Aspects of Mathematical Knowledge

—

The Tetrapod Model

Michael Kohlhase

Computer Science, FAU Erlangen-Nürnberg

Sept. 4. 2023, Tetrapod Workshop @ CICM'23

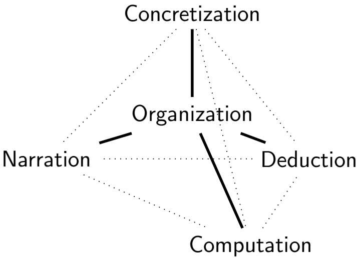
1 A Holistic Model for “Doing Math”; see [Car+21]

Knowledge Representation is only Part of “Doing Math”

- **Definition 1.1.** One of the key insights is that the mathematics ecosystem involves a body of knowledge externalized in an ontology that provides organization and combines the following four aspects:
  - Deduction: exploring theories, formulating conjectures, and constructing proofs
  - Computation: simplifying mathematical objects, re contextualizing conjectures. . .
  - Concretization: collecting concrete examples/models, applying mathematical knowledge to real-world problems and situations.
  - Narration: devising both informal and formal languages for expressing mathematical ideas, visualizing mathematical data, presenting mathematical developments, organizing and interconnecting mathematical knowledge

"Doing Math": as a Tetrapod

- We call the endeavour of creating a computer-supported mathematical ecosystem "Project tetrapod" as it needs to stand on four legs.



- **Collaborators:** KWARC@FAU, McMaster University

Tetrapod Example: The Fibonacci Numbers

- We present the five aspects of math using the Fibonacci numbers

Tetrapod Example: The Fibonacci Numbers

- We present the five aspects of math using the Fibonacci numbers
- **Organization:** We give the syntax/notation, possibly types, and possibly a definition/specification (so we remember)

```
theory natarith =
  nat : type
  plus : nat −> nat −> nat
  ...
```

```
theory fibonacci =
  include ?natarith
  fib : nat −> nat | # F (1)
  fib_base = ⊢ fib 0 = 1 ∧ fib 1 = 1
  fib_step = ⊢ fib (n+2) = fib (n+1) + fib n
```

Tetrapod Example: The Fibonacci Numbers

- We present the five aspects of math using the Fibonacci numbers
- **Organization:** We give the syntax/notation, possibly types, and possibly a definition/specification (so we remember)
- **Concretization:** We enumerate the values in a concrete data type (a list of integers)

$$fib = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, \ldots]$$

Tetrapod Example: The Fibonacci Numbers

- We present the five aspects of math using the Fibonacci numbers
- **Organization:** We give the syntax/notation, possibly types, and possibly a definition/specification (so we remember)
- **Concretization:** We enumerate the values in a concrete data type (a list of integers)
- **Narration:** We talk/write about the Fibonacci numbers, e.g. in English. (addressed to humans)

  > *Fibonacci numbers are named after Italian mathematician Leonardo of Pisa, later known as Fibonacci. In his 1202 book Liber Abaci, Fibonacci introduced the sequence to Western European mathematics, although the sequence had been described earlier in Indian mathematics, as early as 200 BC in work by Pingala on enumerating possible patterns of Sanskrit poetry formed from syllables of two lengths.* [Wikipedia]

Tetrapod Example: The Fibonacci Numbers

- We present the five aspects of math using the Fibonacci numbers
- **Organization:** We give the syntax/notation, possibly types, and possibly a definition/specification (so we remember)
- **Concretization:** We enumerate the values in a concrete data type (a list of integers)
- **Narration:** We talk/write about the Fibonacci numbers, e.g. in English. (addressed to humans)
- **Computation:** E.g. a python implementation

```
def fib():
    """ Generates the Fibonacci numbers, starting with 0 """
    x, y = 0, 1
    while 1:
        yield x
        x, y = y, x+y
```

Tetrapod Example: The Fibonacci Numbers

- We present the five aspects of math using the Fibonacci numbers
- **Organization:** We give the syntax/notation, possibly types, and possibly a definition/specification (so we remember)
- **Concretization:** We enumerate the values in a concrete data type (a list of integers)
- **Narration:** We talk/write about the Fibonacci numbers, e.g. in English. (addressed to humans)
- **Computation:** E.g. a python implementation
- **Deduction:** We give theorems and proofs about the Fibonacci numbers

$$\forall n \in \mathbb{N}. F_n = \sum_{k=0}^{\left\lfloor \frac{n-1}{2} \right\rfloor} \binom{n-k-1}{k}.$$

Tetrapod Example: The Fibonacci Numbers

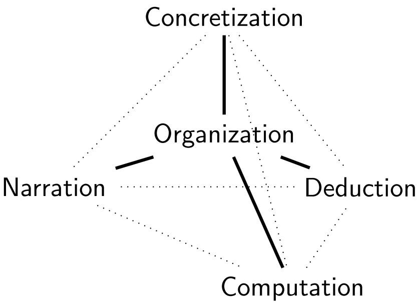
- We present the five aspects of math using the Fibonacci numbers
- **Organization:** We give the syntax/notation, possibly types, and possibly a definition/specification (so we remember)
- **Concretization:** We enumerate the values in a concrete data type (a list of integers)
- **Narration:** We talk/write about the Fibonacci numbers, e.g. in English. (addressed to humans)
- **Computation:** E.g. a python implementation
- **Deduction:** We give theorems and proofs about the Fibonacci numbers
- **Observation 1.8.**
  - *In traditional Maths, we effortlessly combine all of these.*
  - *In computational Maths, we use different formats and workflows.*
- **But:** There is great potential for the aspect-specific tools to interact synergistically! (Challenge for CICM)

