Building and Using a Library of Formal Mathematics

Robert L. Constable

Cornell University

Reading, England
October 2002

Questions and Issues

What is formal mathematics?

Why is it interesting?

How much is there, and why is it produced?

Why is it valuable to collect in a formal library?

What technical challenges are associated with it?

- Mathematics/Logic
- Computer Science
- Information Science

How might formal libraries evolve?

How do formal libraries fit into the agenda of modern science?

What Is Formal Mathematics?

Broad view to Narrow view

Consider computational mathematics

If we implement the computations on a computer, then elements of the mathematics are formal:

- We parse the code
- We type check the code
- We execute the code

What the machine does is formal.

Computational Mathematics

Computational mathematics has informal elements as well; for instance, reasoning about the code. Some reasoning steps might also be formal; for example:

- Direct computation
- Symbolic computation
- Extended type checking

Proofs

Proofs about code use direct computation and symbolic computation.

For example, suppose $root(n)$ computes the integer square root of a natural number, e.g.

$$root(n^2) = n$$

$$root(24) = 4$$

(Over $\mathbb{N}$ we don't try for $root(x)^2 = x$.)

Root Program

$$
\begin{aligned}
root(n) == \boldsymbol{if}\ n = 0\ \boldsymbol{then}\ 0 \\
\boldsymbol{else}\ \boldsymbol{let}\ r = root(n-1)\ in \\
\boldsymbol{if}\ (r+1)^2 > n\ \boldsymbol{then}\ r\ \boldsymbol{else}\ r+1\ \boldsymbol{fi} \\
\boldsymbol{fi}
\end{aligned}
$$

We argue by direct computation that

$$root(0) = 0, \qquad root(1) = 1$$

We argue symbolically that if $r^2 \leq n-1$ and $(r+1)^2 > n$ then

$$r^2 \leq n < (r+1)^2$$

We also reason about programs of a more general form ("general recursion" as well as primitive).

For example, here is another square root locator:

$$r := 0$$
$$\textbf{\textit{while}}\ (r+1)^2 \le n$$
$$\quad \textbf{\textit{do}}\ \ r := r+1\ \textbf{\textit{od}}$$
$$\{r^2 \le n < (r+1)^2\}$$

Loop invariant method establishes the assertion in the program:

$$r := 0$$

$$\{r^2 \le n\}$$

$$\textbf{\textit{while}}\ (r+1)^2 \le n$$

$$\textbf{\textit{do}}\ \{(r+1)^2 \le n\}$$

$$r := r+1$$

$$\{r^2 \le n\}$$

$$\textbf{\textit{od}}$$

$$\{r^2 \le n\}\ \ \&\ \{(r+1)^2 > n\}$$

Recursive form of the while program:

$$\textbf{\textit{while}}\ (r+1)^2 \le n \quad \textbf{\textit{do}}\ r := r+1\ \textbf{\textit{od}}$$

$$wrt(n,r) == \textbf{\textit{if}}\ (r+1)^2 \le n\ \textbf{\textit{then}}\ wrt(n,r+1)\ \textbf{\textit{else}}\ r$$

$$r := 0$$
$$\textbf{\textit{while}}\ (r+1)^2 \le n \quad \textbf{\textit{do}}\ r := r+1\ \textbf{\textit{od}}$$

$$wrt(n,0)$$

Theorem $\forall n : \text{¥} \,.\, wrt(n,0)^2 \leq n < (wrt(n,0)+1)^2$

Proof by induction on n

base $wrt(0,0) = 0$

induction let $r = wrt(n-1,0)$

Cases on $(r+1)^2 > n$ or $(r+1)^2 \leq n$:

- case $(r+1)^2 > n$ (see next slide)
- case $(r+1)^2 \leq n$ (see subsequent slide)

Qed

- case $(r+1)^2 > n$

  then $wrt(n,r) = r = wrt(n-1,r)$,

  hence $r^2 \le n < (r+1)^2$ and

  for all $y < r$, $wrt(n,y) = wrt(n,r)$

  since $y+1 \le r$, $(y+1)^2 \le r^2 < n-1$,

  thus $wrt(n,r) = wrt(n,0)$.

- case $(r+1)^2 \leq n$

  then $wrt(n,r) = wrt(n,r+1)$ and

  for all $y \leq r,\ \ wrt(n,y) = wrt(n,r+1)$.

  Also, $wrt(n,r+1) = r+1$ since $((r+1)+1)^2 > n$

  follows from $n-1 < (r+1)^2 \leq n$ by Arithmetic.

An aside for later use

In constructive formal logics such as Coq, Nuprl, and MetaPRL, we can extract code from a proof of:

Theorem (Root) $\forall n : ¥ \,.\, \exists r : ¥ \,.\, r^2 \leq n < (r+1)^2$

Proof by induction on n

base take $r = 0$.

induction, suppose $r_0^2 \leq n - 1 < (r_0 + 1)^2$.

(continued next slide)

Cases on $(r_0+1)^2 > n$ or $(r_0+1)^2 \le n$

case $(r_0+1)^2 > n$ take $r = r_0$

since $r_0^2 \le n < (r_0+1)^2$

case $(r_0+1)^2 \le n$ take $r = r_0+1$

since $((r_0+1)+1)^2 > n$

Qed

Typing While Loops

The Booleans, $\mathbb{B}$, have constants *true, false*

For $y = \mathbb{B}$, we type the loop as :

$$\textit{\textbf{while}}\ y\ \textit{\textbf{do}}\ x := 0\ \textit{\textbf{od}} \in \overline{¥}\,.$$

The meaning is that if the loop halts, then its value is of type ¥ .

Partial Objects

More generally we say:

$$a \in \overline{A}$$

If a halts implies that $a \in A$. The elements of A are partial objects. We say that $\overline{A}$ is a bar type.

Computational Reasoning

Reasoning computationally about “bar types” introduces concepts which are not consistent with some forms of non-computational reasoning involving P or $not\ P$.

Next is an example.

