Leibniz: A Digital Scientific Notation

Konrad HINSEN

Centre de Biophysique Moléculaire, Orléans, France
and
Synchrotron SOLEIL, Saint Aubin, France

24 November 2016

Computational science today

Z. Merali, Nature **467**, 775 (2010)

SCIENTIFIC PUBLISHING

A Scientist's Nightmare: Software Problem Leads to Five Retractions

Until recently, Geoffrey Chang's career was on a trajectory most young scientists only dream about. In 1999, at the age of 28, the protein crystallographer landed a faculty position at the prestigious Scripps Research Institute in San Diego, California. The next year, in a ceremony at the White House, Chang received a 2001 *Science* paper, which described the structure of a protein called MsbA, isolated from the bacterium *Escherichia coli*. MsbA belongs to a huge and ancient family of molecules that use energy from adenosine triphosphate to transport molecules across cell membranes. These so-called ABC transporters perform many

G. Miller, Science **314** 1856 (2007)

Unexpected behavior

OPEN ACCESS Freely available online

PLoS one

The Effects of FreeSurfer Version, Workstation Type, and Macintosh Operating System Version on Anatomical Volume and Cortical Thickness Measurements

Ed H. B. M. Gronenschild[1,2]*, Petra Habets[1,2], Heidi I. L. Jacobs[1,2,3], Ron Mengelers[1,2], Nico Rozendaal[1,2], Jim van Os[1,2,4], Machteld Marcelis[1,2]

E. Gronenschild *et al.*, PLoS ONE **7**(6) e38234 (2012)

Misuse of black-box software

PNAS

Cluster failure: Why fMRI inferences for spatial extent have inflated false-positive rates

Anders Eklund[a,b,c,1], Thomas E. Nichols[d,e], and Hans Knutsson[a,c]

[a]Division of Medical Informatics, Department of Biomedical Engineering, Linköping University, S-581 85 Linköping, Sweden; [b]Division of Statistics and Machine Learning, Department of Computer and Information Science, Linköping University, S-581 83 Linköping, Sweden; [c]Center for Medical Image Science and Visualization, Linköping University, S-581 83 Linköping, Sweden; [d]Department of Statistics, University of Warwick, Coventry CV4 7AL, United Kingdom; and [e]WMG, University of Warwick, Coventry CV4 7AL, United Kingdom

Edited by Emery N. Brown, Massachusetts General Hospital, Boston, MA, and approved May 17, 2016 (received for review February 12, 2016)

A. Eklund, T.E. Nichols, H. Knutsson, PNAS **113**(28) 7900 (2016)

Why can’t we fix this?

- Insufficient software engineering
- Testing numerical software is very difficult.
- Most scientific software has no specification.
- Most scientific software is badly documented.

Testing floating-point code

- Reliable tests check for equality...
- ... but exact equality never happens with floats.
- ... but exact equality never happens with floats. Why ???
- ... therefore we must introduce tolerances ...
- ... but nobody knows how to choose them.

Full discussion:
K. Hinsen, The Approximation Tower in Computational Science: Why Testing Scientific Software Is Difficult, Comp. Sci. Eng. **17**(4) 72-77 (2015)

Specifications and documentation

Informal specifications / documentation

- Theory: should be published in journal articles etc.
- Practice: complexity of models and methods prevents a full description

Formal specifications

- Unknown to most computational scientists
- No domain-specific languages and tools

