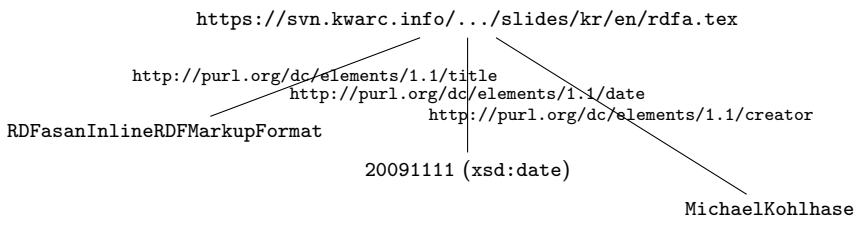
Intuitively: RDF is a way to write down ABox information in a web-scalable way.

RDFa as an Inline RDF Markup Format

▷ Problem: RDF is a standoff markup format (annotate by URIs pointing into other files)

▷ **Example 125**

```
<div xmlns:dc="http://purl.org/dc/elements/1.1/">
    <h2 property="dc:title">RDF as an Inline RDF Markup Format</h2>
    <h3 property="dc:creator">Michael Kohlhase</h3>
    <em property="dc:date" datatype="xsd:date"
        content="20091111">November 11., 2009</em>
</div>
```



OWL as an Ontology Language for the Semantic Web

▷ Idea: Use Description Logics to talk about RDF triples.

▷ An RDF triple is an ABox entry for a role contraint hRs

▷ **Example 126** h is the resource for Ian Horrocks, s is the resource for Ulrike Sattler, and R is the the relation "hasColleague" in

```
<rdf:Description about="some.uri/person/ian_horrocks">
  <hasColleague resource="some.uri/person/uli_sattler"/>
</rdf:Description>
```

Idea: Now collect similar resources in *classes*, and state rules about them in a way, so that we can use inference to make kwnowledge explicit that was implicit before (saves us lots of work!)

▷▷ Idea: We know how to do this, this is just $\mathcal{ALC}$+!!!

The OWL Language

▷ Three species of OWL

- ▷ OWL Full is union of OWL syntax and RDF
- ▷ OWL DL restricted to FOL fragment
- ▷ OWL Lite is "easier to implement" subset of OWL DL

▷ Semantic layering

- ▷ OWL DL $\hat{=}$ OWL Full within DL fragment
- ▷ DL semantics officially definitive
- ▷ OWL DL based on SHIQ Description Logic($\mathcal{ALC}$ + nubmer restrictions, transitive roles, inverse roles, role inclusion
- ▷ OWL DL benefits from many years of DL research
- ▷ Well defined semantics, formal properties well understood (complexity, decidability)
- ▷ Known reasoning algorithms, Implemented systems (highly optimized)

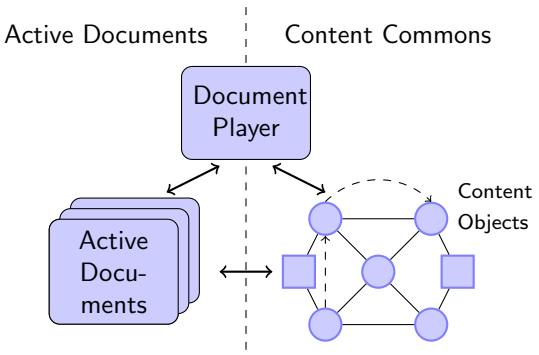
17 Planetary: A Social Semantic eScience System

The PLANETARY System

- The PLANETARY system is a Web 3.0 system for semantically annotated document collections in Science, Technology, Engineering and Mathematics (STEM).
- Web 3.0 stands for extension of the Social Web with Semantic Web/Linked Open Data technologies.
- documents published in the PLANETARY system become flexible, adaptive interfaces to a content commons of domain objects, context, and their relations.
- PLANETARY is based on the Active Documents Paradigm (see next)
- **Example 127 (Example installments)**
  - `arxivdemo.mathweb.org` (presentation/structural Level: arXiv)
  - `gencs.kwarc.info` (semantic level: PantaRhei course system)
  - `logicatlas.omdoc.org` (fully formal level: Logic Representations)
  - `planetbox.kwarc.info` (Technology Sandbox)
- The PLANETARY system is finalist in the Elsevier Executable Papers Challenge.

