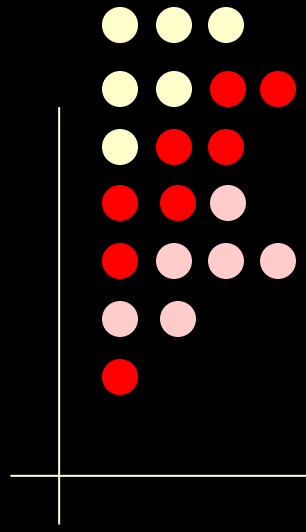
OFFICE OF NAVAL RESEARCH (ONR)

CORNELL UNIVERSITY DEPARTMENT OF COMPUTER SCIENCE

CORNELL

Design of an Interactive Digital Library of Formal Algorithmic Knowledge

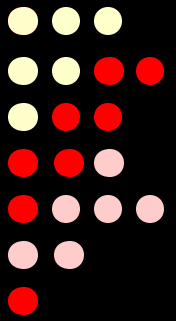


December 12, 2002 / Stanford University

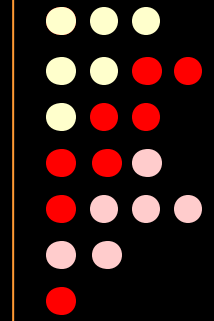
Stuart Allen
Mark Bickford
Robert Constable
Jason Hickey (Cal Tech)
Richard Eaton
Christoph Kreitz
Lori Lorigo
Jim Caldwell (Wyoming)

Outline of the Talk

- The ONR Digital Library Project
- Concepts for Designing the FDL
- Current Status of the FDL
- Questions and Issues

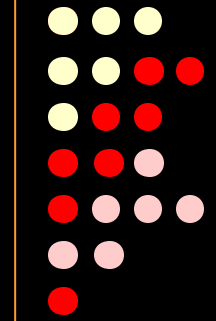


ONR Digital Library Project



“ … to create a digital library of algorithms and constructive mathematics useable for program and software construction.”

ONR Digital Library Project

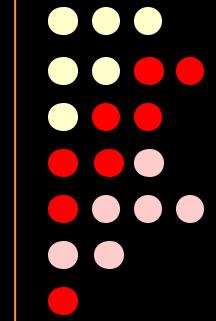


" … to create a digital library of algorithms and constructive mathematics useable for program and software construction."

Goals

- Semantics-based interactive infrastructure
- Create, collect and organize formal content
- Use the DL to build reliable software

ONR Digital Library Project

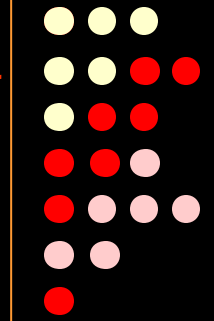


" … to create a digital library of algorithms and constructive mathematics useable for program and software construction."

Benefits

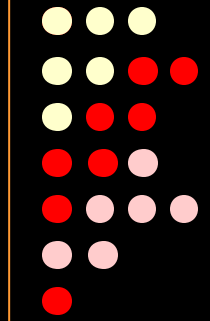
- Basis for highly reliable, responsive software
- Acceleration of discovery
- Wider access to formal content

Meaning of Formal: Background



"… having a precise meaning or objective criteria of correctness … ideally computer verifiable, based on syntactic form."

Meaning of Formal: Background



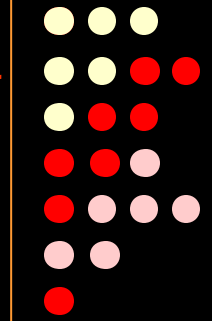
"… having a precise meaning or objective criteria of correctness … ideally computer verifiable, based on syntactic form."

i.e. root program: computes the integer square root of a natural number , i.e. root(n^2) = n, root(16) = 4

$$root(n) == \textbf{\textit{if}}\ \ n = 0\ \textbf{\textit{then}}\ 0$$
$$\textbf{\textit{else let}}\ r = root(n-1)\ in$$
$$\textbf{\textit{if}}\ \ (r+1)^2 > n\ \textbf{\textit{then}}\ r\ \textbf{\textit{else}}\ r+1\ \textbf{\textit{fi}}$$
$$\textbf{\textit{fi}}$$

We can prove program correctness using direct computation and algebra.

Meaning of Formal: Background



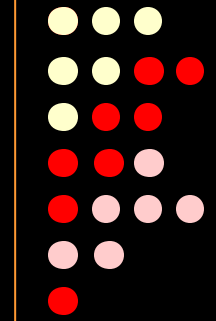
"… having a precise meaning or objective criteria of correctness … ideally computer verifiable, based on syntactic form."

i.e. halting problem using the bar type, A bar
n is in A bar means "if n halts then n is in A".

