Capturing the Content of (Computer) Science

Scaling Up without Dumbing Down

MICHAEL KOHLHASE

School of Engineering & Science
International University Bremen

School of Computer Science
Carnegie Mellon University

http://www.faculty.iu-bremen.de/mkohlhase

(Re)-Use of academic content in (Computer) Science

"Universities generate content every day through their courses and seminars. Then they throw it away. There is a certain charm with this approach, but it is not cost effective. Universities operate like renaissance quartets based on live performances. . . . Content storage and reuse are also important to test and ameliorate performance and to generate an institutional memory."

Dennis Tsichritzis, "Reengineering the University," CACM June 1999.

- This gripe is only meant for educational material (but applies throughout)
- Can we do something about it? (on a technological level?)
- Idea: apply (Computer) Science & Engineering techniques (content markup)
  - capture the logical structure of the material (seen as data)
  - as a basis for re-use and services (high-level system support)

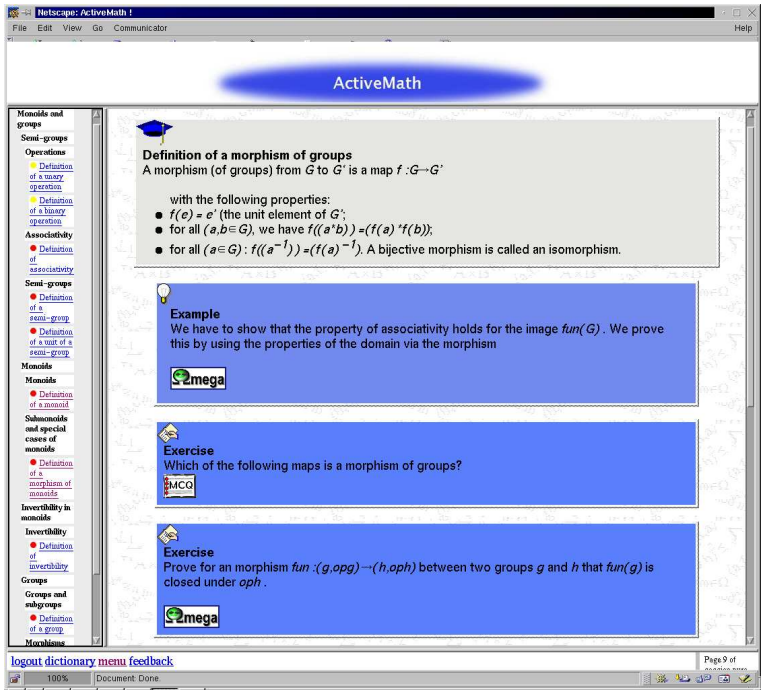
Why are we talking about this now?

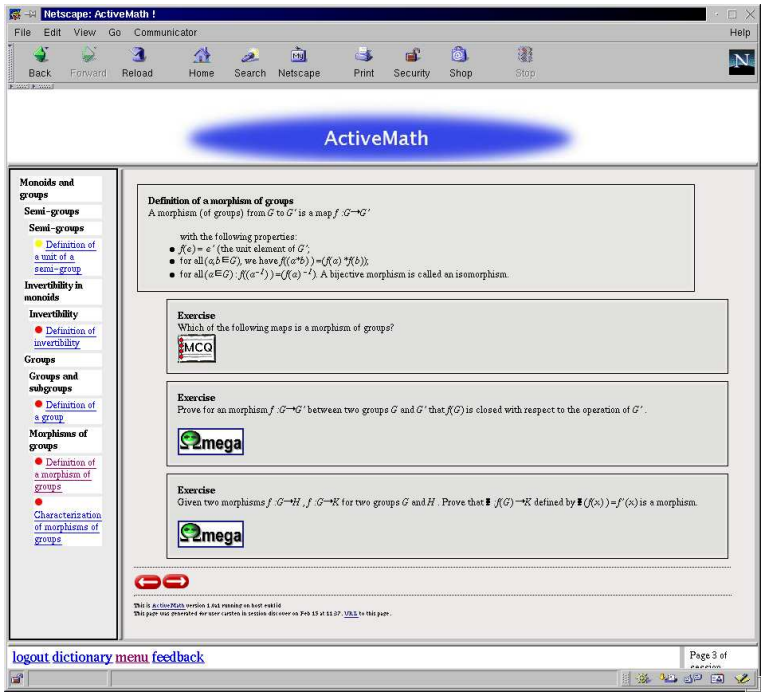
- We are already using some of content-based methods in CS,
  - Formal methods (program analysis/verification/synthesis)
  - Software engineering (CASE tools)
  - Simulation (instead of Experiments)
- But: this does not cover mainstream Sci&Eng practice/education.
- Why now? We are building a content-based infrastructure for the Internet.
  - Automated Reasoning and Knowledge Representation is at the center
  - XML as a common language family basis. (universal tool support.)
  - "vertical standards" for resp. disciplines (e.g. MATHML, CHEMML)
  - Challenge: scale up content methods to cover the WWWeb.
- Scaling Up without Dumbing Down (the meat is in the interfaces)