The Active Documents Paradigm

- **Definition 128** The active documents paradigm (ADP) consists of
  - *semantically annotated documents* together with
  - background ontologies (which we call the content commons),
  - *semantic services* that use this information
  - a document player application tha embeds services to make documents executable.

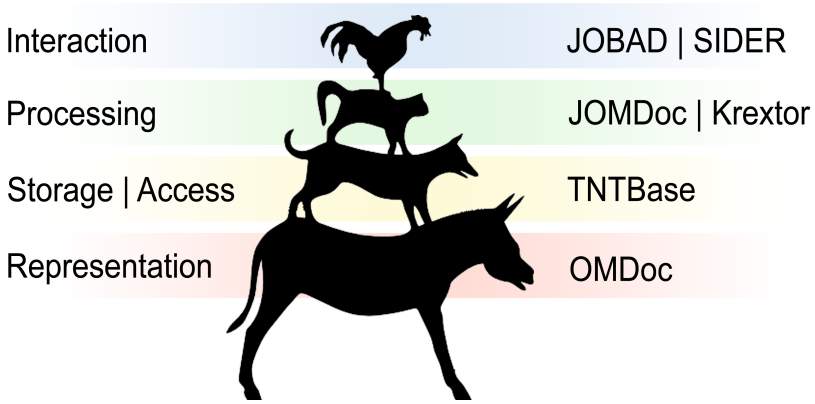


- **Example 129** Services can be program (fragment) execution, computation, visualization, navigation, information aggregation and information retrieval

18 Realizing Planetary

Realizing PLANETARY: The KWARC stack

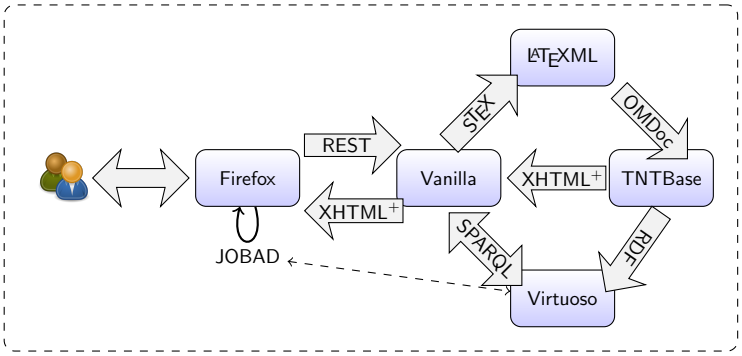
We have already developed the necessary tools/systems over the last decade



PLANETARY is the ideal test bed to integrate them.

Assembling PLANETARY: System Architecture

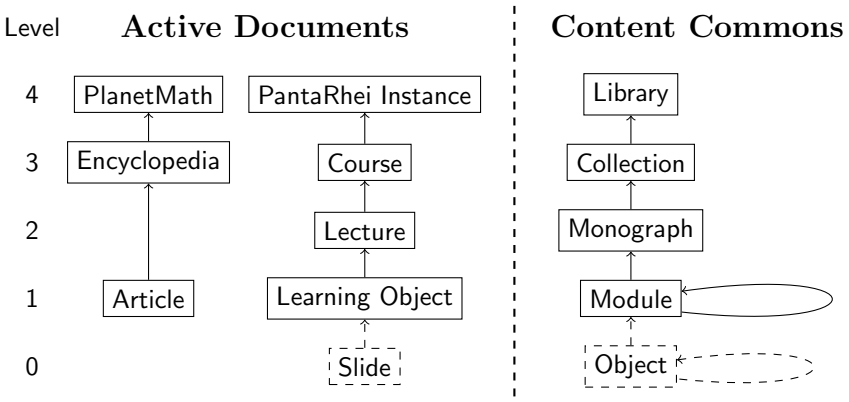
▷ PLANETARY functionality can be achieved by integrating existing components.