$\neg\exists h$:A bar$\rightarrow$ **Bool**. $\forall$n:A bar. *h(n) = true iff n halts*

We use type theory to form the expressions, and check them mechanically using axioms of the theory.

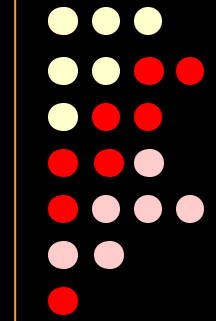
Strategy for Meeting ONR Goals



- Attract a community of contributors
  - MathWeb, PVS, ORA, MetaPRL, JProver, HOL, Isabelle, …
- Account for correctness in a multi-logic, multi-prover environment
  - Insolvability result is inconsistent with

    $\forall p: p \vee \neg p$ (classical logic)
- Provide semantics-based library services

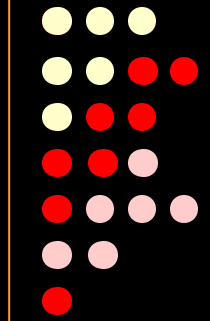
Challenges and Problems



- Relatively small and disconnected community
- Formal proving is still hard work
  - Expansion factor
  - Shallow base of mathematical facts
  - Demanding skill set

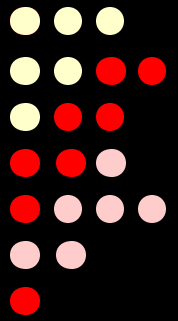
Outline of the Talk

- The ONR Digital Library Project
- Concepts for Designing the FDL
- Current Status of the FDL
- Questions and Issues

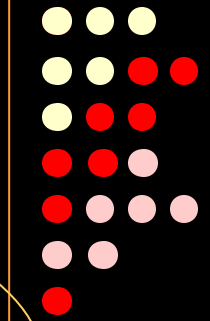


Basis for FDL Design

- The FDL contains formal objects
- These objects have interdependencies and informative links forming a directed graph
- The FDL is interactive
  - search, retrieval, archival and creation and submission of new content

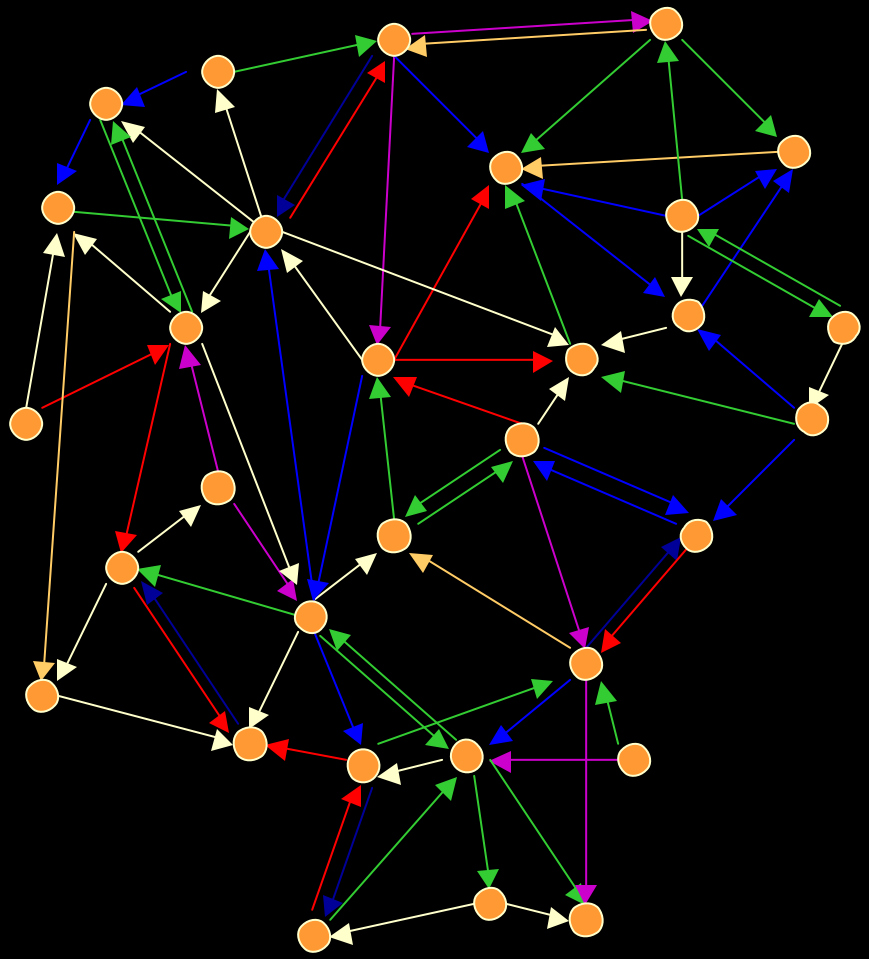


Objects



FORMAL	rules	definitions	algorithms, code
	conjectures	theorems	specifications
	inferences	certificates	proofs, partial proofs
INFORMAL	displays	articles	documentation