Absolute Undecidability

Theorem $\neg \exists h : \overline{¥} \rightarrow \mathbb{B}. \forall n : \overline{¥}.$

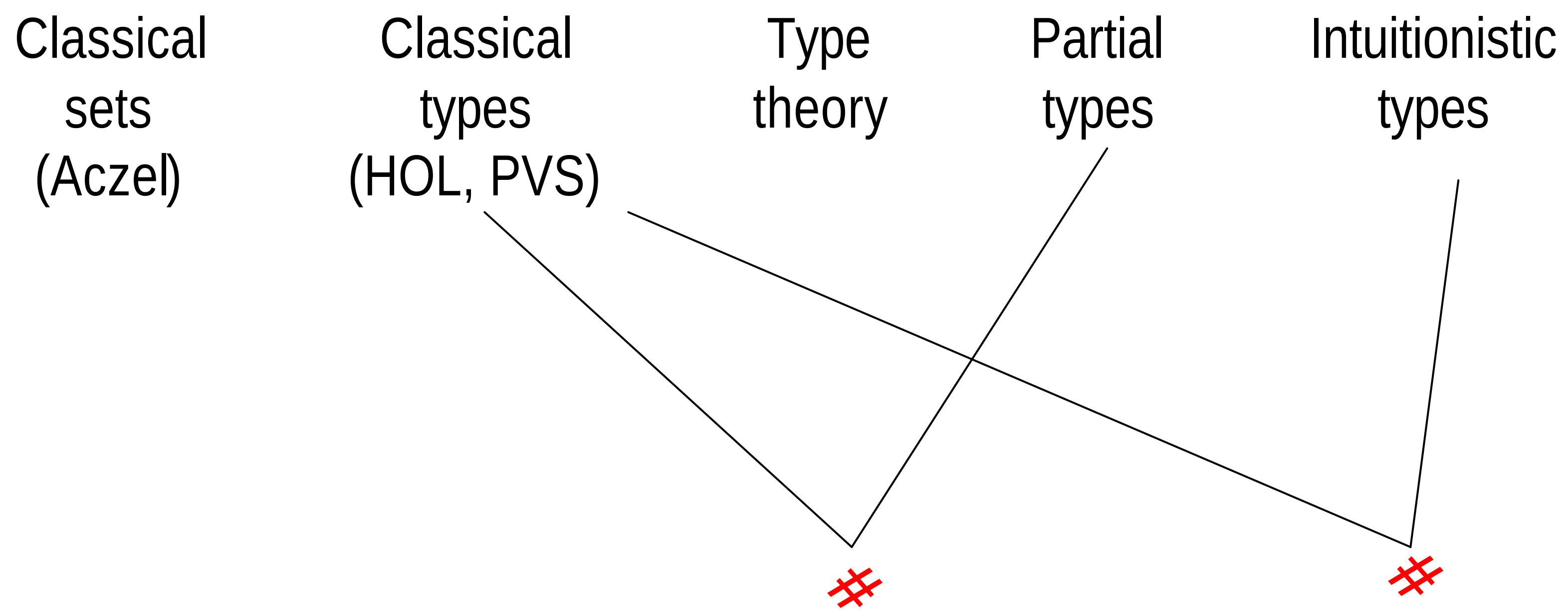
$$(h(x) = true \; iff \;\; x \; halts)$$

The functions in $\overline{¥} \rightarrow \mathbb{B}$ are computable, so this says that halting is not decidable.

If P or $not \; P$ holds, then $\overline{¥} \rightarrow \mathbb{B}$ includes non-computable functional relations, including a halting detector.

Living with Incompatible Theories

Using the language of types, we can define incompatible theories:



Importance of Metamathematics

Establishing that HOL and Nuprl domain theory are incompatible is a result in metamathematics (logic).

Outline

- The ONR Digital Library Project
- Concepts for Formal Digital Library (FDL) design
- Current status of FDL
- Questions and issues

Objective of ONR Program:

To create a digital library of algorithms and constructive mathematics useable for program and software construction.

What Does "Formal" Mean?

The BAA refers to machine-checked mathematics presented in a consistent formal logical theory that is implemented.

This meaning of "formal" is technical. It is more narrow than what many people mean in daily use.

Goals

1. Build a semantics-based interactive logical library infrastructure
2. Create, collect and organize formal computational mathematics content
3. Apply the formal interactive DL in designing and creating reliable software (especially for CIP/SW)

Benefits to Society

- Basis for highly reliable and responsive software
- Acceleration of scientific discovery

  mathematics

  computer science

  computational science

  metamathematics
- Wider access to content (participatory science)
- Topics in a new science of information

  formalized mathematics publication

  scholarly publication in general (arXiv)

  quantitative metamathematics

Strategy

1. Attract a community of contributors who share formal knowledge and the connected mathematically literate articles

2. Account for correctness in a multi-logic, multi-prover (including tactic-style) environment

3. Provide semantics-based library services at many scales

Challenges and Problems

1. Community using formal proofs is relatively small
   - Market for formal proofs is small
     – proof technology not widely used in software
     – proof technology not widely used in science and math
     – proof technology not widely used in education
   - Formal proving is still hard work
     – expansion factor
     – shallow base of basic mathematical facts
     – demanding skill set (programming + math + design)

Challenges and Problems

2. Community is disconnected
   - Each group uses a different system
   - Almost no sharing (logical difficulties, practical ones)
   - Systems change or go extinct