Experiment in CS Education: CMU Course Capsules

- Formal representation in OMDoc/MBase is a lot of work, but: can be the basis for added-value services that offer
  - Efficient retrieval/searching of content (via MBase)
  - Reuse of content, use of other external systems
- dynamic generation of mathematical documents depending on preferences and abilities of the user (user model referencing MBase)
  - Guided Tours: e.g. transitive hull of the dependency relation (Hi, I am Michael, I am interested in the fundamental theorem of algebra)
  - Overview on a certain subject (explicit structure)
  - Preparation for exam (examples, problems present)
  - Follow/Review a predefined course (complex document structure)

Presentation for Novices and Advanced





- what can be adapted? (via user model)
  - content selection, examples, exercises; availability of systems
  - presentation: level of abstraction, notation, language and modality

Content vs. Presentation by Example

Format	Representation	Content?
LaTeX	`{\bf 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)

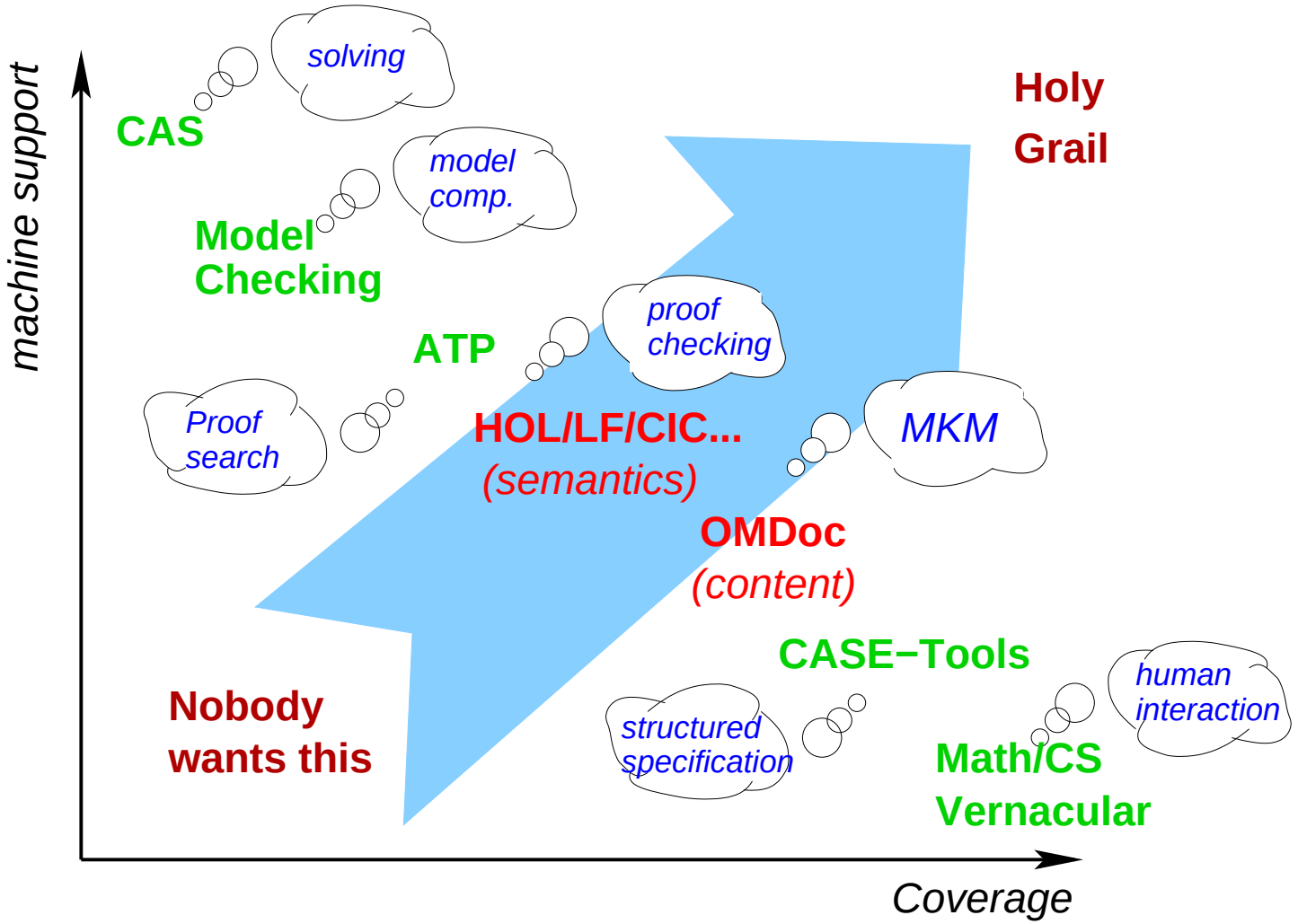
Content vs. Semantics/Formalization

- 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)

Situating OMDoc: Math Knowledge Management



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>
```

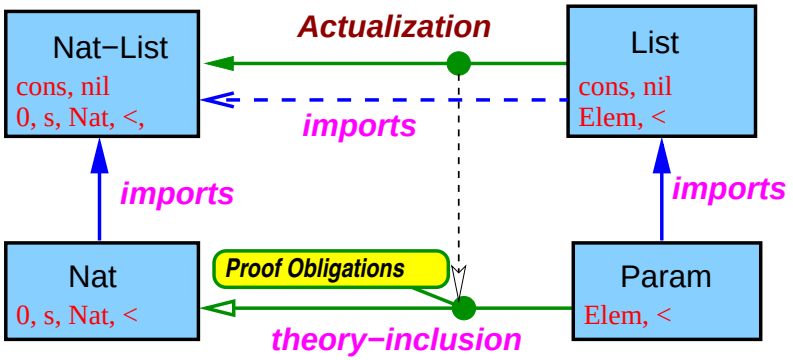
Statement level:

- Defi nition, Theorem, Proof, Example
- semantics explicit forms and refs.

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