Examples of Languages/Systems by Aspect

- **Example 1.9.**

Aspect	KR Langs (examples)	KPTs (examples)
Organization	ontology languages (OWL), description logics (ALC)	reasoners, SPARQL engines (Virtuoso)
Concretization	relational databases (SQL, JSON)	databases (MySQL, MongoDb)
Computation	programming languages (C)	interpreters, compilers (gcc)
Deduction	logics (HOL)	theorem provers (Isabelle)
Narration	document languages (HTML, LaTeX)	editors, viewers

Reminder: The TetraPod



2 The OEIS: Online Encyclopedia of Integer Sequences

OEIS: Open Encyclopedia of Integer Sequences

- **Definition 2.1.** An intger sequence is a function $s\colon \mathbb{N}\to\mathbb{Z}$.
- **Applications:** Every parametric phenomenon that can be counted.
- **Example 2.2.** A000944: Number of polyhedra (or 3-connected simple planar graphs) with n nodes (0, 0, 0, 1, 2, 7, 34, 257, 2606,...)
- **Example 2.3.** A001222: Number of prime divisors of n counted with multiplicity (0, 1, 1, 2, 1, 2, 1, 3, 2, 2, 1, 3, 1, 2, 2,...)
- **Example 2.4.** A031214: First elements in all OEIS sequences (in order) (1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...)
- **Intuition:** If phenomena grow with the same sequence $\rightsquigarrow$ related?
- **Idea:** Collect many integer sequences (Neil Sloane 1965 $\rightsquigarrow$ OEIS)
  - started as a book: *A Handbook of Integer Sequences* 1973 (2372 sequences)
  - online since 1994 (16.000 sequences $\rightsquigarrow$ `http://oeis.org`)
  - OEIS Foundation: 2009 (Creative Commons License)
  - today: $\geq$ 350 000 sequences

OEIS Data Representation

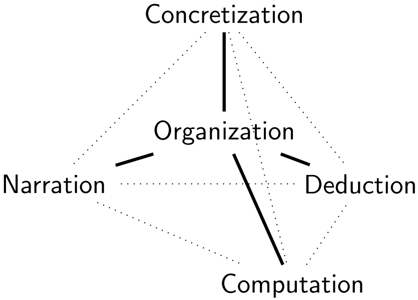
- One "record" per sequence with fields including
  - Identifier: A??????
  - start values (DB Key)
  - name (maybe with short explanation)
  - author
  - references to papers
  - program code (Mathematica, Pari, . . .)
  - Formulae

  All in ASCII files keyed by one-letter line prefixes.

- **Example 2.5 (Fibonacci Numbers).**

```
%I A000045 M0692 N0256
%S A000045 0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987
%N A000045 Fibonacci numbers: F(n) = F(n−1) + F(n−2) with F(0) = 0 and F(1) = 1.
%D A000045 V. E. Hoggatt, Jr., Fibonacci and Lucas Numbers. Houghton, Boston, MA, 1969.
%F A000045 F(n) = ((1+sqrt(5))^n−(1−sqrt(5))^n)/(2^n∗sqrt(5))
%F A000045 G.f.: Sum_{n>=0} x^n ∗ Product_{k=1..n} (k + x)/(1 + k∗x). — _Paul D. Har
%F A000045 This is a divisibility sequence; that is, if n divides m, then a(n) divides a(m)
%A A000045 _N. J. A. Sloane_, Apr 30 1991
```

Reminder: The TetraPod



3 Charaterizing the Tetrapod

The Tetrapod Aspects interact synergistically

- Organization gives a uniform vocabulary and structure to the other four, e.g.
  - for disambiguating annotations in narration
  - as column legends in concretization schemata
  - for system interoperability between computational systems.
  - to collect all knowledge (theorems) about a concept (deduction)

  The other four aspects give additional “semantics” to organization.

The Tetrapod Aspects interact synergistically

- Organization gives a uniform vocabulary and structure to the other four, e.g.
  - for disambiguating annotations in narration
  - as column legends in concretization schemata
  - for system interoperability between computational systems.
  - to collect all knowledge (theorems) about a concept (deduction)

  The other four aspects give additional "semantics" to organization.
- Narration occurs naturally as documentation of the other four, e.g.
  - as comments in programs (computation),
  - as introductions to theorems (deduction),
  - as row legends in concretization schemata,
  - as glossary explanations in organization.

The Tetrapod Aspects interact synergistically

- Computation can compute
  - the values in concretization

The Tetrapod Aspects interact synergistically

- Computation can compute
  - the values in concretization
- Concretization can provide
  - examples and counter-examples for deduction,
  - unit tests benchmkarks for computation.
  - data for machine learning methogs.