Climate models

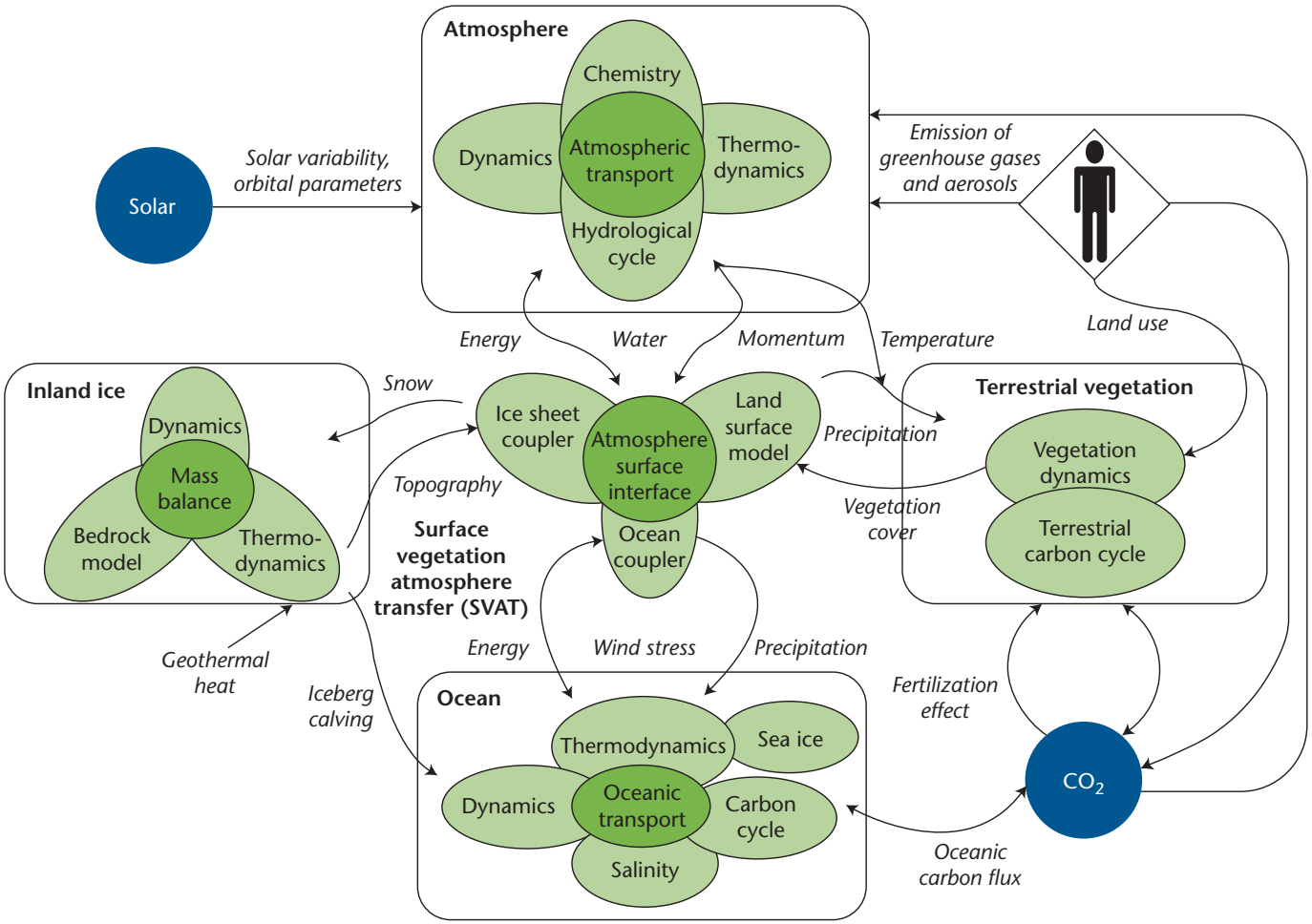
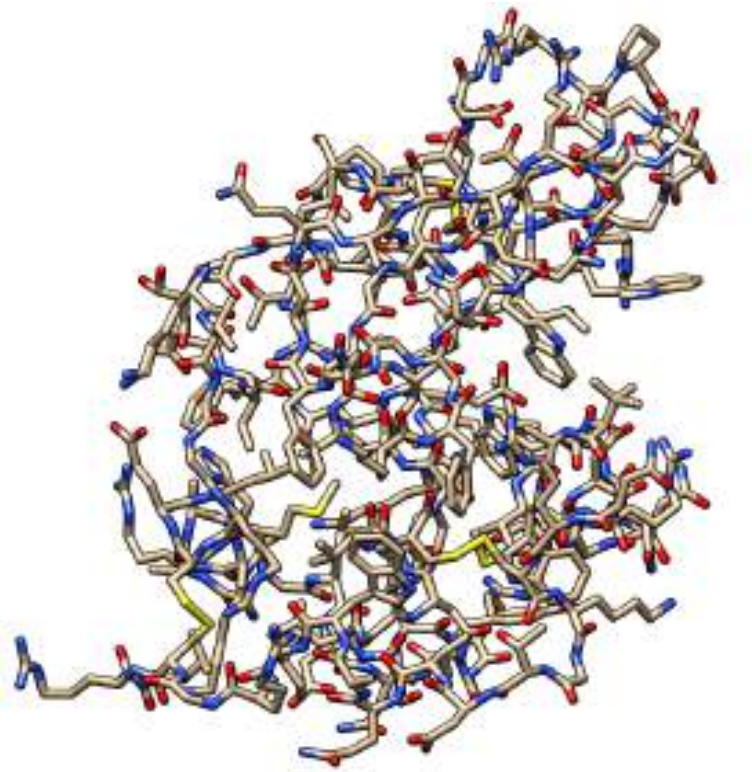


Figure 1. Conceptual view of the components and couplings of a coupled Earth system model.

From: Easterbrook & Johns, Comp. Sci. Eng. **11**(6), 65-74 (2009)

Protein models

A small protein (lysozyme) in the Amber99 force field



- 1960 atoms (1001 shown)
- 183 distinct atom types
- interaction energy: a function of 5880 variables, with a few thousand numerical parameters

Remember that this is a **small** protein!

Biomolecular force fields (1/2)

This is how a typical research paper describes the AMBER force field:

$$
\begin{aligned}
U = & \sum_{\text{bonds } ij} k_{ij}\left(r_{ij} - r_{ij}^{(0)}\right)^2 \\
& + \sum_{\text{angles } ijk} k_{ijk}\left(\phi_{ijk} - \phi_{ijk}^{(0)}\right)^2 \\
& + \sum_{\text{dihedrals } ijkl} k_{ijkl}\cos\left(n_{ijkl}\theta_{ijkl} - \delta_{ijkl}\right) \\
& + \sum_{\text{all pairs } ij} 4\epsilon_{ij}\left(\frac{\sigma_{ij}^{12}}{r^{12}} - \frac{\sigma_{ij}^{6}}{r^{6}}\right) \\
& + \sum_{\text{all pairs } ij} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \\
& + \sum \left(\quad^{(0)}\right)^2
\end{aligned}
$$

Biomolecular force fields (2/2)

Protein parameter files for Amber 12, in somewhat documented formats:

lines	filename
984	amino12.in
814	aminoct12.in
782	aminont12.in
533	frcmod.ff12SB
744	parm99.dat

For the details... read the source code!