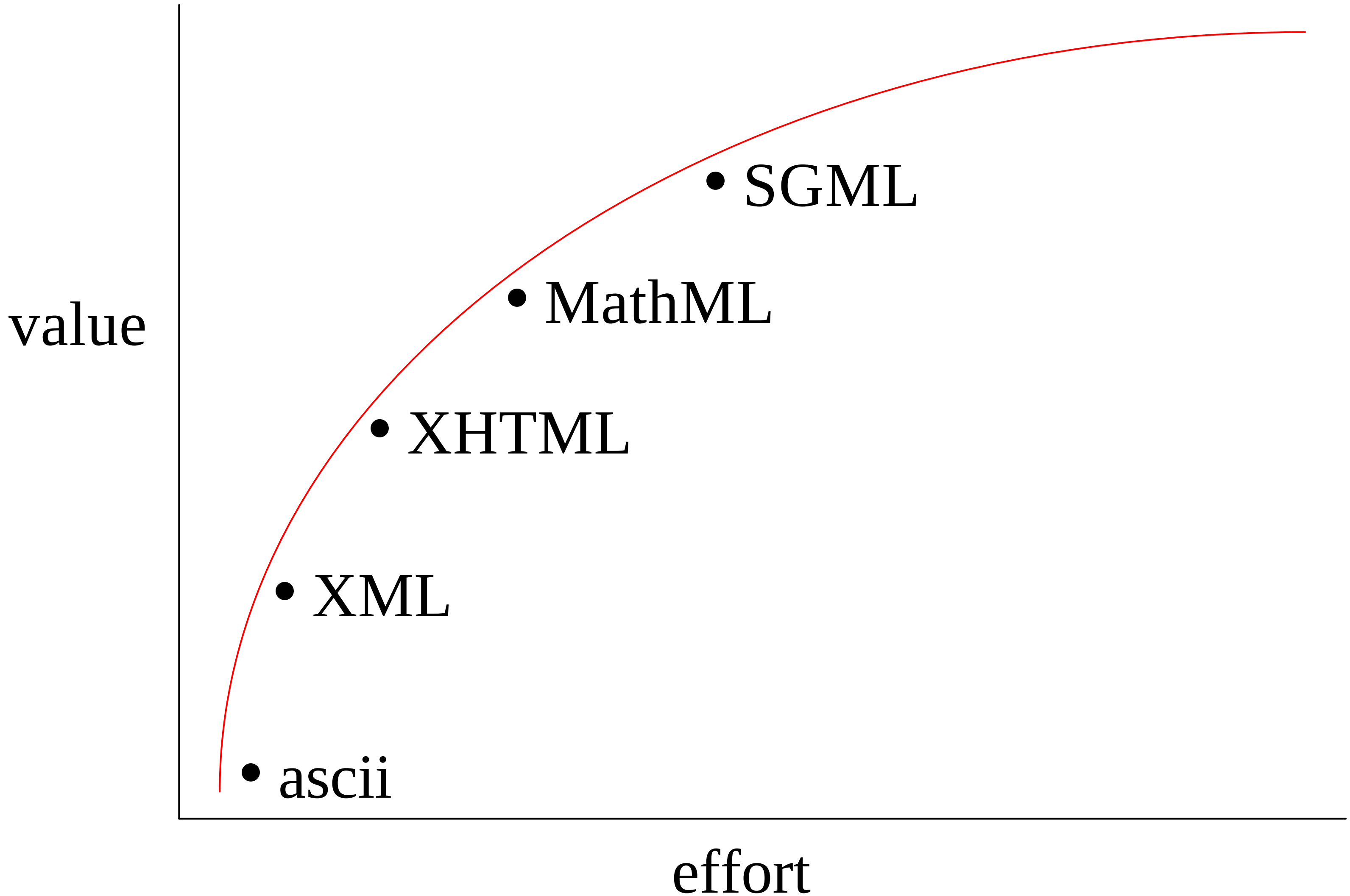
Digital Library Approach to the Challenges

1. Widen the community by
   - library will increase the services provided
   - library will decrease the effort to create proofs (seen from experience)
2. Connect the community through a common service – the digital libraries approach

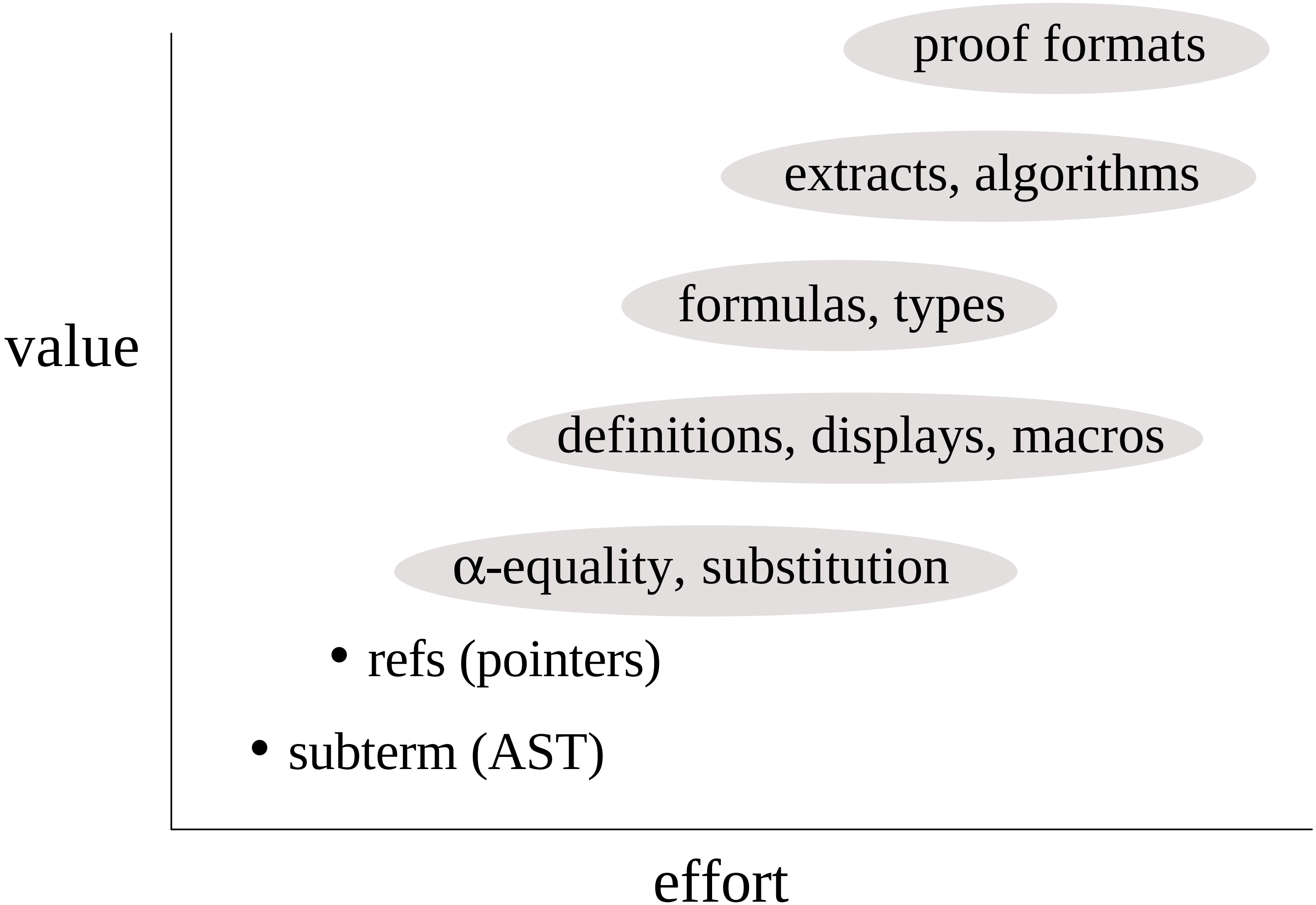
Outline

- The ONR Digital Library Project
- Concepts for Formal Digital Library (FDL) design
- Current status of FDL
- Questions and issues

