

Modern Mechanized Mathematics

Jacques Carette

McMaster University

Joint Lab Meeting – Friday May 29th, 2009

joint work with William M. Farmer

Overview

- 1 Context
 - Current tools
 - A little history
 - Calculemus
- 2 Problem
 - Mathematics process
 - Comparing tools
 - Challenges
- 3 Solution Pieces
- 4 Conclusion
- 5 Examples

Categories of Tools for Mathematics

Note that each category also contains many small specialized systems.

- Numeric
 - ▶ Statistics – SPSS, S-Plus, R
 - ▶ The rest (calculus, linear algebra, ...) – Matlab, Scilab
- Symbolic
 - ▶ Computation – Mathematica, Maple, MuPAD, SAGE, Axiom, Aldor
 - ▶ Proofs – Coq, Isabelle, Mizar, PVS, HOL, IMPS, Ω mega, ...
 - ▶ Both – Theorema, Focalize, HOL-Maple, PVS-Maple, MathScheme

Rough user count from various web sources:

- Stats 100-200 million
- Matlab 30-40 million
- Computer Algebra 5-6 million
- Theorem Provers (including SAT & Model checking) 400,000.
- Interactive Theorem Provers 30,000-40,000.

Categories of Tools for Mathematics

Note that each category also contains many small specialized systems.

- Numeric
 - ▶ Statistics – SPSS, S-Plus, R
 - ▶ The rest (calculus, linear algebra, ...) – Matlab, Scilab
- Symbolic
 - ▶ Computation – Mathematica, Maple, MuPAD, SAGE, Axiom, Aldor
 - ▶ Proofs – Coq, Isabelle, Mizar, PVS, HOL, IMPS, Ω mega, ...
 - ▶ Both – Theorema, Focalize, HOL-Maple, PVS-Maple, MathScheme

Rough user count from various web sources:

- Stats 100-200 million
- Matlab 30-40 million
- Computer Algebra 5-6 million
- Theorem Provers (including SAT & Model checking) 400,000.
- Interactive Theorem Provers 30,000-40,000.

Categories of Tools for Mathematics

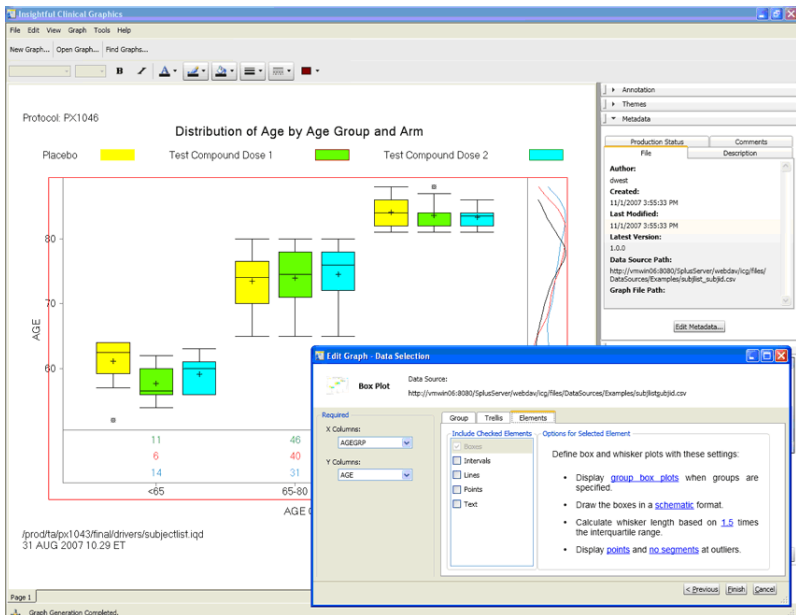
Note that each category also contains many small specialized systems.