- ▷ Vanilla for discussions, user management, caching, (standard forum [?])
- ▷ TNTBase for versioned XML storage, OMDoc presentation
- ▷ JOBAD integrates semantic services into documents
- ▷ Virtuoso is a triple store for semantic relations
- ▷ LaTeXML transforms LaTeX/sTeX to XHTML+MathML+RDFa

Layers of Documents/Content

▷ Content and narrative structures come at different conceptual layers



▷ Different layers support different functionality

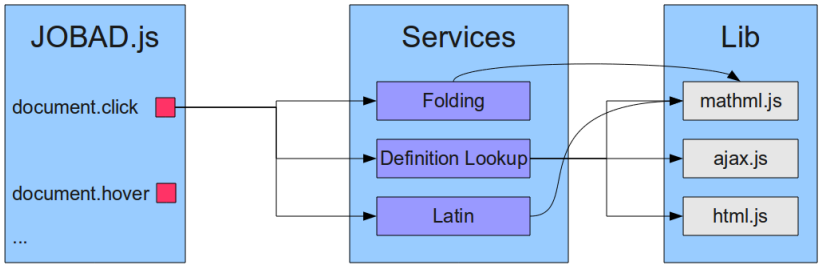
JOBAD: Embedding Semantic Services into Web Docs

▷ JavaScript API for (J)OMDoc Based Active Documents

▷ runs inside client browser (Firefox currently)

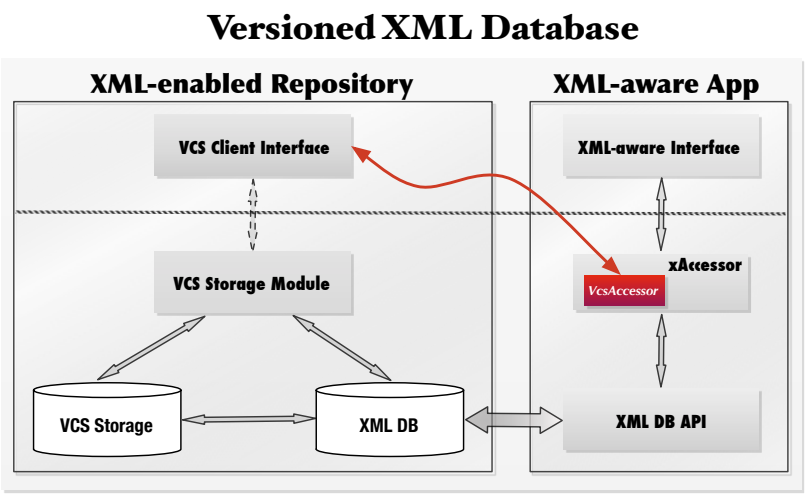
▷ provides client-only or server-based features (extensible framework)
based on semantic annotations in XHTML+MathML+RDFa documents

▷ Project home page: `https://jomdoc.omdoc.org/wiki/JOBAD`



TNTBase: Versioned Storage for XML

▷ The TNTBase system is a versioned storage system for XML documents. It combines the functionality and interfaces of Subversion with those of an XML database.



OMDoc in a Nutshell (three levels of modeling)

Formula level: OpenMath/C-MathML

▷ Objects as logical formulae

▷ semantics by ref. to theory level

```
<OMA>
 <OMS cd="arith1" name="plus"/>
 <OMS cd="nat" name="zero"/>
 <OMV name="N"/>
</OMA>
```

Statement level:

▷ Definition, Theorem, Proof, Ex.

▷ semantics explicit forms and refs.