For the algorithms that select the parameters for a given protein structure... read the source code!

Read the source code!

Streamlining Development of a Multimillion-Line Computational Chemistry Code

Robin M. Betz and Ross C. Walker | San Diego Supercomputer Center

Software engineering methodologies can be helpful in computational science and engineering projects. Here, a continuous integration software engineering strategy is applied to a multimillion-line molecular dynamics code; the implementation both streamlines the development and release process and unifies a team of widely distributed, academic developers.

Betz & Walker, Comp. Sci. Eng. **16**(3), 10-17 (2014)

Can we do science with computers?

Science requires **verifiability** of all findings, but:

- We cannot discuss computational models and methods
  ... because we cannot write them down.
- We cannot compare different models and methods
  ... for the same reason.
- We cannot test scientific software
  ... because we have no specifications.
- Users don’t understand their software
  ... because it’s too complex ...
  ... and there is no specification.

Similar issues elsewhere (1/2)

Reports and Articles

Social Processes and Proofs of Theorems and Programs

Richard A. De Millo
Georgia Institute of Technology

Richard J. Lipton and Alan J. Perlis
Yale University

Communications of the ACM, 1979

Similar issues elsewhere (2/2)

In Computers We Trust?

As math grows ever more complex, will computers reign?

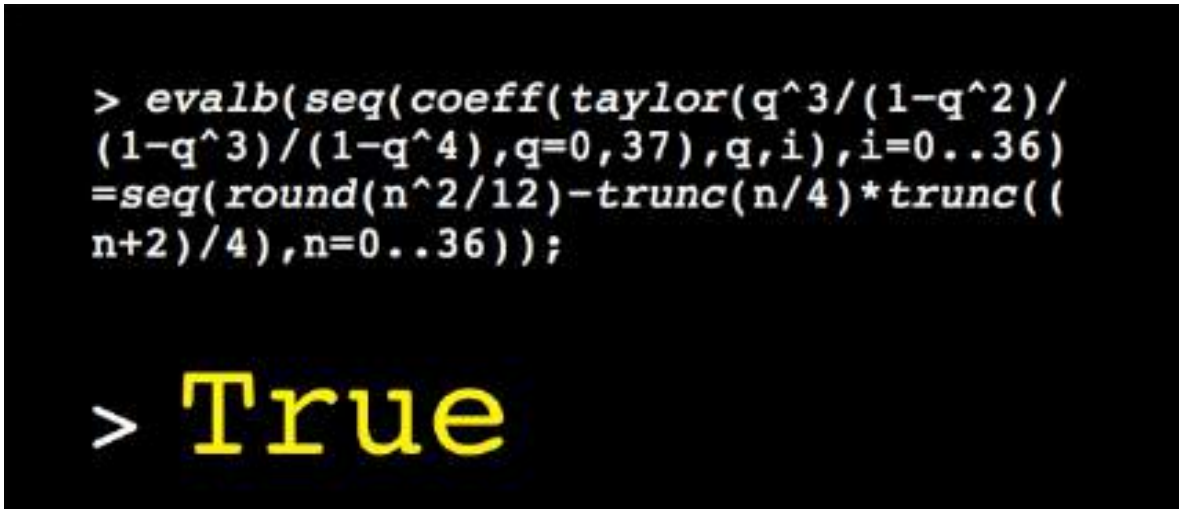


Illustration by Simons Science News

This simple computation, written with math software called Maple, verifies a formula for the number of integer triangles with a given perimeter.

By Natalie Wolchover

Quanta Magazine, 2013

Digital Scientific Notations

Informal language

Digital Scientific Notation

formal ⟷

⟷ embedded into scholarly discourse

automated (partial) validation ⟷

⟷ verifiable by human readers

programming language

Full story

Essay and Opinion

Scientific notations for the digital era

Konrad Hinsen[1,2]

Authors' affiliations

1. Centre de biophysique moléculaire (CBM), CNRS : UPR4301
2. Synchrotron SOLEIL.

K. Hinsen, The Self-Journal of Science, 2016

Commented summary

thesis

Digital science

Technology is transforming science. We can calculate faster than ever before, and gather more precise data, of an ever-expanding variety. We now store staggering volumes of information, and draw on it ever more creatively. Communication between scientists is changing too, even if the scientific paper still remains the standard element of information transfer and recording. The web, e-mail and blogs, as well as social media including Facebook and

Computer software of innumerable kinds has established itself implicitly within the fabric of scientific practice.

apparatus was set up and used; we accept theoretical arguments only if each step is

reduced to numerics. The result is a series of diabolical tendencies all too familiar to computational scientists. As he notes, scientists can develop software using the best practices of software engineering and the result may still compute something different from what its users think it computes. Of course, concerned users can in principle consult the documentation, but how do they know that documentation is complete and accurate?

