Content Markup: Principles and Consequences

Andreas Strotmann
Universität zu Köln, ZAIK/RRZK

OpenMath Workshop -- Bremen 2003

Introduction

Presentation of results from my dissertation "***Content Markup Language Design Principles***"

- Ladislav J. Kohout, major professor
- Mika Seppälä, R. van Engelen, K. Gallivan, H. Levitz, committee members
- official date May 2003, FSU CS
  - www.cs.fsu.edu/research/reports
  - etds.lib.fsu.edu/

Introduction (ctd.)

- Focus on ideas with results that may influence development of OpenMath 2.0
- including those that have influenced development of MathML 2.0, 2nd ed.

- To be continued this afternoon with concrete proposals for OpenMath 2.0

Research Topic

- Understanding Content Markup Language Design
  - *well*… "towards a better understanding of…"
  - because existing language designs have been flawed
  - … due to lack of a deeper understanding
- Approach Based on an Observation
  - Content markup languages are knowledge communication languages for heterogeneous systems
  - there is only one known high-quality solution to the knowledge communication problem: human language

Research Ansatz

- => The Linguistics Approach
  - Linguists (and others) have been studying "engineering solutions" of human language for a long time, with impressive results
  - Proposal: transfer "engineering solutions" to content markup language designs
- But: How do we "prove" this works???
  - Formal proof clearly impossible…
  - *The proof of the pudding is in the eating*

Research Method

- Application of select tools from linguistics to content markup language design
  - Language architecture: layers & components
  - Compositionality Principle
  - Categorial Semantics
- Successful transfer of these non-trivial "corollaries" supports main "conjecture"
  - + Outlook to as yet untried tools adds weight