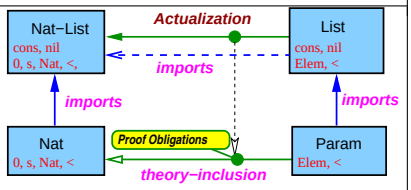
```
<defn for="plus" type="rec">
 <CMP>rec. eq. for plus</CMP>
 <FMP>X + 0 = X</FMP>
 <FMP>X + s(Y) = s(X + Y)</FMP>
</defn>
```

Theory level: Development Graph

▷ inheritance via symbol-mapping

▷ theory-inclusion by proof-obligations

▷ local (one-step) vs. global links



LaTeXML: Converting TeX/LaTeX Documents to XML

▷ **Definition 130** LaTeXML converts LaTeX documents to XHTML+MathML

- ▷ re-implement the TeX parser in `perl`. (do not expand semantic macros)
- ▷ needs LaTeXML bindings for all LaTeX packages and classes (specify the XML for the emitter)

▷ Case Study: Converting the arXiv into XHTML+MathML (70% coverage of 550 k documents)

sTeX, a Semantic Variant of TeX/LaTeX

▷ Problem: Need content markup formats for semantic services, but Mathematicians write LaTeX

▷ Idea: Enable the author to make structure explicit and disambiguate meanings

- ▷ use the TeX macro mechanism for this (well established)
- ▷ the author knows the semantics best (at least she understands)
- ▷ the burden is is alleviated by manageability savings (MKM on TeX/LaTeX)

▷ **Definition 131 (sTeX Approach)** Semantic pre-loading of TeX/LaTeX documents.

- ▷ Introduce semantic macros: e.g. `\union{a,b,c}` $\rightsquigarrow a \cup b \cup c$
- ▷ Mark up discourse structure: (largely invisible)
  e.g. `\begin{sproof}[id=Wiles,for=Fermat]...\end{sproof}`
- ▷ Generate PDF and OMDoc from that (via LaTeXML [?])