M. Buchanan, Nature Physics **12** 630 (2016)

Leibniz - a digital notation for the physical sciences

- My first attempt at a Digital Scientific Notation
- Still in a very early stage
- Any feedback welcome:
  - Open Science project on Digital Scientific Notations in general
  - Leibniz on GitHub

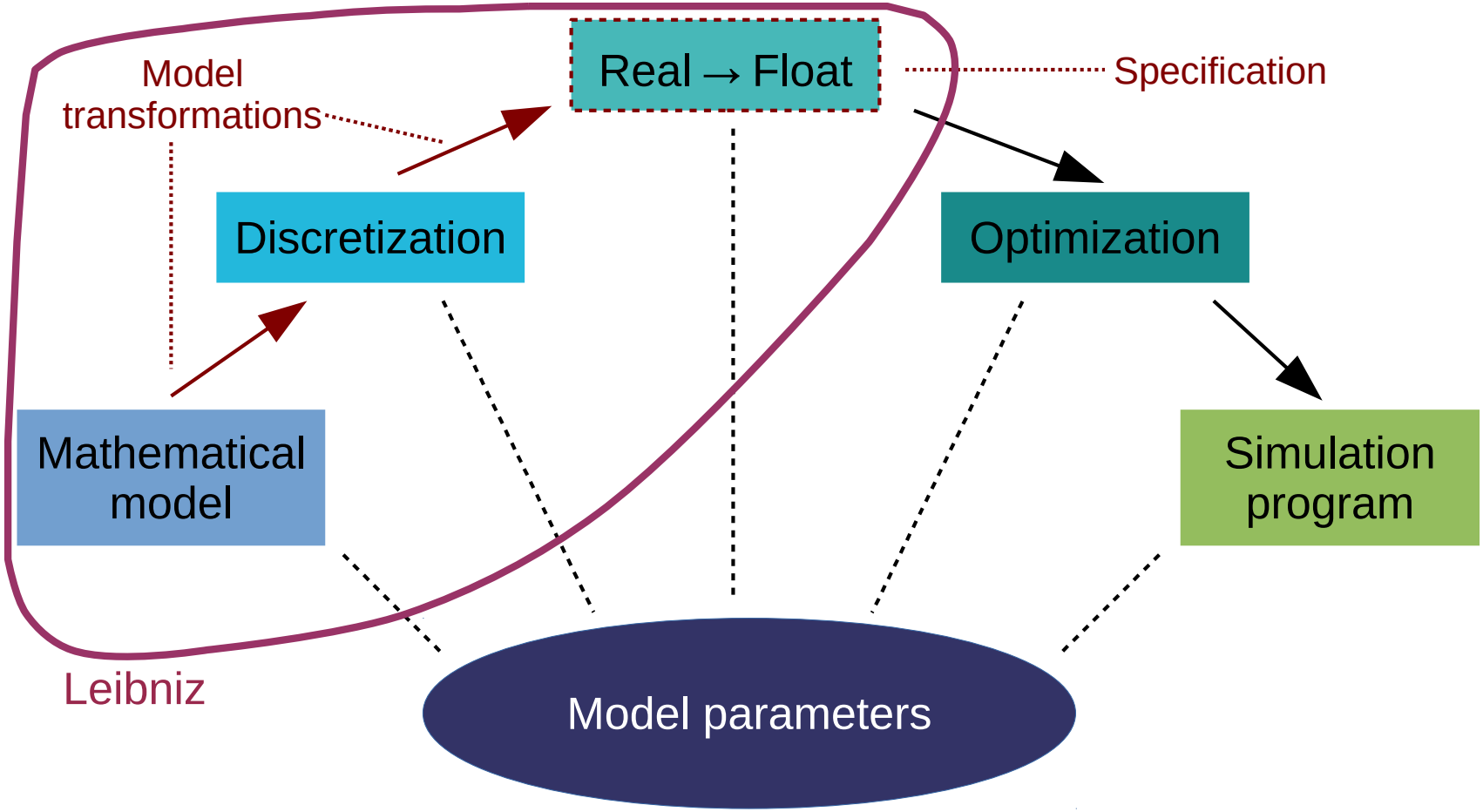
Gottfried Wilhelm Leibniz (1646-1716)

Leibniz attributed all his discoveries in mathematics to the development and use of good notations.

Ideas most relevant to Digital Scientific Notations:

- Leibniz's notation in calculus: dy/dx, $\int$ etc.
- *Calculus ratiocinator*: computing, formal logic, etc.