DL Shared Data Formats



Formal DL Data Formats



Terms (Abstract Syntax Trees)

$t = op(t;\ldots;t)$ for t a term

$Term = op \times Term\ List$

with binding structure

$op(\overline{v}_1.t_1;\ldots;\overline{v}_n.t_n)$ $\overline{v}_i$ list of binding variables

$Op = OpName\{i_1 : F_1;\ldots;i_k : F_k\}$

i can be reference objects or values

Conceptual Basis for Design and Implementation

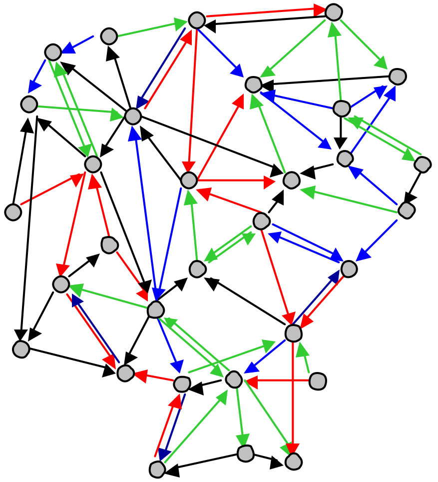
Important features

- Logical library keeps track of

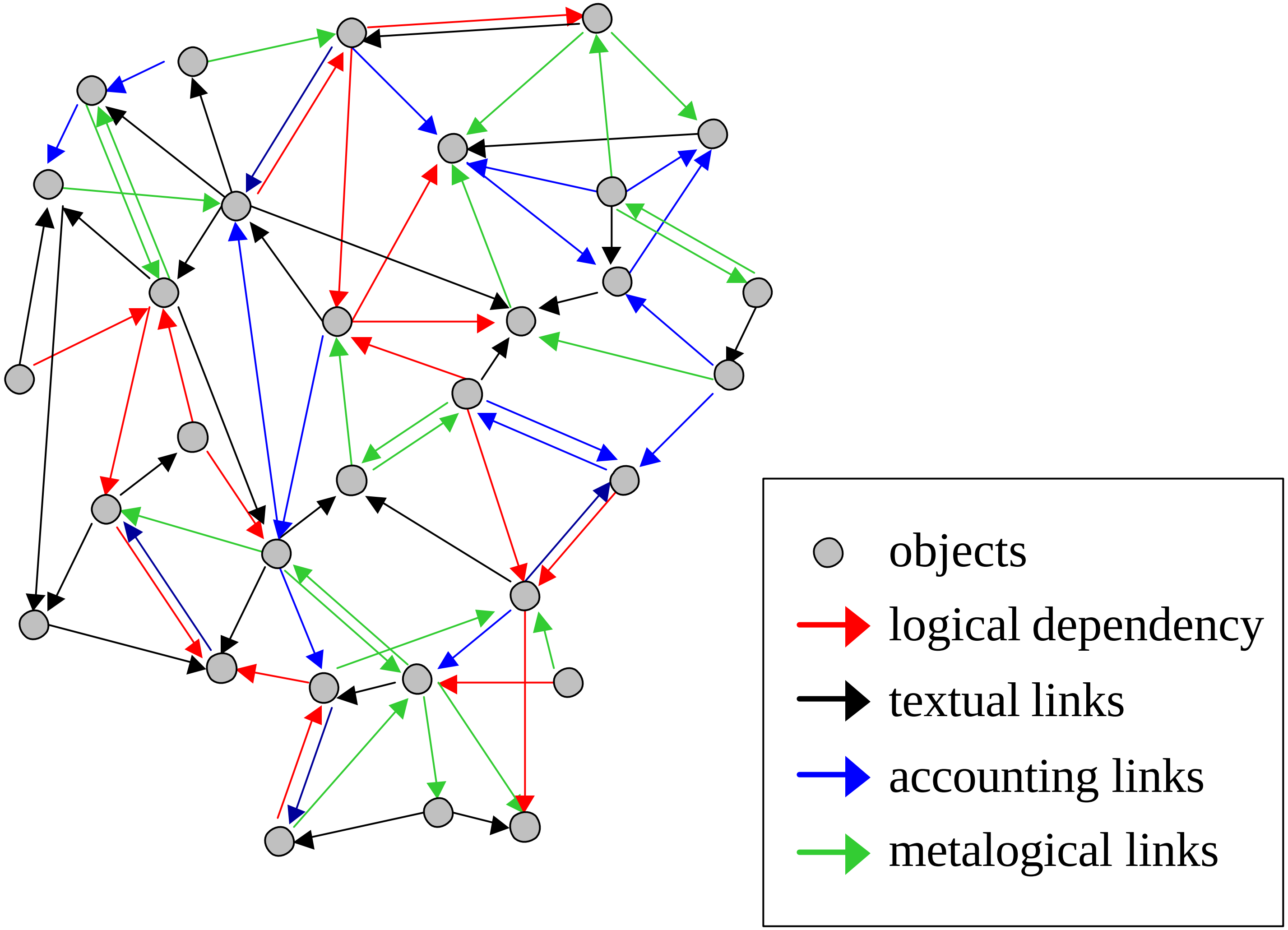
  evidence

  dependencies

  objects form a graph



Information Graph of the FDL



FDL contains formal objects

- rules
- definitions
- algorithms, code
- conjectures
- specifications
- theorems
- inferences
- proofs, partial proofs
- certificates