- Numeric
 - ▶ Statistics – SPSS, S-Plus, R
 - ▶ The rest (calculus, linear algebra, ...) – Matlab, Scilab
- Symbolic
 - ▶ Computation – Mathematica, Maple, MuPAD, SAGE, Axiom, Aldor
 - ▶ Proofs – Coq, Isabelle, Mizar, PVS, HOL, IMPS, Omega, ...
 - ▶ Both – Theorema, Focalize, HOL-Maple, PVS-Maple, MathScheme

Rough user count from various web sources:

- Stats 100-200 million
- Matlab 30-40 million
- Computer Algebra 5-6 million
- Theorem Provers (including SAT & Model checking) 400,000.
- Interactive Theorem Provers 30,000-40,000.

And your experience may look like



And your experience may look like

The screenshot displays the RStudio environment with several windows open:

- R Console:** Shows R code for generating a 3D surface plot from a 2D density plot. The code includes functions for density estimation and plotting, with a conditional branch for "paysage" orientation.
- R Data Editor:** Displays a table with columns "height" and "weight". The row with height 61 and weight 123 is selected.
- Quartz (2) - Active:** Shows a 2D plot of a density function with axes labeled "long" and "lat".
- R Workspace Browser:** Lists objects in the workspace, including data frames like "women" and "x", and other objects like "dati", "g", "l", "n", "opar", "pie.sales", "pin", "scale", and "usr".
- R Package Manager:** Shows the status of installed and available packages, including "graphics", "grid", "lattice", and "methods".
- RGL device 1 (active):** Displays a 3D surface plot of a terrain, colored with a gradient from green to yellow to red.

And your experience may look like

The screenshot displays the MATLAB environment with several windows open:

- Workspace:** Lists variables such as A (a 1x3 array [1 2; 3 4]), Amp (1), Channel1-3 (1000x1 double arrays), ChannelTime (1000x1 double array), D (4-D uint8 array), and DataSet1 (1x7550 double array).
- Array Editor - L:** Shows a 2x2 grid of values:

	1	2
1	-0.16776	-0.17522
2	-0.099968	-0.092546
- Figures - Figure 1:** A 3D surface plot titled "Electrode Charge (pC)" with axes ranging from 0 to 70.
- Editor:** Contains MATLAB code:

```
18 %% Low frequency
19 % First define a sample re
20
21 y=sin(2*pi*f1*t);
22 plot(t, y);
```
- Command Window:** Shows the following commands and output:

```
>> plot (Channel13,
>> figure
>> surf (surfacemap)
>> f=theta(t) sin(2*pi*f
f =
    theta(t) sin(2*pi*f
```
- Emission Tests:** A 2D line plot showing CO₂ concentration versus Airfuel Ratio. It includes data for Test 1 (blue diamonds), Test 2 (green circles), and a Quadratic Fit (red line). The fit equation is $y = -0.29x^2 + 9x - 55$. A data point is highlighted at X: 17.4 and Y: 16.01.

And your experience may look like

Maple 17

File Edit View Insert Format Tools Drawing Data Spreadsheet Tools Window Help

Test Math Drawing Plot Animation

2D Input Times New Roman 12 B U L E

Estimation of the Model Parameters

Consider the differential equation $M y''(t) + b y'(t) + k y(t) = u(t)$.
 In terms of M , b and k , the corresponding transfer function is,

$$\frac{1}{-4 M \kappa^2 \omega^2 + 21 b \kappa \omega + k}$$

The transfer function (in the s domain) is converted to Fourier transform representation.