IU B

Web Standards for Formula Markup

language	MATHML	OPENMATH
by	W3C Math WG	OPENMATH society
origin	math for HTML	integration of CAS
coverage	cont+pres; K-14	content; extensible
status	Version 2.2e (VI 2003)	standard (IV 2001)
activity	maintenance	perparing OM2
Info	`http://w3c.org/Math`	`http://www.openmath.org`

C-MathML and OpenMath are equivalent (almost)

OpenMath	MathML
`<OMBIND>`	`<apply>`
`<OMS cd="quant1" name="forall"/>`	`<forall/>`
`<OMBVAR>`	`<bvar>`
`<OMATTR>`	
`<OMATP>`	
`<OMS cd="sts" name="type"/>`	
`<OMS cd="setname1" name="N"/>`	
`</OMATP>`	
`<OMV name="a"/>`	`<ci type="nat">a</ci>`
`</OMATTR>`	
`</OMBVAR>`	`</bvar>`
`<OMA>`	`<apply>`
`<OMS cd="relation1" name="geq"/>`	`<geq/>`
`<OMV name="a"/>`	`<ci type="nat">a</ci>`
`<OMI>0</OMI>`	`<cn>0</cn>`
`</OMA>`	`</apply>`
`</OMBIND>`	`</apply>`

Motivation for CodeML Design

- Program Code contains the structured objects of CS!
- Integrated framework for content and presentation of program code.
  (like MathML is for mathematical formulae)
- Why Presentation Markup?
  - good program visualization needs syntactic/static analysis (∼IDE)
  - resources (Grammar, type-checker, libraries) don't exist in browser
  - multi-modal output (browser, PDA, braille, aural, ...)
  - infrastructure for tagging/referencing program snippets
- Why Content Markup?
  - unambiguous reference and abstract syntax
  - format/version-independent definition of program concepts

Code Markup Language (CodeML)

- Example:

```
public int find (int x) {
    if (s[x] < 0) return x;
    return  find (s[x]);}
```

Presentation	Content

`<cpg close=";" breaks="o-cb">`
`  <cpo type="built-in">return</cpo>`
`  <cpg>`
`  <cpo type="built-in>find</cpo>`
`  <cpg open="(" close=")">`
`  <cpo type="definiens">s</cpo>`
`  <cpg open="[" close="]">`
`  <cpi>x</cpi>`
`  </cpg>`
`  </cpg>`
`  </cpg>`
	`<apply>`
	`  <ccsym cd="java.proc" name="return"/>`
	`  <apply>`
	`  <ccsym cd="union-find" name="find"/>`
	`  <apply>`
	`  <ccsym cd="array" name="select"/>`
	`  <ccv name="s"/>`
	`  <ccv name="x"/>`
	`  </apply>`
	`  </apply>`
	`</apply>`

Multilingual Comments and Pseudo-code

Presentation -CodeML	Multilingual Pseudo-code

<cpg>
  <cpo>then</cpo>
  <cpo>return</cpo>
  <cpi>x</cpi>
  <cpc>
  <cpt>
  we are at the root
  </cpt>
  <cpt xml:lang="de">
  Wir sind in der Wurzel
  </cpt>
  </cpc>
</cpg>
	procedure fi nd (x)
	  if s[x] is negative
	  then return x (*we are at the root*)
	  otherwise return s[x] (*way to go!*)
	--- 	
	procedure fi nd (x)
	  if s[x] ist negativ
	  then return x (*Wir sind an der Wurzel*)
	  otherwise return s[x] (*auf gehts!*)

- multilingual groups of <cpt> elements for descriptive text
- <cpc> for comments, <cpd> for descriptive text (s[x] is negative)

Parallel Markup in CodeML

```
<semantics>
  <ccdef export="find">... <cpg id="foo">...</cpg>...</ccdef>
  <pcode format="pseudo">...<cpg xref="foo"> ...  </cpg>...</pcode>
  <pcode format="java">...<cpg xref="foo"> ... </cpg>...</pcode>
  <rawcode format="java"> ...</rawcode>
</semantics>
```

- Parallel markup allows to cross-reference (enables format synopsis)
  - find corresponding regions in different implementations
  - associate program code with comments/specification/pseudo-code
  - compare/interchange instructional code with otptimized code fragments

CodeML: Summary

- Presentation Elements: (to control the visual appearance)
  - `cpg`, `cpbr` (grouping and breaking)
  - `cpo`, `cpi`, `cpb`, `cptype` (token elements)
- Content Elements: (to specify the semantics)
  - `ccv`, `cccsym` (variables, constants)
  - `apply`, `bind`, `ccdef` (application, binding, definition)
- Descriptive text/migration: `cpt`, `cpd`, `cpc`
- Parallel Markup: `semantics`, `pcode`, `rawcode`
- Metadata: Extension of Dublin Core
- Caveats: do not produce/read/compile directly
  $10\times$ size blowup before compression

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>
```