Table of Contents

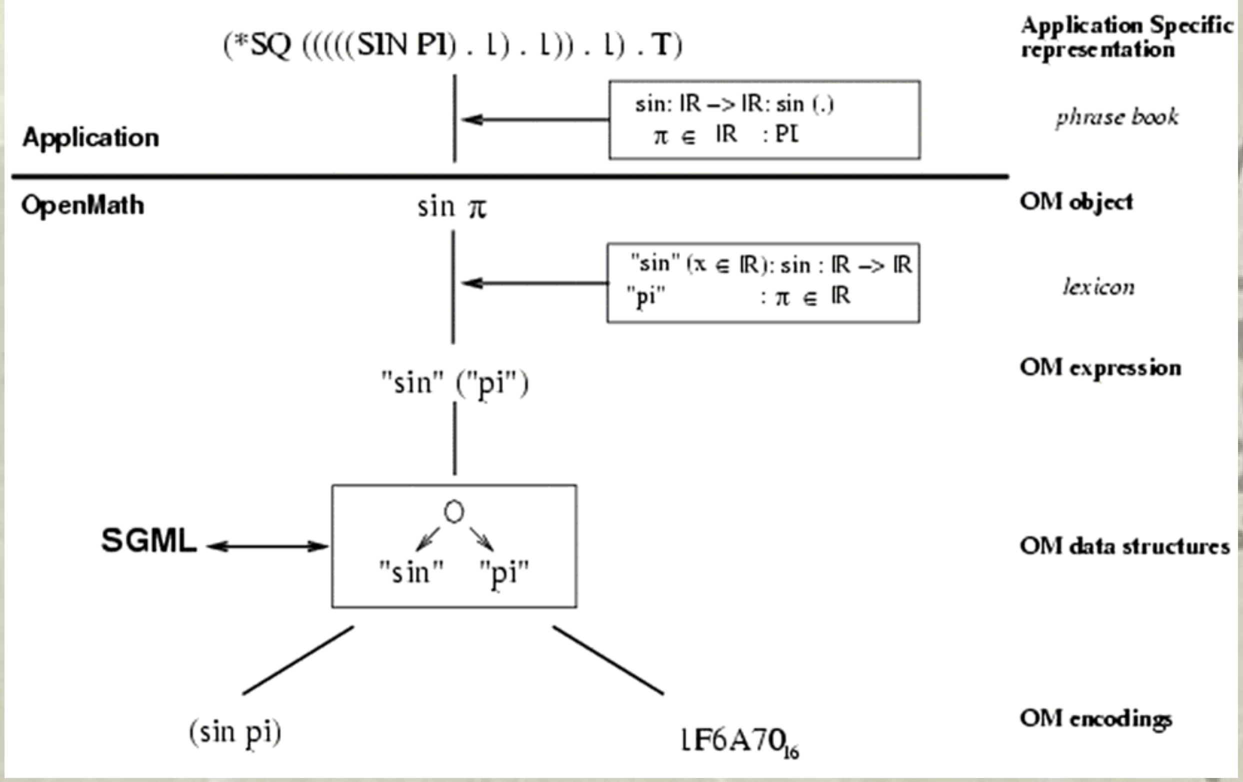
Linguistics Parallel: Motivation

- Human language universals
  - developed under intense evolutionary pressure
  - => provide a "good" engineering solution
    - => use "principles", but ignore "parameters"
  - Human language solves similar problem to content markup
    - communication of meaning between similar but different intelligent agents
    - => study design principles of language in order to design good content markup languages

Linguistics Ansatz: Language Layers

- Linguistics background
  - Language components:
    - Morpho-syntax,
    - Syntax,
    - Structural (“categorial”) semantics,
    - Lexical semantics,
    - Pragmatics,
    - Semiotics

Application: Language Layers for OpenMath



- published in "OpenMath Objectives" *('95/98)*

Linguistics Ansatz: Syntax Layer

- Linguistic background (& proposals)
  - "X-bar" : "typed" tree structure ((head arguments…) modifiers…)
    - cf. OpenMath ((head arguments…) attributions…)
    - head determines "type"
    - modifiers ~ named arguments with defaults
  - "Government and Binding"
    - syntax for scopes (cf. OM Binder, MML <bvar>)
    - syntax for co/cross-references (cf. MML id=, ref=)

Linguistics Ansatz: Syntax- Semantics Interface

- Linguistics Background
  - Compositionality Principle (*aka* Frege-Prinzip)
    - "Meaning of compound expression is function of syntactic composition rule and meanings of parts"
    - Research principle underlying Formal Semantics
    - Many applications in CS
  - Categorial Semantics
    - Categories of lexical items with identical behavior
    - Meaning category from category of parts & syntax
    - Categorial type logics/ type systems

Linguistics Ansatz: Pragmatics

- Linguistic Background
  - syntactic categories correspond roughly to language layers
    - NPs - who, what… : static semantics
    - VPs - doings: dynamic semantics (?)
    - IPs - judgments: pragmatics?
  - Words can systematically shift levels
  - categories can all be nested inside each other
- Content markup currently ~ Noun Phrase
  - KQML, OMdoc ~> VP (actions, actors)
    - mutual inclusion property still missing

Compositionality

- Compositionality
  - in CS usually understood to constrain semantics given syntax
  - in philosophy of language, "systematicity" only possible given compositional world view
  - in linguistics, compositional semantics and syntax constrain each other
  - here: allowable syntactic structure constrained by intended semantic structure

Compositionality and Language Design

- "Meaning of compound is function of meaning of parts and syntactic construct"
  - often: "exists homomorphism from syntactic algebra to semantic algebra"
  - hence: distinct semantic constructors require distinct syntactic constructors
  - usual ingredients: numbers, variables, names; application
  - also needed: variable binding, typing syntax

Compositionality and language analysis

- Compositionality analysis of an example
  - determine semantic decomposition(s)
  - determine distinct semantic constructors
    - many, but not limitless possibilities
- Analysis of existing languages
  - determine if systematic representations of semantic constructors exist
- Design of new language
  - construct homomorphism

Compositionality and Variable Binding Syntax

- Variable binding has special semantics
  - cannot be reduced to combination of other regular language ingredients and application
  - => need special syntax
- Do knowledge communication languages have systematic special binder syntax?
  - Yes: OpenMath, MathML
  - No: KIF, CA user languages
    - But KIF 3 defines lambda as special syntax

Compositionality and Higher-Order Operators

- Special syntax for variable scoping
  - => “parts” must include body and bound vars
  - (more parts are possible, e.g. binder)
- => Do specific language ingredients that represent variable binding require these as necessary parts?
  - Counter examples: KIF “setof”, MathML “min”
- When not: construct examples with errors

Compositionality: Practical Consequences

- Found errors in KIF 3.0, dpANS KIF, MathML
- OpenMath Binding Objects explicitly added to improve compositionality (S. Watt)
- Languages with explicit typing require special type assignment syntax (missing in all content languages we have looked at)