The estimated parameter set is given as:

$$\begin{bmatrix} k = 2.9820 & \Delta k = -0.180 & 0 \\ M = 4.9209 & \Delta M = 0.0791 & 0 \\ b = 1.9037 & \Delta b = 0.0963 & 0 \end{bmatrix}$$

The Phase and Magnitude plot for this system is:

Units (SI):

- [mm] [m] [r]
- [N] [kg] [Pa]
- [W] [J] [K]
- [T] [A] [V]
- [C] [Ω] [F]
- [H] [rad] [sr]
- [mol] [lx] [Jm]
- [S] [Wb] [Np]

Memory: 7.12M Time: 1.53s Math Mode

And your experience may look like

The screenshot displays the Mathematica software interface with three windows open:

- Trigonometry.nb:** Features a section titled "Complex Trigonometric Functions" with a 3D surface plot and a color gradient selection tool.
- SurfaceMorphing.nb:** Shows a "Morphing between two parametric surfaces" section with a 3D visualization of a torus and a grid of images illustrating the transition from a sphere to a Möbius strip.
- Graph.nb:** Displays "Graphs and Number Properties" with a grid of fractal-like patterns.

And your experience may look like

3d (sage_notebook) - Firefox

File Edit View Go Bookmarks Tools Help

http://sage.math.washington.edu:8100/70

n0 n1 n2 MSRI 2006 Summe... HOME gmail UWMATH sage notebook SAGE: System for ... JavaScript Guide

_scratch (sage_notebook) 3d (sage_notebook)

SAGE sage notebook

Help | History | Text | Text2 | Print | Evaluate | Hide | Show | Upload | Download | Interrupt | Restart

Worksheet: 3d

```
{22}
def f(x,y):
    return cos(sqrt(x^2 + y^2))
def c(x,y,z):
    return (abs(z),0,abs(sin(sqrt(x^2+y^2))))
testplot(f,range(-5,5,.1),range(-5,5,.1),colorf=c).show(axes=False)
```

{23}

Done

And your experience may look like

The screenshot shows a web browser window titled "tarski.v - WebProof - Iceweasel". The address bar contains "http://prover.cs.ru.nl/". The browser has several tabs open, including "Camino - Wikipedia, th...", "RISC Activity Databas...", "HTML XHTML Entities", and "tarski.v - WebProof".

The main content area is divided into two panes. The left pane contains a list of logical axioms and a theorem proof:

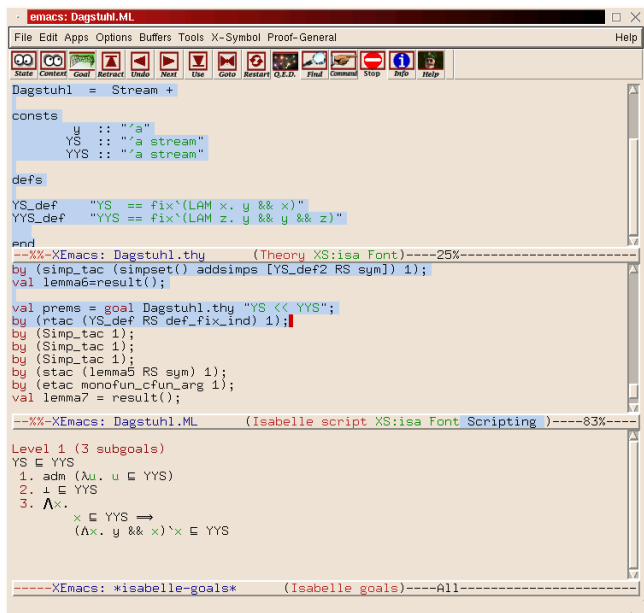
```
Variable Eq : A -> A -> Prop.  
Axiom Assym : forall x y : A, R x y -> R y x -> Eq x y.  
Axiom Trans : forall x y z : A, R x y -> R y z -> R x z.  
  
Variable f : A -> A.  
Axiom Incr : forall x y : A, R x y -> R (f x) (f y).  
  
Variable M : A.  
Hypothesis Up : forall x : A, R x (f x) -> R x M.  
Hypothesis Least : forall x : A,  
  (forall y : A, R y (f y) -> R y x) -> R M x.  
  
Theorem Tarski_lemma : Eq M (f M).  
  cut (R M (f M)).  
  intro.  
  apply Assym; trivial.  
  apply Up.  
  apply Incr; trivial.  
  apply Least.  
  intros.  
  apply Trans with (f y); trivial.  
  apply Incr.  
  apply Up; trivial.  
Qed.
```

The right pane shows the current state of the proof, including subgoals and hypotheses:

```
2 subgoals  
H : R M (f M)  
-----  
R (f M) (f (f M))  
  
subgoal 2 is:  
R M (f M)
```

The status bar at the bottom of the browser window shows "Done" and "Tor Disabled".