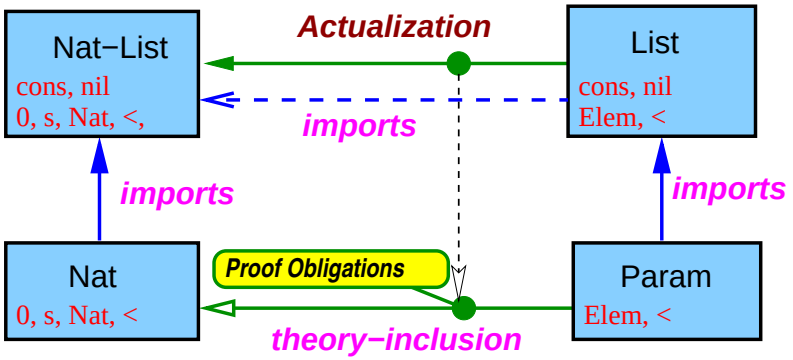
Statement level:

- Defi nition, Theorem, Proof, Example
- semantics explicit forms and refs.

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



Example: an OMDoc Defi nition

```
<definition id="c6s1p4.d1" for="monoid">
  <CMP> A <with role="definiens">monoid</with> is a structure
    (M, *, e) where (M, *)  is a  semi-group and e is a unit for *. </CMP>
  <FMP>∀M, *, e.s_grp(M, *) ∧  unit(e) ⇒  monoid(M, *, e)</FMP>
</definition>
```

- Make use the co-occurrence of formal and informal representations by cross-referencing (OMDoc extends OpenMath)

```
<CMP> A <with role="definiens">monoid</with> is a structure
  <OMOBJ xref=""/> if <OMOBJ  xref=""/>  is a  semi-group
  and <OMOBJ xref=""/> is a  unit. </CMP>
<CMP xml:lang="de"> Eine Struktur <OMOBJ xref=""/> heißt
<with role="definiens">monoid</with> ist falls <OMOBJ xref=""/>
 eine Halbgruppe ist und <OMOBJ xref=""/> eine Einheit. </CMP>
<FMP system="STLC">∀M, *, e.s_grp(M, *) ∧ unit(e) ⇒ monoid(M, *, e)</FMP>
<FMP system="OWL">Mon ⊑  s_grp ⊓ ∃unit.</FMP>
```

Modular Specifi cation of Presentation/System Data

- OMDoc needs presentation style sheets. (raw OMDoc illegible)
- Problem: Documents define new symbols with specialized notations
- Solution: Store presentation information locally (meta-language)

  generate document/collection-specific style sheets

```
<presentation for="binomial" parent="OMA">
 <use format="default" fixity="infix">choose</use>
 <use format="TeX" lbrack="\bigl({" rbrack="}\bigr)">\atop</use>
 <use format="pmml" element="mfrac" attributes="linethickness='0'"/>
</presentation>

<presentation for="power" parent="OMA" fixity="infix"
   crossref-symbol="no" precedence="200" bracket-style="lisp">
 <use format="html" fixity="prefix" element="sup"/>
 <use format="TeX">^</use>
 <use format="pmml" element="msup"/>
</presentation>
```

- Added Benefit: also use for generating output for external systems

Example from the DLMF (Airy Functions)

A.7 Asymptotic Expansions

Definition (zeta-def) A local abbreviation $\zeta = \frac{2}{3} \cdot z^{\frac{3}{2}}$

Definition (Al.AS.Z.u-def)

u_0=1 $\quad$ u_s=$\frac{(2\cdot s+1)\cdot(2\cdot s+3)\cdot(2\cdot s+5)\cdot\cdots\cdot(6\cdot s-1)}{216^s\cdot !(s)}$

Definition (Al.AS.Z.v-def)

v_0=1 $\quad$ v_s=$-(\frac{6\cdot s+1}{6\cdot s-1} \cdot u_s)$

Theorem (eq:Al.AS.A)