Categorial Types

- analysis tool for content markup languages
  - has been applied to mathematical formulas from 1935!
- ~ type-level generalization of λ calculus called Lambek calculus
  - application, abstraction, reduction rules…
  - types of atoms “ignored”/ factored out
  - unification of concrete types left as an SEP
  - interaction between categorial (structural) and concrete (lexical) type system generally benign
    - Dörre, Manandhar: On constraint-based Lambek calculi, 97

Categorial Types: Application to OpenMath

- Categorial types for OpenMath
  - proposal of full categorial type system
    - compatible with existing systems
    - compositional categorial type assignment function for all OM Object constructors (application, binding, attribution, error)
  - flushed out and fixed severe OM spec error
    - current syntax of OpenMath cannot allow intended Currying semantics
    - problem traced to extra “part” of Binding object

Categorial Types: Application to MathML (to do)

- MathML 2.0, 2nd ed, finally ready for categorial type system (10/2003)
  - Special syntax for variable binding, domain-of-application...
  - Applies systematically to any operator, not just a few
  - Systematic correspondence between „functional“ and „binder“ usage patterns conforms to our categorial type system view

Principles

- Compositionality Principle
- Radical Lexicalism
- “Categorial” Semantics

- Linguistics Parallel

The Compositionality Principle

Consequences

Consequences… from Compositionality Principle

- Every class of qualitatively different semantic constructs requires its own special syntactic construct
  - atomic: variable, name, strings, numbers…
  - structural: application (positional and **named arguments**), binding, **typing** information
  - pragmatic: command, question…

Systematicity

- If a class of concepts is open-ended, it should be handled systematically
  - MathML 1 --> 2: make binding and domains of application available beyond closed set of ops
  - MathML 2, 2nd ed.: make them available systematically, including equivalence of functional and binder formulation of generalized quantifiers
  - OM2 draft: binder symbols are regular ops, too!

Lexicalism

- Clean factorization of semantics into
  - structural (a.k.a. “categorial”) semantics
  - and lexical semantics (“ontologies”)
- meaning(s) of a word is lexical entry
  - complex types/semantics as context specs
- semantic interpretation or type inference rules exclusively in structural terms
  - lexicon does not allow adding rules

Lexicalism

- Lexical type theory “orthogonal” to structural (“categorial”) type theory
  - Result from formal semantics (linguistics) lit: For a large class of categorial type theories (L2 and lower) and a large class of lexical type theories (lattice), their combination is very well behaved (e.g. decidability depends on lexical “plugin” type theory, *not* on categorial “framework” type theory)

Categorial Types

- L2 encompasses
  - application and abstraction types
  - unification over type variables
  - currying (and much more)
  - no quantification over type variables (!!!)
  - no explicit typing (lexical typing only)
  - no(?) domain-of-application
- result applies to simpler theories, too

Proposals

Cleanup

- Make sure standard encodings can
  - encode all OM objects
    - remove arbitrary size limits in binary encoding
  - do round-trip encoding
- compatibility with MathML v 2 ed 2
  - equivalence of uses of binder symbols in application or binding objects
  - domain-of-application

Standardize Formal Structural Type System

- Extend STS to become full-fledged standard "categorial" type system for OpenMath
  - we can (but don't have to) define currying properly here!
  - Compatible with proposal for semantic attrs.
  - Equivalence of functional and binder uses
    - compatible with MathML 2, 2nd ed.
    - incompatible with some OM2 proposals!
- Possible, but not necessarily trivial

A Consequence for Types

- Every type theory for OpenMath must extend the Standard Structural Type System
- Common fundamental type constructors
  - abstraction (mapping, n-ary mapping)
  - application (reduction)
  - natural numbers, reals, complex numbers…
  - type descriptor (e.g.)
  - CD or required entries in type theory def., e.g.

“Categoriality” property

- Prove that extended STS works properly with embedded type systems
  - along the lines of existing proof of compatibility of L2 type logic with type lattices
  - potential problems:
    - explicit type assignments
    - n-ary operators

MathML compatibility

- Common type theory may serve as basis for formal proof of compatibility between OpenMath 2.0 and MathML 2.0, 2nd ed.
  - Requires research!

OpenMath Layers

- Consider re-introducing an extra OpenMath layer as originally proposed in “Objectives”
  - intermediate layer defined as structural (“categorial”) semantics layer

Reconsider Binding Objects

- It turns out that the binder argument to a binding object complicates its semantics considerably
  - e.g. makes it impossible to define a currying rule without introducing a categorial theory
- Consider pros and cons of replacing binding objects by lambda objects.
  - These lend themselves naturally to currying