Approximation chain in numerical simulations

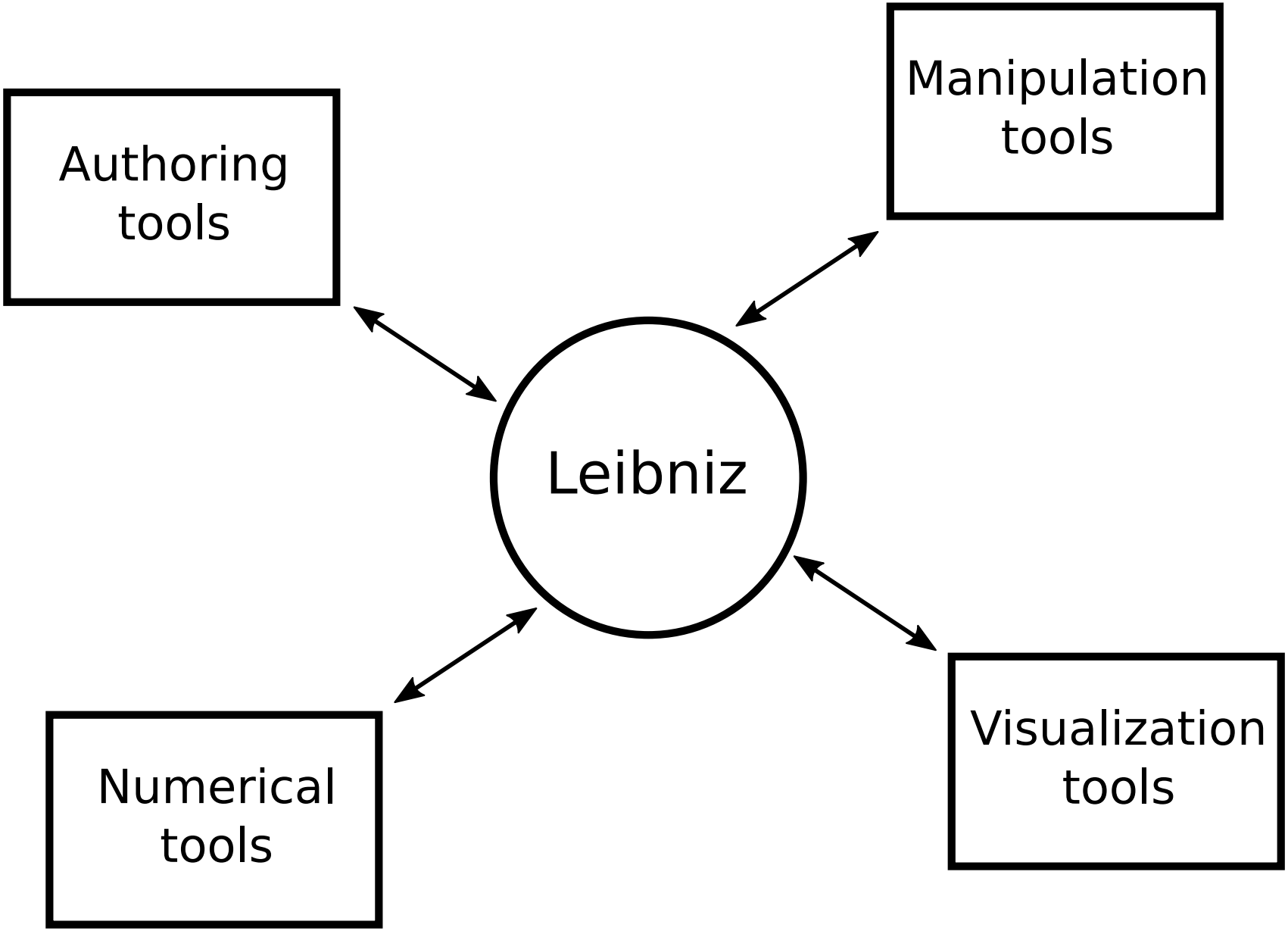


Leibniz syntax

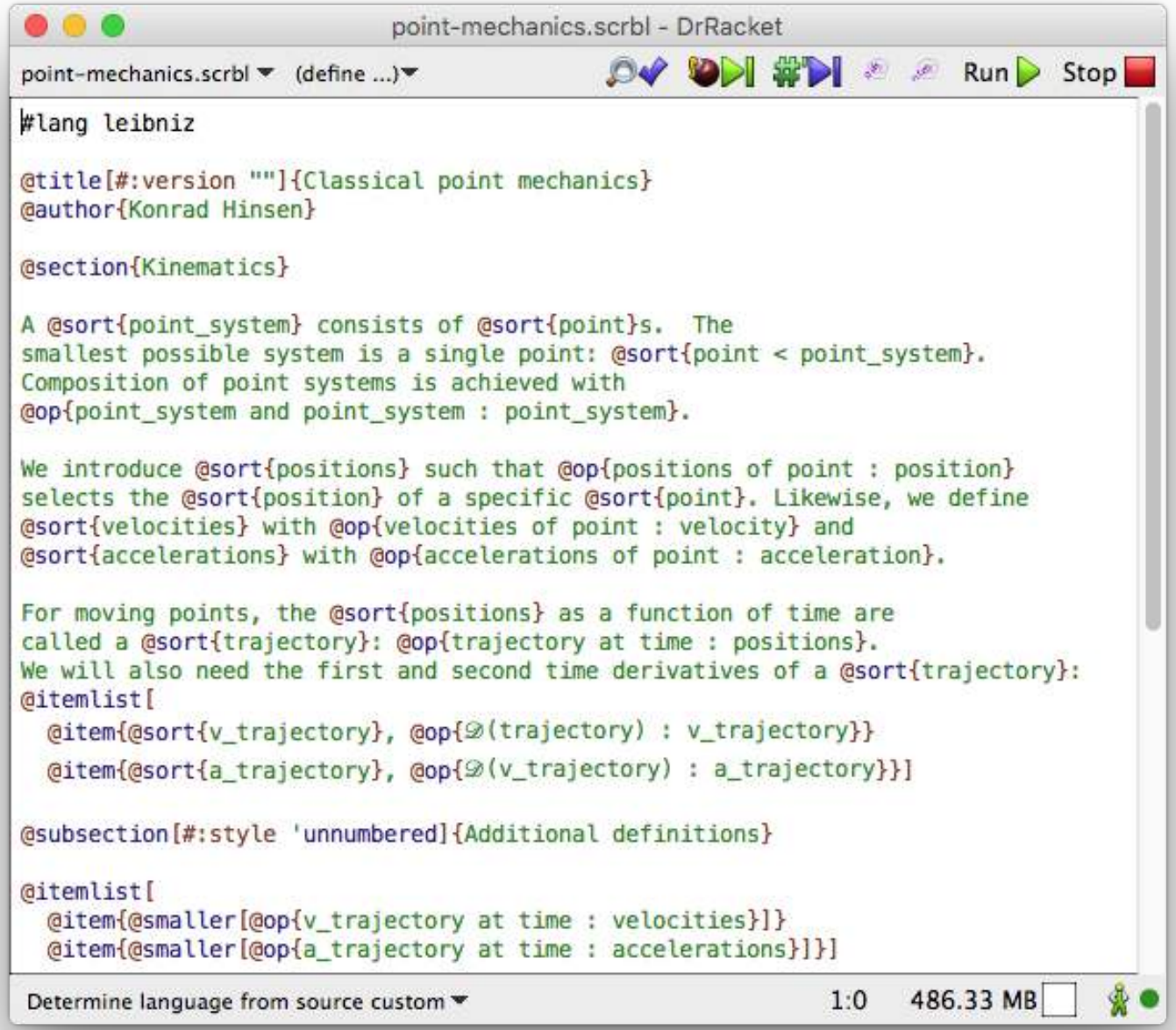
```
<context id="point-kinematics">
  <sorts>
    <sort id="point_system" />
    <sort id="velocities" />
    <sort id="a_trajectory" />
    <sort id="positions" />
    <sort id="velocity" />
    <sort id="position" />
    <sort id="v_trajectory" />
    <sort id="accelerations" />
    <sort id="acceleration" />
    <sort id="trajectory" />
    <sort id="point" />
    <sort id="time" />
  </sorts>
  <subsorts>
    <subsort sort="point" sort="point_system" />
  </subsorts>
  <ops>
    <op id="at">
      <arity>
        <sort id="a_trajectory" />
        <sort id="time" />
      </arity>
      <sort id="accelerations" />
    </op>