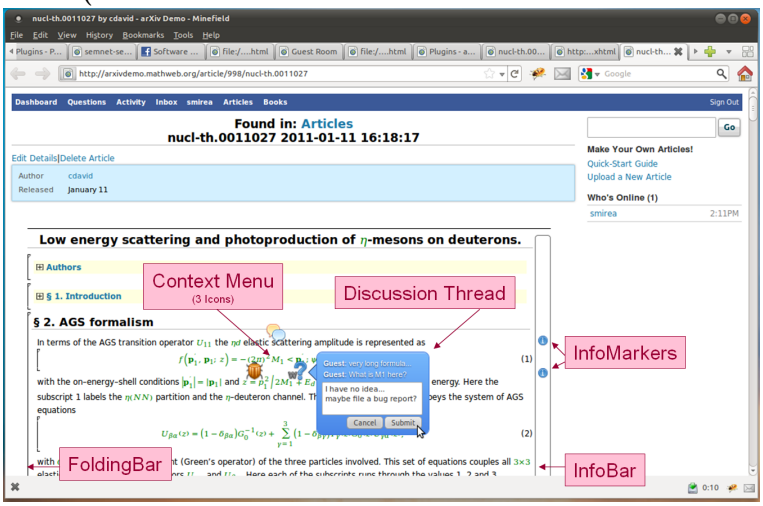
`http://trac.kwarc.info/sTeX/`

19 Levels of Service in Planetary

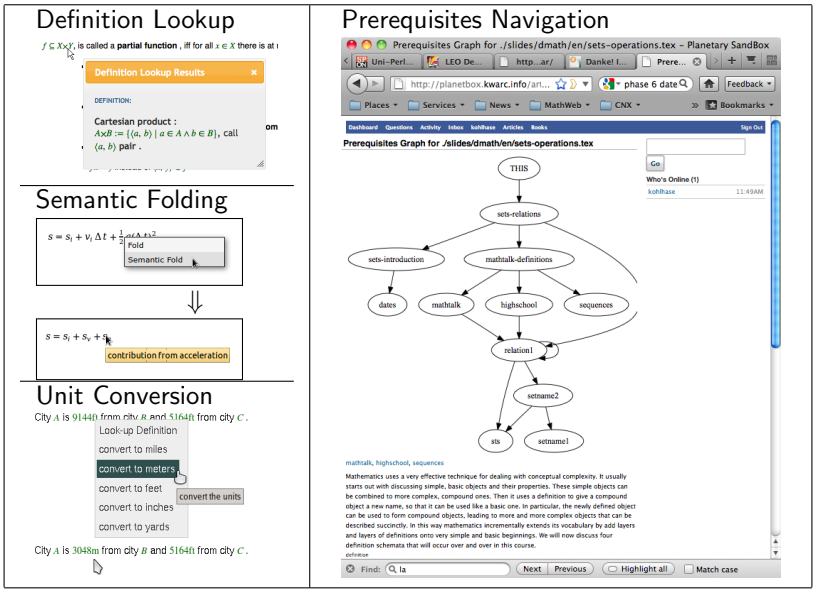
PLANETARY at the Presentation/Structural Level

▷ PLANETARY can make use objects and relations at various levels,

▷ **Example 132 (arXivdemo: Document Structure and Presentational Math)**

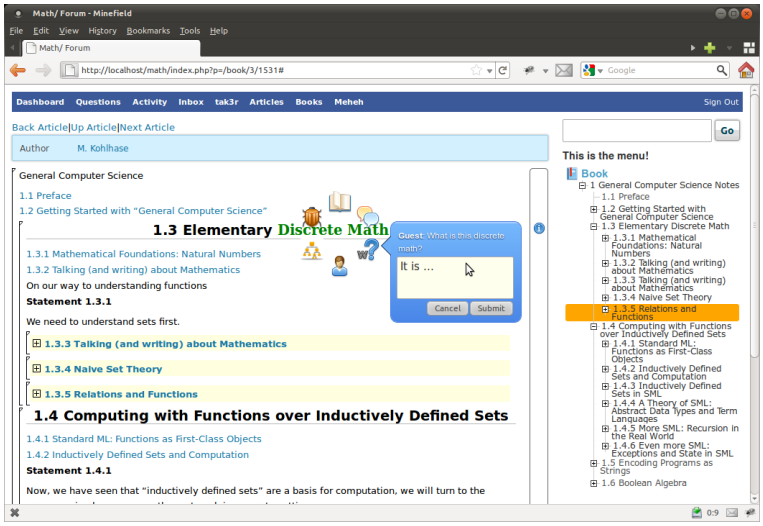


User Services at the Semantic Level in PLANETARY



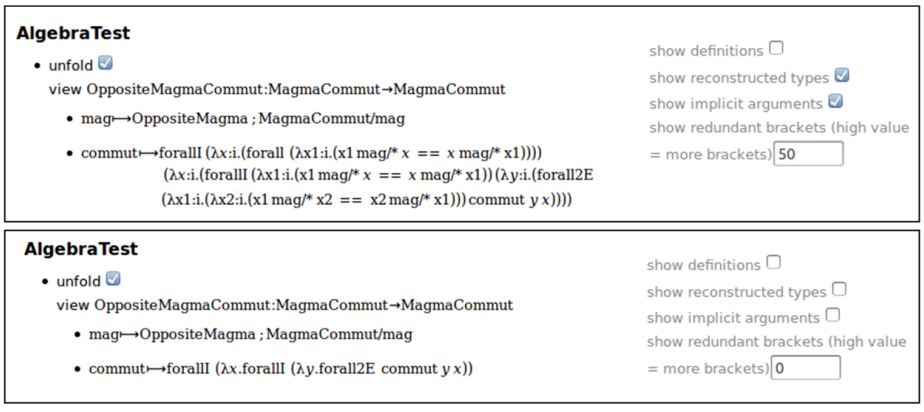
PantaRhei: Semantic Course Knowledge Exploration

▷ PantaRhei is a semantic course knowledge exploration system based on the PLANETARY system.