The Tetrapod Aspects interact synergistically

- Computation can compute
  - the values in concretization
- Concretization can provide
  - examples and counter-examples for deduction,
  - unit tests benchmkarks for computation.
  - data for machine learning methogs.
- Deduction can justify
  - shortcuts in computations
  - completeness of concretizations.

The Tetrapod Aspects interact synergistically

- Computation can compute
  - the values in concretization
- Concretization can provide
  - examples and counter-examples for deduction,
  - unit tests benchmkarks for computation.
  - data for machine learning methogs.
- Deduction can justify
  - shortcuts in computations
  - completeness of concretizations.
- **But:** to reap these synergies we need
  - a combined – or at least linked – representation format
  - multi/cross-aspect tools.

Complementary Advantages of the Aspects

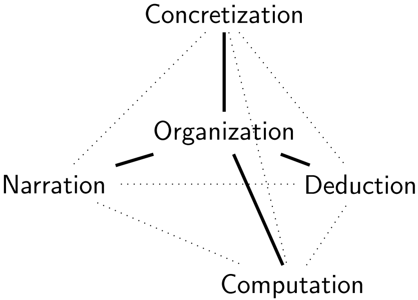
- Characterizing the Aspects: (corners and sides of the tetrahedron)

Aspect	objects	characteristic		
		advantage	joint advantage of the other aspects	application
ded.	formal proofs	correctness	ease of use	verification
comp.	programs	efficiency	well-definedness	execution
concr.	concrete objects	queriability	abstraction	storage/retrieval
narr.	texts	flexibility	formal semantics	human understanding

- Characterizing the two-aspect systems (edges of the tetrahedron)

Aspect pair	characteristic advantage
ded./comp.	rich meta-theory
narr./conc.	simple languages
ded./narr.	theorems and proofs
comp./conc.	normalization
ded./conc.	decidable well-definedness
comp./narr.	Turing completeness

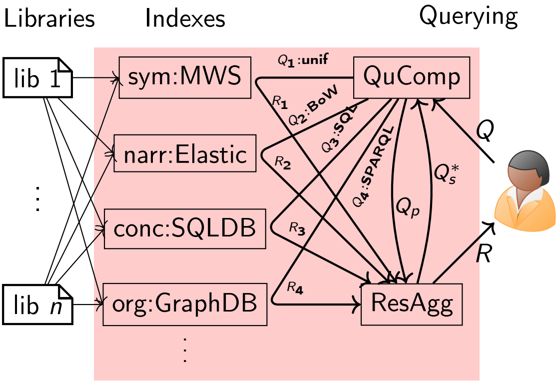
Reminder: The TetraPod



4 Tetrapod some food for thought?

Towards Tetrapodal Search

- An Architecture for Tetrapodal Search



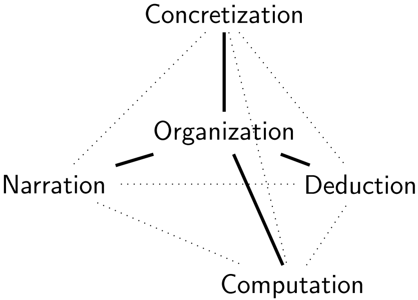
Levels of Semantics – e.g. in Math/CS

- **Documentation**: answers given as text
  - pro: easy to read for humans, critical to build intuitions
  - con: often ambiguous, contradictory, or incomplete
- **Specification**: correct answers defined by rule system (also called calculus or inference system)
  - pro: good stepping stone between the other two levels
  - con: accomplishes the pros of neither of them
- **Implementation**: answers computed by algorithm
  - pro: easy to automate, critical for efficiency and scale
  - con: essentially impossible to understand or analyze
- **Unit testing**: set of query/answer pairs
  - pro: easy to write, automate
  - con: does not cover the whole semantics

Tetrapodal Aspects of Semantics – e.g. in Math/CS

- **Documentation**: answers given as text
  $\widehat{=}$ The narration aspect of semantics
  - pro: easy to read for humans, critical to build intuitions
  - con: often ambiguous, contradictory, or incomplete
- **Specification**: correct answers defined by rule system (also called calculus or inference system)
  $\widehat{=}$ The deduction aspect of semantics
  - pro: good stepping stone between the other two levels
  - con: accomplishes the pros of neither of them
- **Implementation**: answers computed by algorithm
  $\widehat{=}$ The aspect of semantics
  - pro: easy to automate, critical for efficiency and scale
  - con: essentially impossible to understand or analyze
- **Unit testing**: set of query/answer pairs
  $\widehat{=}$ The concretization aspect of semantics
  - pro: easy to write, automate
  - con: does not cover the whole semantics

Reminder: The TetraPod



References I

[Car+21] Jacques Carette et al. "Big Math and the One-Brain Barrier – The Tetrapod Model of Mathematical Knowledge". In: *Mathematical Intelligencer* 43.1 (2021), pp. 78–87. DOI: 10.1007/s00283-020-10006-0.