```

...

Software tools around Leibniz



Leibniz authoring environment (just a demo for now)



Output for human readers

Classical point mechanics

1 Kinematics

A ***point_system*** consists of ***point***s. The smallest possible system is a single point: ***point*** $\subseteq$ ***point_system***. Composition of point systems is achieved with ***point_system*** *and* ***point_system*** : ***point_system***.

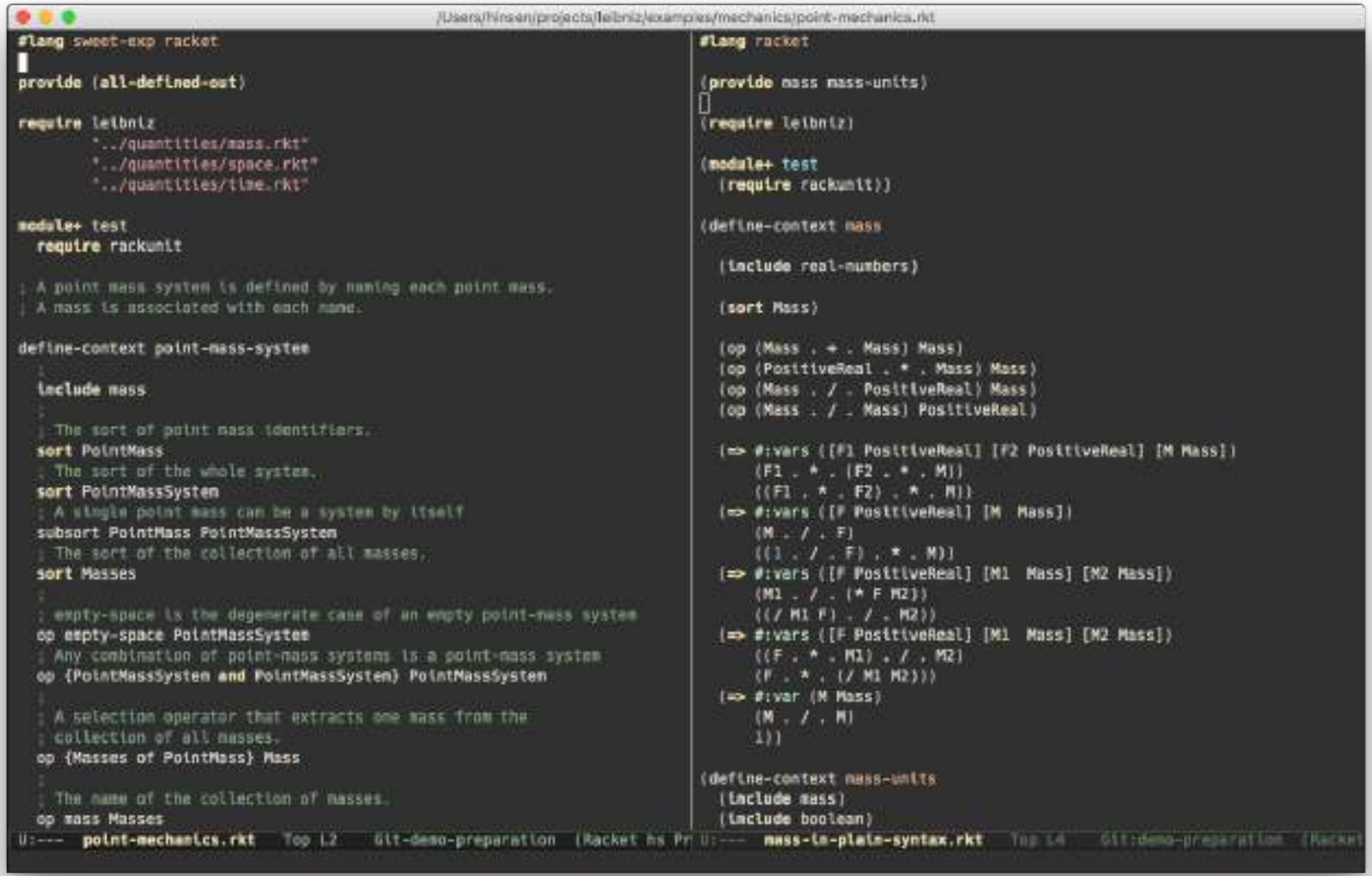
We introduce ***positions*** such that ***positions*** *of* ***point*** : ***position*** selects the ***position*** of a specific ***point***. Likewise, we define ***velocities*** with ***velocities*** *of* ***point*** : ***velocity*** and ***accelerations*** with ***accelerations*** *of* ***point*** : ***acceleration***.