User Services at the Formal Level in PLANETARY

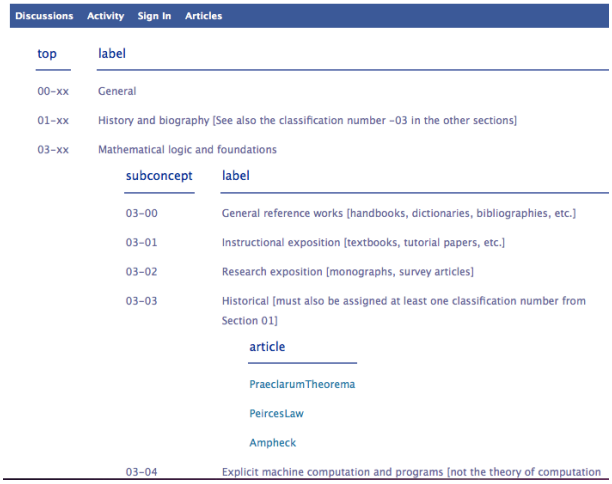
▷ Formal Representations Adapted to Distinct User Settings (Customized via the Dashboard Widget on the Right)



Accessing Encyclopedias via Ontologies

▷ Idea: add classification metadata to articles, harvest as RDF into triplestore, compute access methods via SPARQL queries and SKOS ontology.

▷ **Example 133 (MSC View in PlanetMath)** use the Math Subject Classification



References

[BCHL09] Bert Bos, Tantek Celik, Ian Hickson, and Høakon Wium Lie. Cascading style sheets level 2 revision 1 (CSS 2.1) specification. W3C Candidate Recommendation, World Wide Web Consortium (W3C), 2009.

[BLFM05] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. Uniform resource identifier (URI): Generic syntax. RFC 3986, Internet Engineering Task Force (IETF), 2005.

[Dav67] Donald Davidson. Truth and meaning. *Synthese*, 17, 1967.

[ECM09] ECMAScript language specification. ECMA Standard ECMA-262, ECMA International, December 2009. 5th Edition.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force (IETF), 1999.

[HL11] Martin Hilbert and Priscila López. The world's technological capacity to store, communicate, and compute information. *Science*, 331, feb 2011.

[Kay08] Michael Kay. Saxonica: XSLT and XQuery processing. `http://www.saxonica.com`, 2008.

[Koh08] Michael Kohlhase. Using LaTeX as a semantic markup format. *Mathematics in Computer Science*, 2(2):279–304, 2008.

[Koh10] Michael Kohlhase. sTeX: Semantic markup in TeX/LaTeX. Technical report, Comprehensive TeX Archive Network (CTAN), 2010.

[Mil] Bruce Miller. LaTeXML: A LaTeX to XML converter. Web Manual at http://dlmf.nist.gov/LaTeXML/. seen May2010.

[RHJ98] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. HTML 4.0 Specification. W3C Recommendation REC-html40, World Wide Web Consortium (W3C), April 1998.

[Van] Vanilla forums. project page at http://vanillaforums.org. visited April 2011.

[Vei] Daniel Veillard. The xslt c library for gnome; the xsltproc tool. System Home page at http://xmlsoft.org/XSLT/xsltproc2.html.

[Vol11] Victor Volkman. Classic parsing with flex and bison. http://www.codeguru.com/csharp/.net/net_general/patterns/article.php/c12805__2/Classic-Parsing-with-Flex-and-Bison.htm, 2011. visited Feb 2011.

[Wik11] Wikipedia. Epub — wikipedia, the free encyclopedia, 2011. [Online; accessed 15-March-2011].

[XML] Extensible Markup Language (XML) 1.0 (Fourth Edition). Web site at http://www.w3.org/TR/REC-xml/.

Index