Inferences

$$\boxed{\overline{H}_1 \vdash G_1 \text{ by } J_1,}\ \boxed{\overline{H}_2 \vdash G_2 \text{ by } J_2,}\ \ldots,\ \boxed{\overline{H}_n \vdash G_n \text{ by } J_n}$$

$$\overline{H} \vdash G \text{ by } J$$

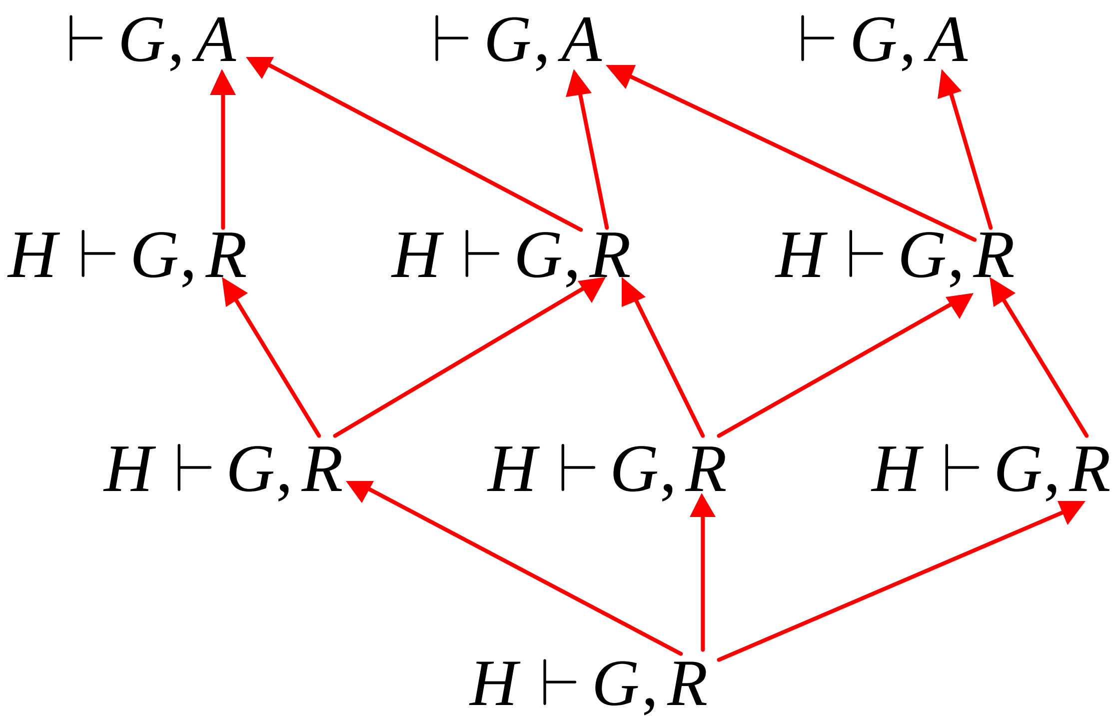
$\overline{H}_i$ a list of formulas (terms)

G_i a formula (term)

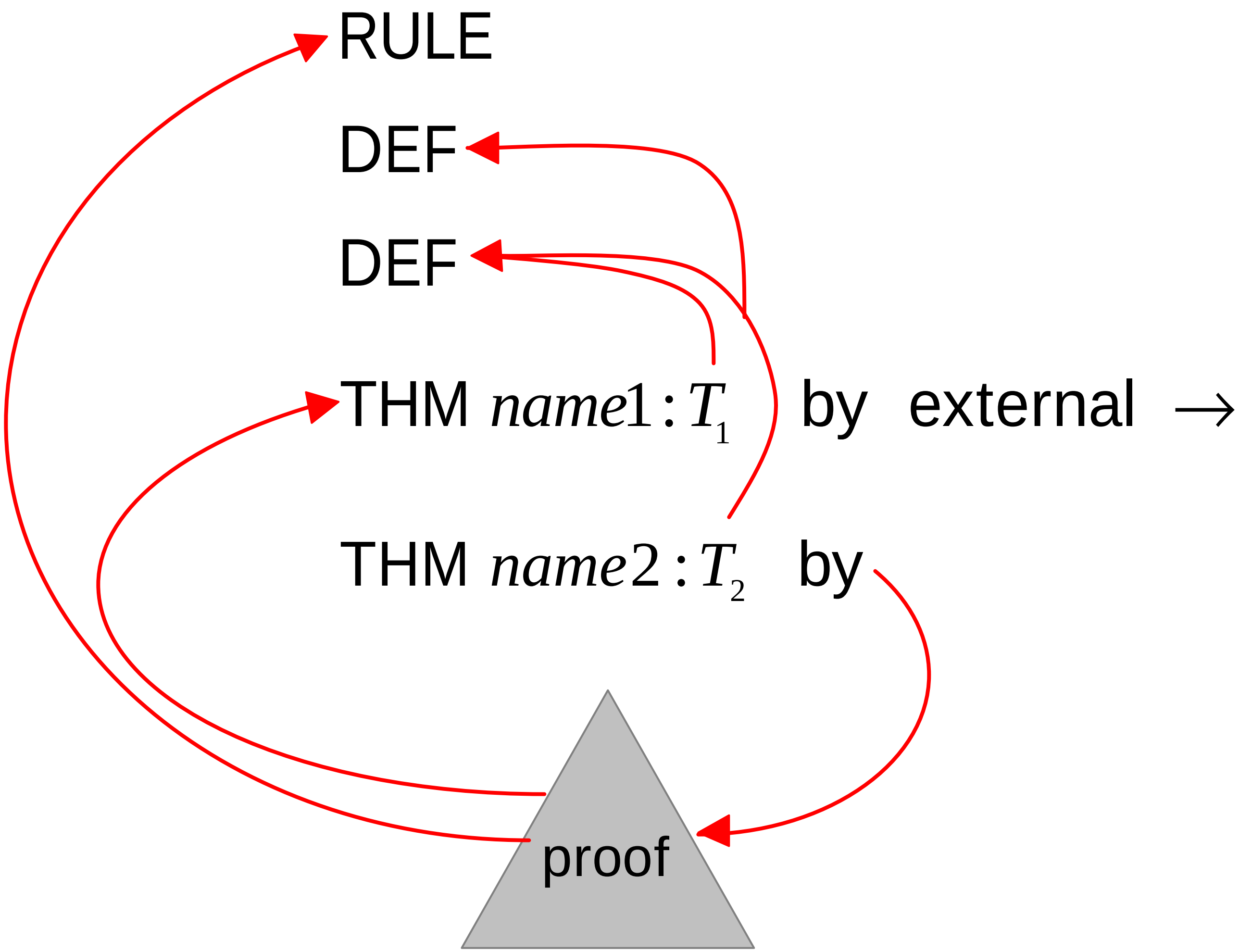
J_i a justification (rule, tactic)