For moving points, the ***positions*** as a function of time are called a ***trajectory***: ***trajectory*** *at* ***time*** : ***positions***. We will also need the first and second time derivatives of a ***trajectory***:

- ***v_trajectory***, $\mathscr{D}$(***trajectory***) : ***v_trajectory***
- ***a_trajectory***, $\mathscr{D}$(***v_trajectory***) : ***a_trajectory***

1.1 Additional definitions

“Development” syntax (works now)



Leibniz in a nutshell

- An algebraic specification language
- based on equational logic
- ... and term rewriting

Today, Leibniz is a subset of Maude with different syntax.

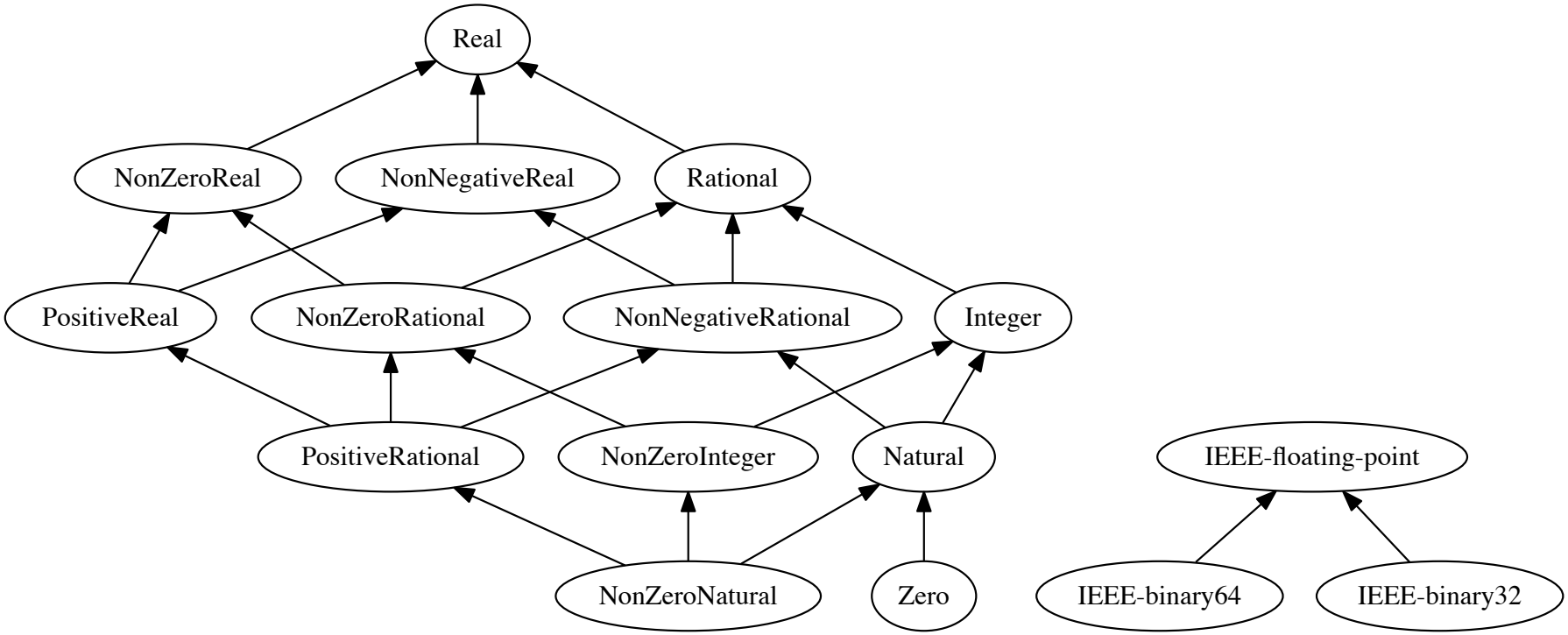
In the future, Leibniz will further diverge from Maude.