And your experience may look like



```
emacs: Dagstuhl.ML
File Edit Apps Options Buffers Tools X-Symbol Proof-General Help

State Context Goal Retract Undo Next Use Goto Restart Q.E.D. Find Command Stop Info Help

Dagstuhl = Stream +

consts
  y  :: "'a"
  YS :: "'a stream"
  YYS :: "'a stream"

defs
  YS_def  "YS == fix `(LAM x. y && x)'"
  YYS_def "YYS == fix `(LAM z. y && y && z)'"

end
--%X-Emacs: Dagstuhl.thy (Theory XS:isa Font)----25%-----
by (simp_tac (simpset() addsimps [YS_def2 RS sym]) 1);
val lemma6=result();

val prems = goal Dagstuhl.thy "YS << YYS";
by (rtac (YS_def RS def_fix_ind) 1);
by (Simp_tac 1);
by (Simp_tac 1);
by (Simp_tac 1);
by (stac (lemma5 RS sym) 1);
by (etac monofun_cfun_arg 1);
val lemma7 = result();

--%X-Emacs: Dagstuhl.ML (Isabelle script XS:isa Font Scripting )----83%-----

Level 1 (3 subgoals)
YS ⊆ YYS
1. adm (λu. u ⊆ YYS)
2. ⊥ ⊆ YYS
3. Λx.
   x ⊆ YYS ⇒
   (Λx. y && x) \x ⊆ YYS

-----XEmacs: *isabelle-goals* (Isabelle goals)----All-----
```

Pre-history

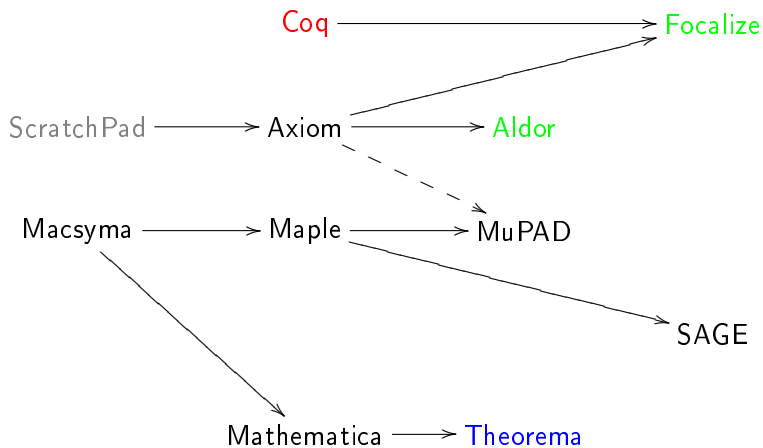
Interactive Theorem Provers:

1968	Automath	Netherlands, de Bruijn
1971	nqthm	US, Boyer & Moore
1972	LCF	UK, Milner
1973	Mizar	Poland, Trybulec

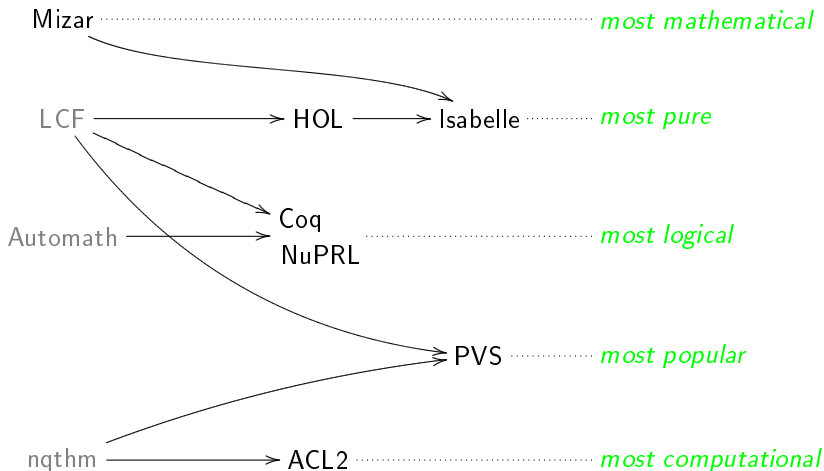
Computer Algebra:

1963	Schoonship	Netherlands, Veltman
1965	Formula Algol	US, Perlis
1967	Macsyma	US, Engelman & Moses
1968	Reduce	UK, Hearn
1968	ALTRAN	US, Hall
1971	ScratchPad	US, Jenks

A short history – Computer Algebra



A short history – (Interactive) Theorem Proving



slide courtesy of Freek Wiedijk

Calculus! (Let us Calculate)

Gottfried Wilhelm Leibniz

Mission

The CALCULUMUS interest group is a loosely coupled network of research groups and individuals interested in joining forces for the design of a new generation of mathematical software systems and computer-aided verification tools based on the integration of the deduction and the computational power of deduction systems and computer algebra systems respectively.

Meetings since 1998, generally alternating between deduction and “computation” conferences.

The Mathematics Process

In mathematics, we

- define new concepts, define new notations
- state, in convenient ways, problems to be solved
- conduct experiments
- make conjectures
- prove theorems
- gain *insight* through proofs, computations and visualization
- turn theorems into algorithms
- communicate our results
- reuse previous results

Our goal: build a tool that helps us do all of that.

Current (oversimplified)

	TP	CA
Orientation	semantics	results
Mathematics	abstract	concrete
Steps	deductive	computational
Rigor	high	low
Context	explicit, axiomatic	implicit, algorithmic
Speed	very slow	fast
Ease of use	low	medium-high

Challenges

- 1 Put computation and proof on equal footing.
 - ▶ Proofs need computations (Poincaré principle), and
 - ▶ computations need proofs (side-conditions)
- 2 Symbolic computation is on **syntax** but about **semantics**
- 3 Efficiency matters. Correctness matters.
- 4 Ease of use is important. Proofs should be precise.
- 5 Modularity. Humans work in very rich theories.
- 6 Small team. Huge task (~ 150 person-years).
- 7 Concrete representations. Sufficient abstraction.
 - ▶ sparse polynomial data-structure. ${}_2F_1(1/2, -3; 1; x)$ is a polynomial.

Solution pieces: Biform theories

Issue

Computation and proof on equal footing.

Definition

A Biform theory consists of *axioms*, *algorithms* and *meaning formulas* linking axioms and algorithms.

Intuition for programmers: specifications and programs.

A “step” in a development can be either from a deduction rule or the application of an algorithm. These are encapsulated in **transformers**.

Constructively \Rightarrow program extraction. But also allows *black boxes*.

Solution pieces: Chiron

Issue

Symbolic computation is on syntax but about semantics.

Chiron is a typed logic based on von Neumann-Bernays-Gödel (NBG) set theory, with facilities to reason about undefinedness as well as the syntax of expressions.

Intuition: a typed logic and a dependently-typed lambda-calculus with `eval` and `quote`.

Lots of convenient features, including *definite* and *indefinite description*, *undefined* as well as *non-denoting* expressions. β -reduction is definable in the logic.

Allows computation (and proof) with objects whose denotations are non-constructive (eg: algorithm for normal-form of piecewise-defined functions over classical \mathbb{R})

Solution pieces: Genericity and Generativity

Issues

Efficiency matters. Correctness matters.

Concrete representations. Sufficient abstraction.

Intuition: C++ templates done right.