Proof

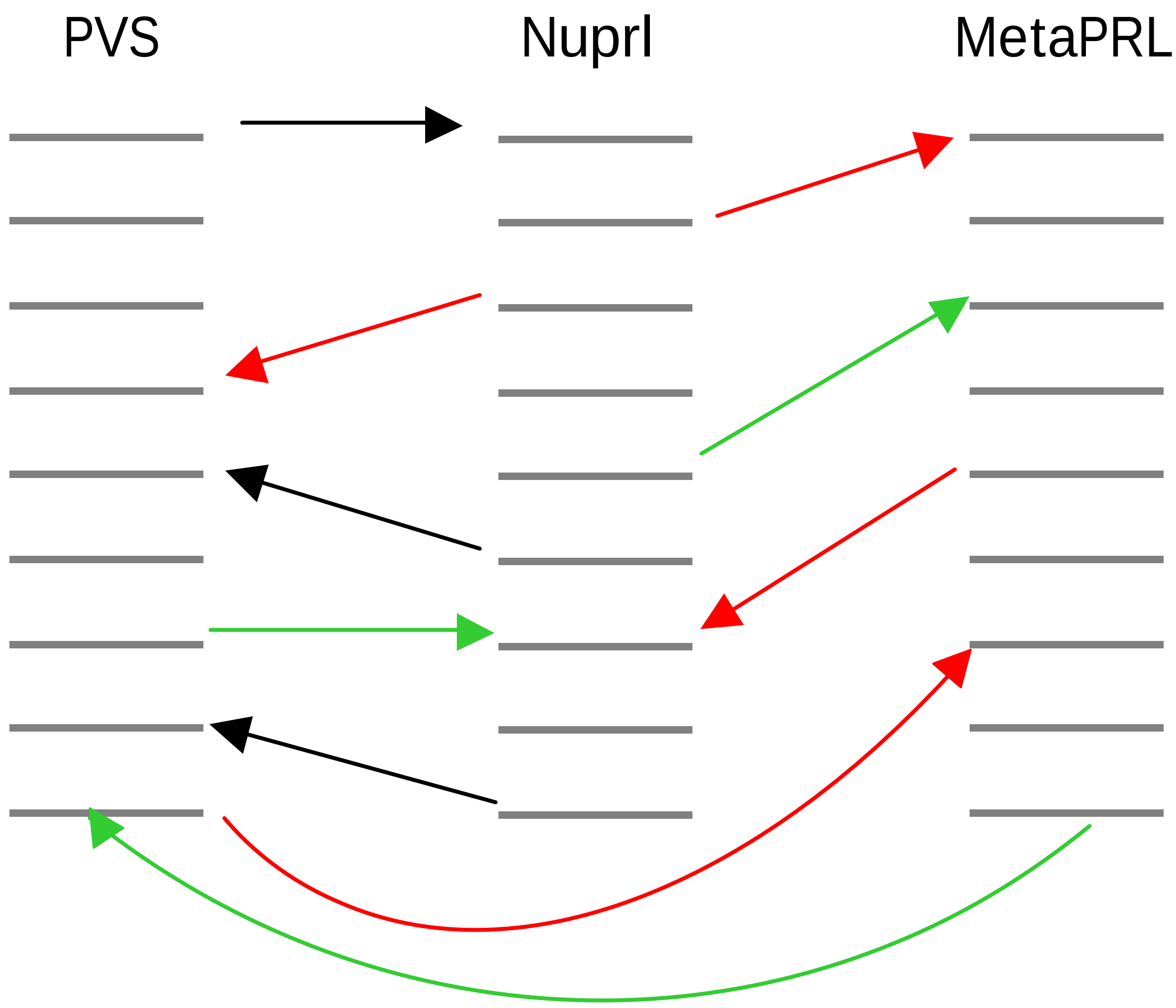
A proof is a dag of inferences



Example of Dependencies



FDL allows sharing among collections

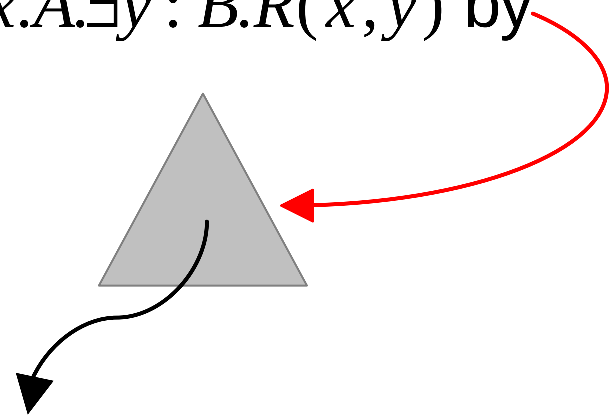


FDL is interactive

- Can create new definitions, claims, conjectures
- Can interactively build proofs
- Can execute algorithms, extracts
- Can search for information
- Can display dependencies
- Can transform entire collections, theories

FDL supports algorithmic mathematics

THM: $\forall x.A.\exists y : B.R(x, y)$ by



THM: $\exists f : A \to B.\forall x.A.R(x, f(x))$

THM: $\forall x : A.R(x, f_0(x))$

THM: f_0 in $\{g : A \to B \mid P(g)\}$

Concepts for FDL Design

- FDL provides an experimental publication medium

  Can solicit exemplary contributions

  - hybrid articles – formal and informal
  - elegant formalizations
  - challenging formalizations
  - expository articles
  - hypothetical formalizations

  Articles directly include shared material

FDL performs archival functions

Automath system Auto QE checked the following formalization of Landau's Grundlagen (August 17, 2004).

Coq 5.0 created the following extract for the Fundamental Theorem of Algebra (June 14, 2003).

Nuprl 5 checked that Total Order (TO) protocol satisfies P (June 5, 2003).

MetaPRL compiler produced C code from TO, and P is preserved (October 19, 2003).

PVS 2.4 proved Menger's theorem (September 15, 2003).