Order-sorted term algebra

- A Leibniz term is either a standard term or a special term.
- A standard term has the form $op(arg1\ \ arg2\ \ \ldots)$, or just *op* for zero arguments.
- Special terms are integers, rationals, and floats.

- Each term has an associated sort.
- Sorts have a partial order, defined by subsort relations.
- Subsort relations form a directed acyclic graph.
- The connected components of this graph are called kinds.
- All terms must be kind-correct (“static typing”)
- Sort errors within a kind can be resolved during rewriting (“dynamic typing”)

Sort graph for the number sorts



A very simple example

```
define-context mass
  ;
  include real-numbers
  ;
  sort Mass
  ;
  ; The sum of two masses is a mass.
  op {Mass + Mass} Mass
  ; The product of a positive number with a mass is a mass.
  op {PositiveReal * Mass} Mass
  ; A mass divided by a positive number is a mass.
  op {Mass / PositiveReal} Mass
  ; The quotient of two masses is a positive number.
  op {Mass / Mass} PositiveReal
```

Using subsorts

```
define-context distance
  ;
  include real-numbers
  ;
  sort Distance
  ;
  sort NonZeroDistance ; an important special case for division
  subsort NonZeroDistance Distance
  ;
  op {Distance + Distance} Distance
  op {Distance - Distance} Distance
  ;
  op {Real * Distance} Distance
  op {NonZeroReal * NonZeroDistance} NonZeroDistance
  op {Distance / NonZeroReal} Distance
  ;
  op {Distance / NonZeroDistance} Real
  op {NonZeroDistance / NonZeroDistance} NonZeroReal
```

Terms

```
define-context distance-test
  include distance
  op d Distance
  op nzd NonZeroDistance

with-context distance-test
  displayln T(d)
  displayln T(nzd)
  displayln T{2 * d}
  displayln T{2 * nzd}
  displayln T{0 * nzd}
  displayln T{d / nzd}
  displayln T{nzd / d}
```

```
Distance:d
NonZeroDistance:nzd
Distance:(* 2 d)
NonZeroDistance:(* 2 nzd)
Distance:(* 0 nzd)
Real:(/ d nzd)
[NonZeroNatural]:(/ nzd d)
```

Rewrite rules

```
define-context distance-with-rules
  ;
  include distance
  ;
  => ∀ D : Distance
     {1 * D}
     D
  => ∀ F1 : Real
     ∀ F2 : Real
     ∀ D : Distance
     {{F1 * D} + {F2 * D}}
     {{F1 + F2} * D}
  => ∀ F : NonZeroReal
     ∀ D : Distance
     {D / F}
     {{1 / F} * D}
  => ∀ F1 : Real
     ∀ F2 : Real
     ∀ D : Distance
     {F1 * {F2 * D}}
     {{F1 * F2} * D}
```

```
define-context distance-test
  include distance-with-rules
  op d Distance

with-context distance-test
  displayln  T{{2 * d} + {3 * d}}
  displayln RT{{2 * d} + {3 * d}}
  displayln RT{{2 * d} + {d / 2}}
```

```
Distance:(+ (* 2 d) (* 3 d))
Distance:(* 5 d)
Distance:(* 5/2 d)
```

Equations and transformations

```
define-context distance-with-equations
  ;
  include distance-with-rules
  ;
  op d1 Distance
  op d2 NonZeroDistance
  ;
  eq #:label eq-1
     d1
     {2 * d2}

with-context distance-with-equations
  ;
  displayln eq(eq-1)
  ;
  displayln
    A
      tr #:var (D Distance) D {D / 2}
      eq(eq-1)
      #:label eq-2
```

```
(eq #:label eq-1 d1 (* 2 d2))

(eq #:label eq-2 (* 1/2 d1) d2)
```

Unusual features

Embedding code into scholarly discourse leads to different priorities from software development.

- No namespaces, no scopes, no modularity.
- Use explicit renaming instead (supported by the authoring environment).

- Minimal built-in contexts: just numbers and booleans.
- No “standard library”.
- Re-use and adapt published libraries instead.
- Prevents the creation of large black-box code libraries.

Play with it yourself

- All the code is on GitHub.
- Written in Racket, which provides excellent support for this kind of project.

Future work

Leibniz

- Associative/commutative operators
- Built-in collections: lists, sets, maybe more
- Interfaces to external data (files, databases, ...)

Support tools

- Authoring environment
- Manipulation tools for numerical work (real $\rightarrow$ float etc.)
- Libraries for popular programming languages

Find funding

- Project outside of every funding agency's categories
- Suggestions welcome!