Mathematics is full of generic concepts and algorithms which can be proven “correct” once and for all. However, efficiency depends crucially on concrete representations. The trick: correctly **specializing** generic algorithms to get efficiency for **concrete** cases.

The tools: typed metaprogramming and partial evaluation.

Solution pieces: Context and Automation

Issue

Ease of use is important. Proofs should be precise.

Context is used pervasively by humans when *doing* mathematics.

- Always reason in a “local context”
- Make specifying context easy (eg: Calculus, Circuit Design, ...)
- Domain Specific Languages everywhere!
- Automate everything that can be (eg: type inference, definedness guards, simplification, etc)

Note: Type inference is undecidable in Chiron. Use abstract interpretation to obtain as good an approximation as possible.

Use partial evaluation (on theories!) to remove most definedness guards automatically upon specialization.

Solution pieces: Little Theories & High Level Theories

Issue

Modularity. Humans work in very rich theories.

The **Little Theories** idea is to assemble larger theories out of small components – modularity taken to an extreme. Note that mathematics is usually presented (in textbooks) in this manner.

A **High Level Theory** (HLT) is a very rich theory (think 800 page textbook on calculus) in which it is **convenient** to work.

Intuition: Mathematics is naturally a giant network of definitions, theorems and algorithms (little theories). Doing mathematics in a HLT hides the network and provides an amazing IDE.

The theory network is the **developer's view** while the HLTs are the **end user's view**.

Solution Pieces: Trustable Communication

Issue

Small team. Huge task (~ 150 person-years).

Idea: Use **translations** and **interpretations** between theories to **transport** results from one system to another.

Intuition: Services! Build system out of (trustable) components.

A service is formally a **transformer** in a biform theory. A **result** is a **theorem**.

Eg: $\text{plus} : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$ is a simple transformer, which can be used as $\text{plus}(5,6)$ and will return the theorem $5 + 6 = 11$.

Technically requires that services be formally specified. Interpretations can be very difficult to construct.

The “solution pieces” of the preceding 6 slides correspond to Publication(s) of J. Carette and/or W.M. Farmer & co-authors.

Applications

Mathematics has applications everywhere, but we feel that mechanized mathematics can bring the biggest benefit to:

- Software development – Software Certification
- Model-based development in science and engineering
 - ▶ eg: parametric PDE + objective function to optimize
- Building a digital mathematics library
- Mathematics education (via automated TA and interactive lab)

Note: mathematics research is purposefully absent from this list.

Machines should work. People should think.

Richard Hamming

Long term goal: mechanize the mathematics process.

Short term goal: integrate theorem proving and computer algebra

We need a new system because all current systems:

- 1 have chosen (in their fundamental design) to tackle only some of the challenges, and
- 2 are being improved only incrementally.

A Biform Theory, using Chiron

Theory Derivative-Real1D {

derivative : $(\mathbb{R} \rightarrow \mathbb{R}) \rightarrow (\mathbb{R} \rightarrow \mathbb{R})$

axiom $\forall f : (\mathbb{R} \rightarrow \mathbb{R}). \forall x : \mathbb{R}.$

$$\text{derivative}(f)(x) \simeq \lim_{\epsilon \rightarrow 0} \frac{|f(x + \epsilon) - f(x)|}{\epsilon}$$

diff : $E_{(\mathbb{R} \rightarrow \mathbb{R})} \rightarrow E_{(\mathbb{R} \rightarrow \mathbb{R})}$

meaning $\forall f : E_{(\mathbb{R} \rightarrow \mathbb{R})}. \llbracket \text{diff}(f) \rrbracket \simeq \text{derivative}(\llbracket f \rrbracket)$

}

If you wanted to use Maple's diff here, you would have to change that meaning formula to

$$\forall f : E_{(\mathbb{R} \rightarrow \mathbb{R})}. (\text{total}(f) \wedge \text{differentiable}(f)) \Rightarrow (\llbracket \text{diff}(f) \rrbracket \simeq \text{derivative}(\llbracket f \rrbracket))$$