Concepts for FDL Design

- FDL supports large-scale operations on collections

    theory translation, e.g. CZF to Type Theory

    cross linking via formal thesaurus

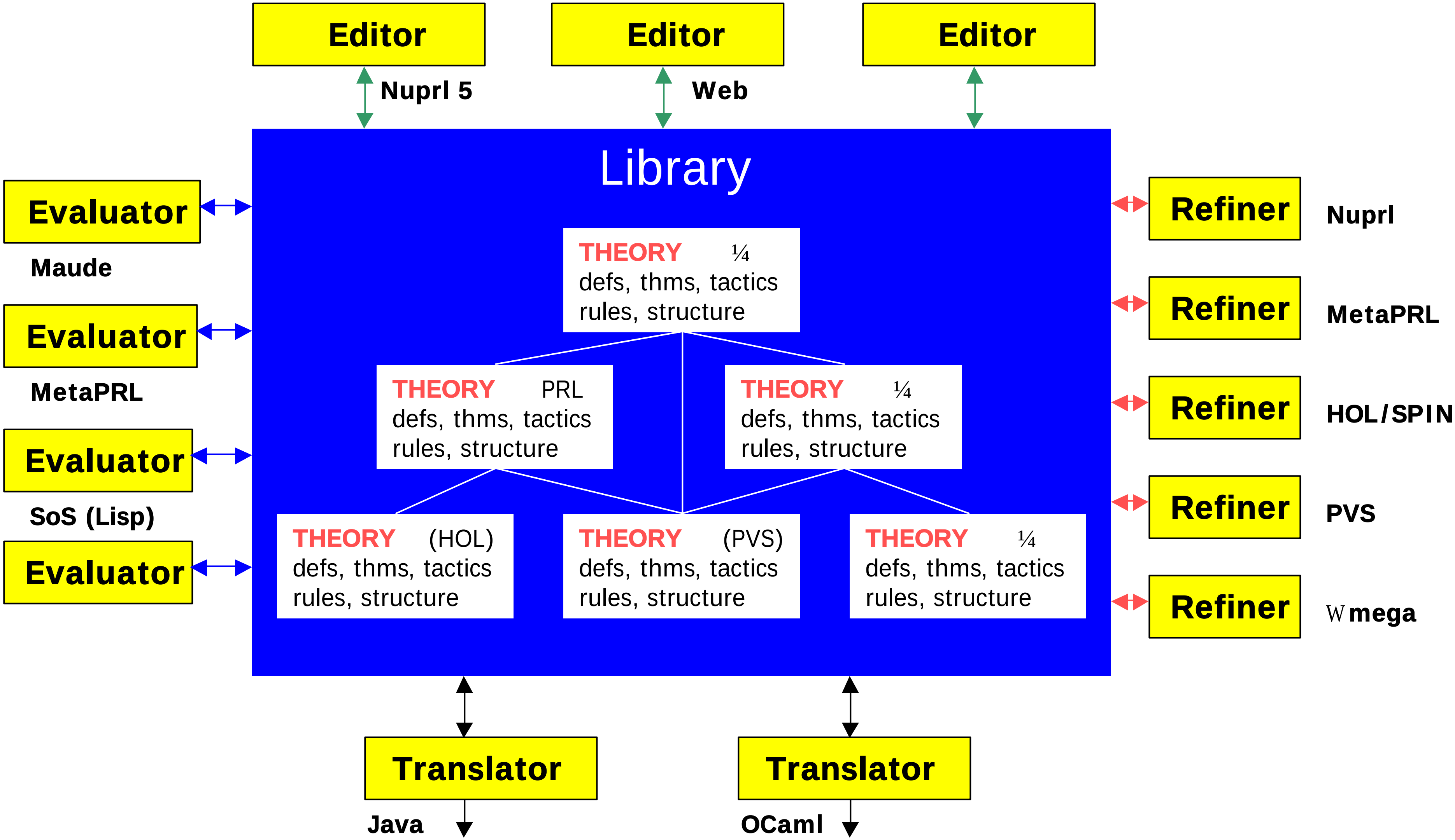
    transplanting theorems

    classical to constructive translations

Outline

- The ONR Digital Library Project
- Concepts for Formal Digital Library (FDL) design
- Current status of FDL
- Questions and issues

Formal Digital Library



Features of Prototype FDL (see Description and Ref Manual)

LISP/ML based system

- 6,000 named functions
- 62,000 lines of code
- 22,000 lines of comments

Some code adapted from LPE and Nuprl currently stores many Nuprl, PVS objects. Limited service will be available from the Web over the course of the year, e.g. accepting PVS files.

We have a customer – ORA.

Prototype FDL – Operations (Manual 3.2)

The basic operations are:

- bind id to object
- unbind id from object
- generate new object id
- lookup object
- activate an object
- deactivate object
- allow garbage collection
- disallow collection

Prototype FDL – Data (Manual 3.1)

Organized to eventually support closed maps

$$\boxed{D \text{ ® } Term(D)}$$

D are object names (abstract)

$Term(D)$ are objects with embedded references

Map is closed under object reference.

Working space is the current closed map.

Basic data structure is the library table.

Closed Maps

$$D \rightarrow Term(D)$$

closed under reference (no dangling pointers)

closed	**open**
$< id, term(\ ) >$	$< id, term(\ ) >$
$< id, term(\ ) >$	$< id, term(\ ) >$
$< id, term(\ ) >$	$< id, term(\ ) >$
$< id, term(\ ) >$	?
$< id, term(\ ) >$	

Prototype FDL – Transaction System (Manual 6.2)

Operations on closed maps can be elegantly implemented by transactions.

For example, deleting an object from map f requires deleting all objects that depend on it (no dangling pointers).

Delete is a database transaction – all or nothing, leaving a closed map.

Transaction management allows crash recovery.

Prototype FDL Content (Manual 7.2)

PVS libraries and refiner

- 20 libraries
- 400 theories
- 900 definitions
- 2,300 lemmas
- 300 theorems
- 200 postulates

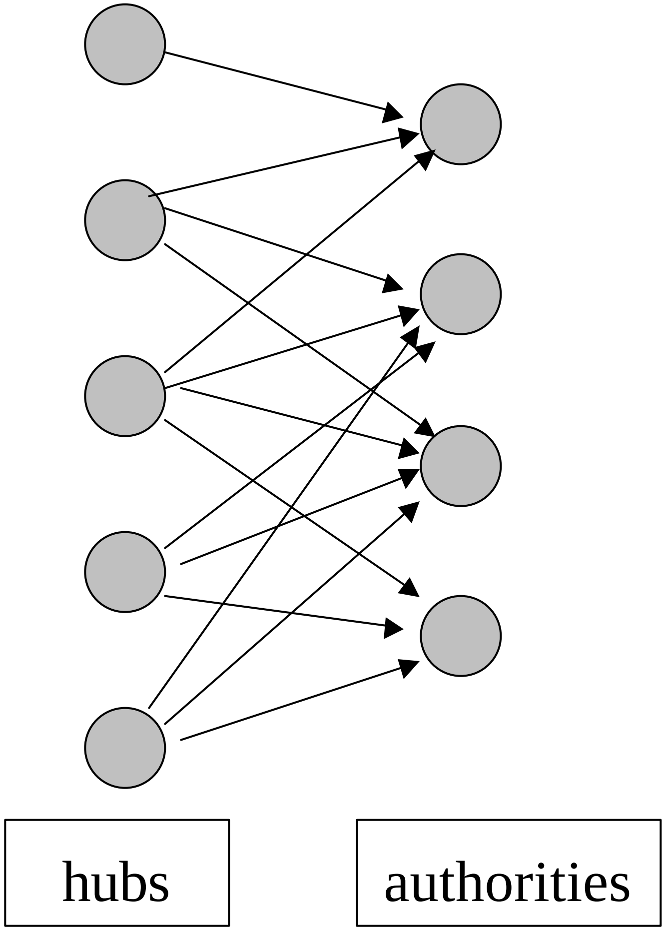
Outline

- The ONR Digital Library Project
- Concepts for Formal Digital Library (FDL) design
- Current status of FDL
- Questions and issues