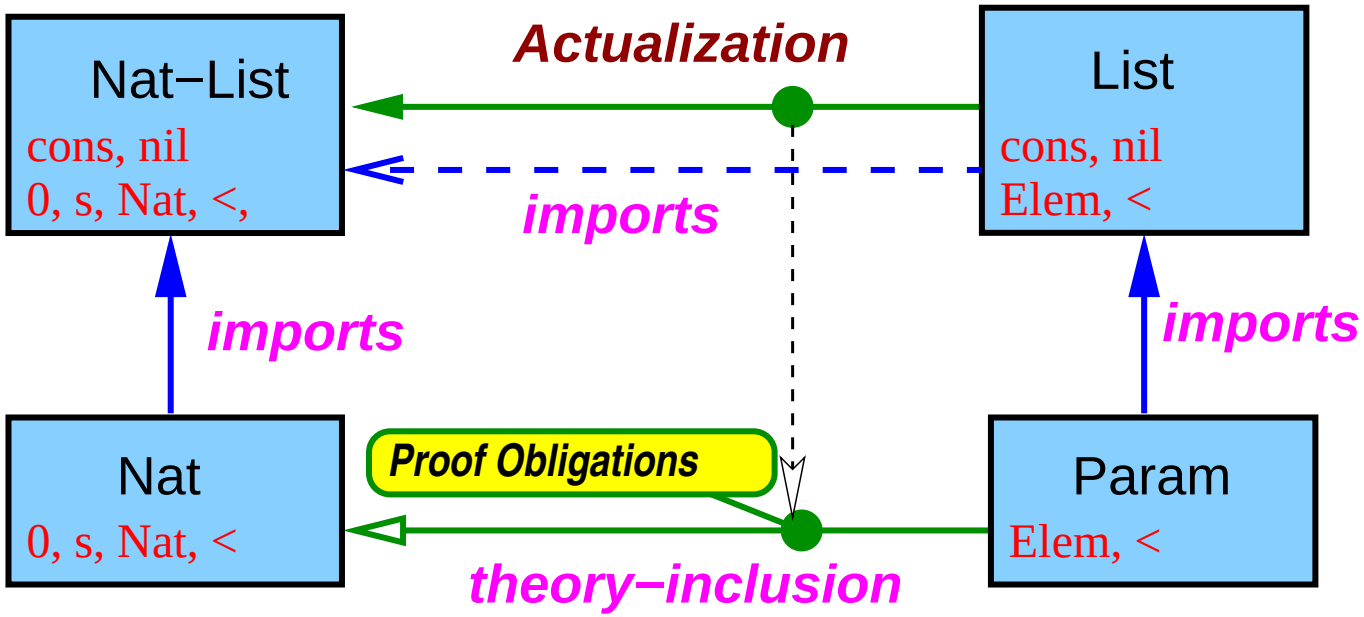
$Ai(z) \sim \frac{e^{-\zeta}}{2\cdot \sqrt[2]{\pi}\cdot z^{\frac{1}{4}}} \cdot \Sigma_{s=1}^{\infty}(-)^s \cdot \frac{u_s}{\zeta^s}$, if $|\mathbf{ph}(z)| < \pi$

Theorem (eq:Al.AS.AP)

$Ai'(z) \sim -\frac{z^{\frac{1}{4}}\cdot e^{-\zeta}}{2\cdot \sqrt[2]{\pi}} \cdot \Sigma_{s=1}^{\infty}(-)^s \cdot \frac{v_s}{\zeta^s}$, if $|\mathbf{ph}(z)| < \pi$

Theory Management in OMDoc

- Theory management follows Dieter Hutter's development graph model
- Definition and Theorem links (global and local)



- Supports a notion of theory reuse and theory change.
- theories and links represented by special OMDoc elements

A Structured Theory: Lists of Natural Numbers

```
<theory id="Param">
  <symbol id="Elem" type="sort"/>
</theory>

<assertion id="testthm"/>
```