A Biform Theory, using Chiron

Theory Derivative-Real1D {

derivative : $(\mathbb{R} \rightarrow \mathbb{R}) \rightarrow (\mathbb{R} \rightarrow \mathbb{R})$

axiom $\forall f : (\mathbb{R} \rightarrow \mathbb{R}). \forall x : \mathbb{R}.$

$$\text{derivative}(f)(x) \simeq \lim_{\epsilon \rightarrow 0} \frac{|f(x + \epsilon) - f(x)|}{\epsilon}$$

diff : $E_{(\mathbb{R} \rightarrow \mathbb{R})} \rightarrow E_{(\mathbb{R} \rightarrow \mathbb{R})}$

meaning $\forall f : E_{(\mathbb{R} \rightarrow \mathbb{R})}. \llbracket \text{diff}(f) \rrbracket \simeq \text{derivative}(\llbracket f \rrbracket)$

}

If you wanted to use Maple's diff here, you would have to change that meaning formula to

$$\forall f : E_{(\mathbb{R} \rightarrow \mathbb{R})}. (\text{total}(f) \wedge \text{differentiable}(f)) \Rightarrow (\llbracket \text{diff}(f) \rrbracket \simeq \text{derivative}(\llbracket f \rrbracket))$$

Code Generation - algorithm families

Encode “design concepts” present in a “software product line” composed of variants of an algorithm.

Case study – Gaussian Elimination & LU Decomposition. Rationale: found 80 different implementations in Maple’s library.

Result: generated code is *identical* to human-written versions.

```
module GVCI = GenericVectorContainer(IntegerDomainL)
module LA = GenLA(GVCI)
```

```
module GenIV5 = GenGE(struct
  module Det =      AbstractDet
  module PivotF =  FullPivot
  module PivotRep = PermList
  module Update =  FractionFreeUpdate
  module Input =   InpJustMatrix
  module Output =  OutDetRank end)
```

Design Concepts

Design Dim.	Abstracts
Domain	Matrix values
Normalization	domain needs it?
ZeroEquivalence	decidability of $= 0$
Representation	Matrix representation
Fraction-free	use of division
Pivoting Strategy	ex:use length?
Pivoting Choice	no/row/column/total
Pivot Rep	list, array, matrix
Full Division	division in domain
Rank	track rank?
Determinant	determinant tracking
Output	choice of output
Packed	L and U as one?
Lower	track lower L ?

Design space for LU Decomposition ≥ 25 dimensional!

Design Dim.	Abstracts
Code Rep	codegen options
UserInformation	user-feedback
Augmented	matrix is augmented
Input	choice of input
Logging	trace algorithm
Structure	ex: tri-diagonal
Warning	warn on 0? pivot
In-place	res. stored in input
Error-on-singular	input (near) singular
Conditioning	cond. numb. est.

Little Theories

```
Theory CarrierType carrier:type
Theory Carrier extends CarrierType U:carrier
Theory PointedCarrier extends Carrier e:U
Theory Unit extends Carrier property singular(x) := forall y in U. x = y
Theory One extends Unit axiom singular(e)
Theory BinaryOperation extends Carrier op:(U, U)->U
Theory PointedBinaryOperation extends Carrier combines BinaryOperation, PointedCarrier
Theory Two combines One, One along Carrier
Theory Bool using Two with e'1 = true, e'2 = false
...
Theory Cancellative extends Magma combines LeftCancellative, RightCancellative
Theory Unital extends Magma using Identity
Theory QuasiGroup extends Magma using Cancellative
Theory Loop extends Magma combines Unital, QuasiGroup
Theory SemiGroup extends Magma using Associativity
Theory Band extends SemiGroup using Idempotency
Theory Group extends Magma combines Loop, Associativity
Theory AbelianGroup extends Group using Commutativity
Theory Monoid extends Magma combines Unital, SemiGroup
Theory CommutativeMonoid extends Monoid using Commutativity
```