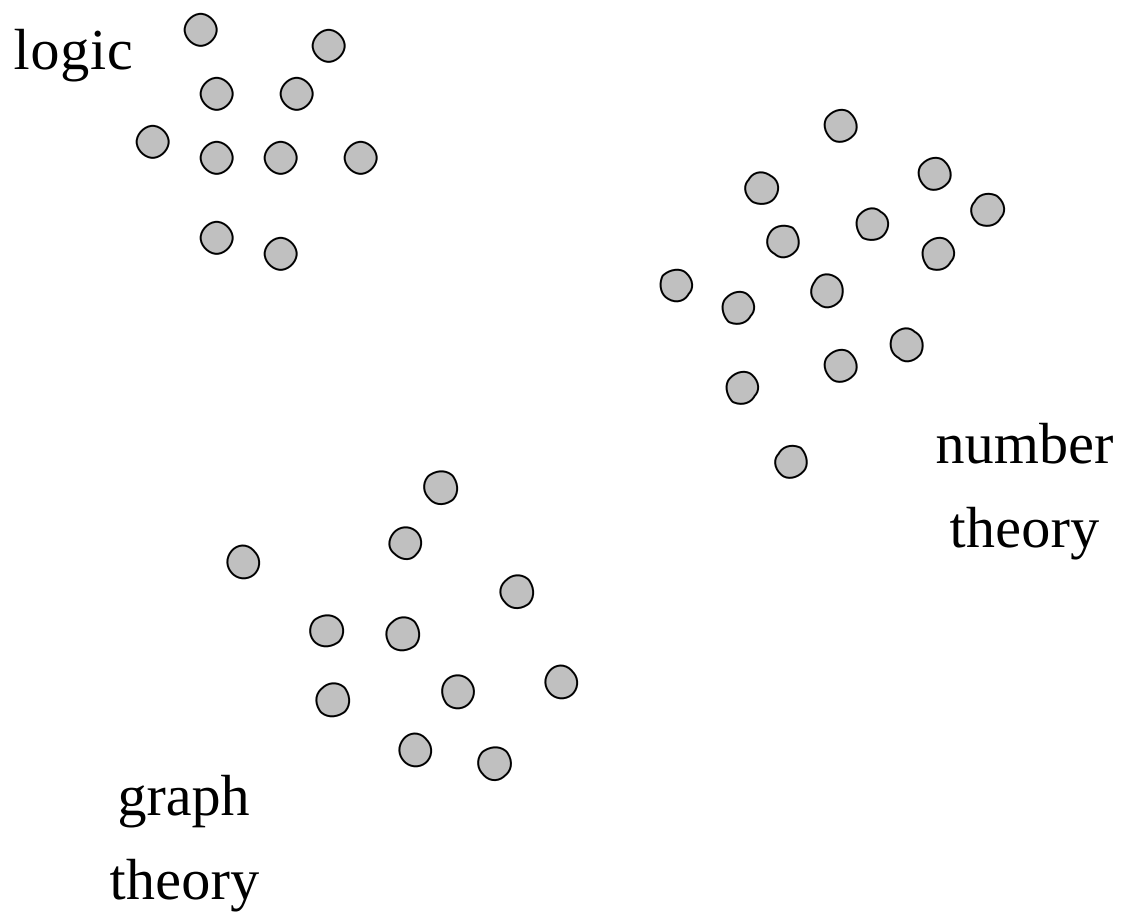
Questions and Issues

- What minimal set of services should an FDL provide?
- What community would be well served by an FDL?
- How can users contribute to an FDL?

Using Kleinberg's Hubs and Authorities



Classifying by Eigenvectors



Rehosting Strategy

- Import Larch theories and proofs as FDL terms
  - generic Yacc/Lex to FDL tool
  - C/C++ connection to FDL
- Build only the Larch prover's "refiner"
  - port from SSL to C++ using existing code
- Make display forms for Larch and Larch/VHDL
  - FDL provides editor attachments

FDL Capabilities – Formal Metamathematics

Deep sharing requires metamathematical results such as

Howe: Classical Nuprl is consistent with HOL

Smith: Nuprl domain theory is not consistent with HOL, PVS

Moran: Extended Classical Nuprl is consistent with HOL and PVS

Services

- Can we justify our data format as essential to a minimal set of services?
- How to search?
- How to justify proofs with code?

Technical Challenges: How to Increase the Value of Formal Material

- Increase access
    - for computing, math, science
    - for publication and dissemination
    - for information science studies
    - for education
- Account for trust
    - store evidence (proofs, dependencies)
    - third-party validation
    - certificates
- Track dependencies
    - logical dependence
    - relevance
- Insure stability of stored objects
    - replayability
    - stable proofs
    - promote stable code

References

Robert Constable	"Types in Logic, Mathematics and Programming," *Handbook of Proof Theory, Chapter X,* Sam Buss, ed., pp. 694-773, Elsevier, 1998.
Nuprl Home Page	*www.nuprl.org* – Math Libraries – ONR Project
Coq Home Page	*coq.inria.fr*
Mathweb Home Page	*www.mathweb.org* (Michael Kohlhase, CMU) – OM Doc – OpenMath – MBase
HELM	(Andrea Asperti, Bologna)
Math ML	(W3C Math Working Group)

References (cont.)

Stuart Allen	“Abstract Identifiers, Textual Reference and Record Keeping,” Technical Report, Computer Science Dept., Cornell University, October 2002.
Allen et. al	“FDL: A Prototype Formal Digital Library,” *www.nuprl.org/html/Digital_Libraries.html*