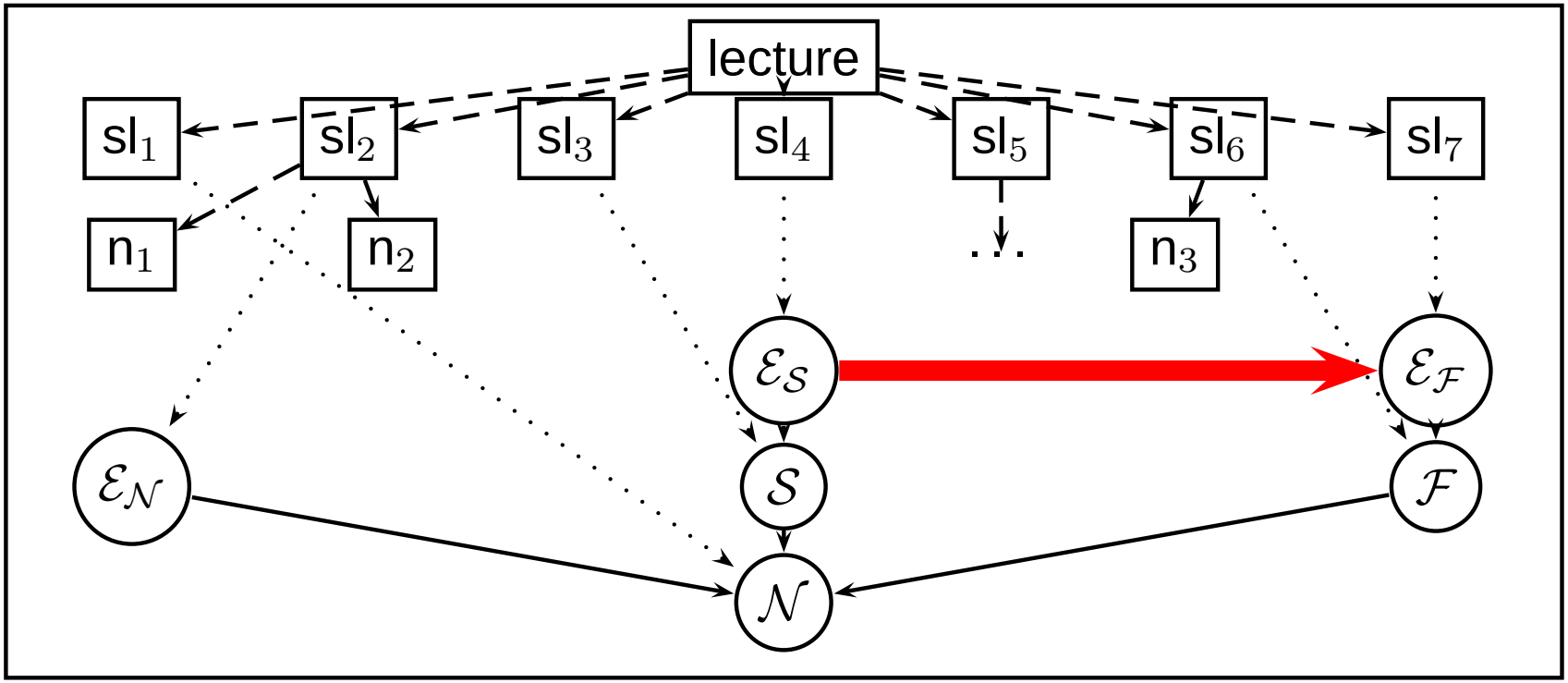
```
<theory id="List">
  <symbol id="List-sort" type="sort"/>
  <symbol id="cons"/>
  <symbol id="nil"/>
</theory>
```

```
<theory id="nat-list.thy">
  <imports id="nat-list.im-nat"     type="global" from="nat-thy"/>
  <imports id="nat-list.im-Element" type="local"  from="List">
    <morphism id="elem-nat"><requation>
        <pattern><OMOBJ><OMS cd="Param" name="Elem"/></OMOBJ></pattern>
        <value><OMOBJ><OMS cd="nat.thy" name="Nat"/></OMOBJ></value>
  </requation></morphism></imports>
  <inclusion item="elem-nat-incl"/>
</theory>

<axiom-inclusion id="elem-nat-incl" from="nat.thy" to="Element" by="testthm">
  <morphism id="elem-nat-incl-morph" base="elem-nat"/>
</axiom-inclusion>
```

Content Structure vs. Narrative Structure

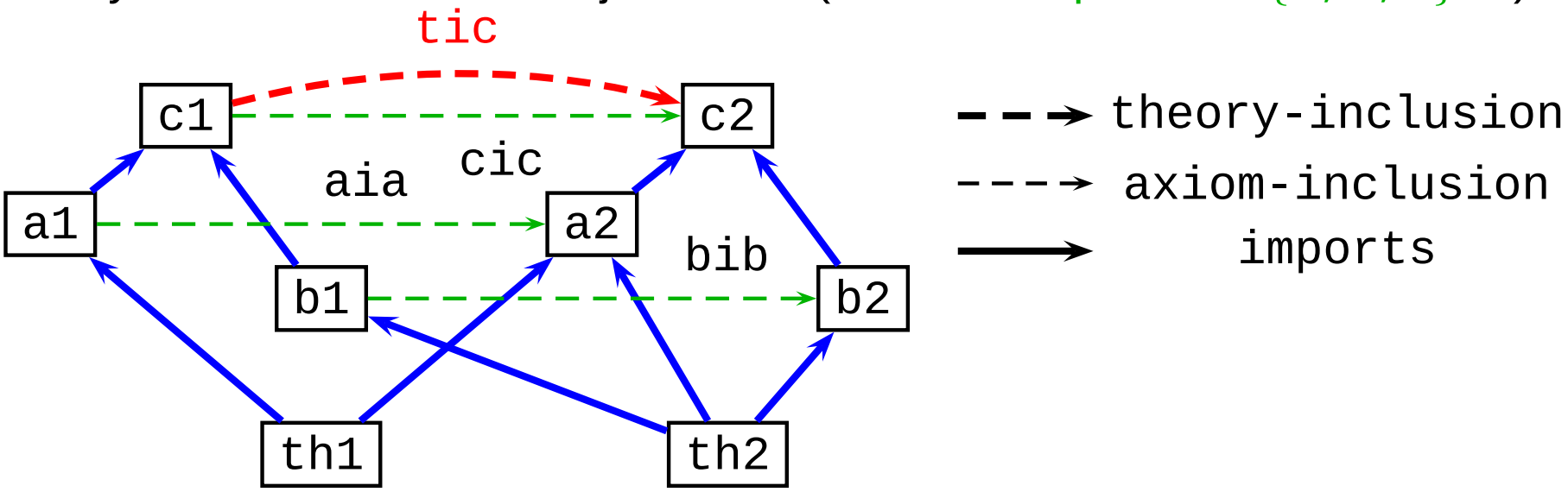
- Document structure is often influenced by, but not identical to semantic structure



- Idea: have two documents content + narrative structure
- Narrative OMDoc: only document structure + narrative elements + links into content OMDoc.

Statements about Theories (Theory Inclusions)

- Theory Inclusion: All axioms in the source th. are theorems in the target
- Application: transport of theorems, change of notation. (e.g. Ricci-Tensor)
  - just generate your theory by supplying a morphism that adapts $\mathcal{R}^{ij}$
- Theory inclusions have to be justified (`tic` decomposes to `{a,b,c}ic`)



Conclusions

- Content markup techniques opens the way for machine support
  - need to extend formal methods (formal methods light $\rightsquigarrow$ content)
  - proven applications in mathematics and CS education
- Building blocks already exist (we stand on the shoulders of giants)
  - MATHML and OPENMATH (for mathematical formulae)
  - OMDoc (for the large-scale structure of scientific knowledge)
- or are under development (need help from real computer scientists)
  - CODEML for structuring program code, CHEMML,...
- It is time to really capture the content of Science & Engineering
- more information: `http://www.mathweb.org/{omdoc|mbase}` `http://www.activemath.org`