Information graph of the FDL



objects

logical dependency

textual links

accounting links

meta-logical links

An Example

proof

article

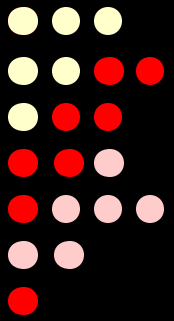
```
* top
∀A:𝕌. ∀l:A List.  ((¬(l = [])) ⇒ (||l|| ≥ 1 ))

BY aux_auto(RepeatMFor 2 (D 0) THENM D 2)

* 1

1. A : 𝕌
⊢ (¬([] = [])) ⇒ (||[]|| ≥ 1 )

BY auto(D 0 THENM D (-1))

* 2

1. A : 𝕌
2. u : A
3. v : A List
⊢ (¬([u / v] = [])) ⇒ (||[u / v]|| ≥ 1 )

BY AbReduce 0 THEN aux_auto(D 0)

* 2 1

4. ¬([u / v] = [])
⊢ (||v|| + 1) ≥ 1

BY InstLemma `non_neg_length` [⌜A⌝;⌜v⌝] THEN Auto
```

certificates

theorem

∀A:𝕌. ∀l:A List. (||l|| ≥ 0)

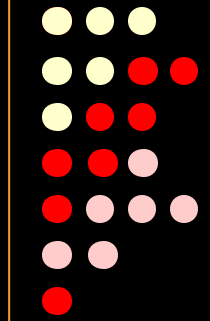
More FDL Design Concepts



- FDL performs archival functions
- *FDL combines theories using meta-logic (in-progress)*
- *FDL performs large-scale translation operations (not yet implemented)*

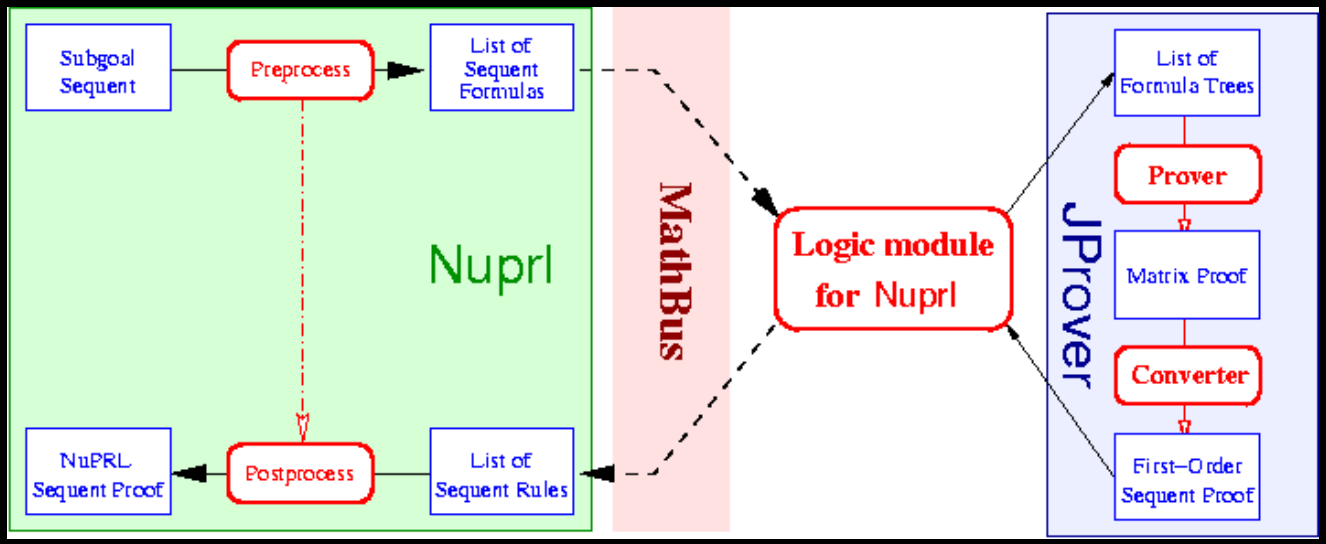
Archival Example

- 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)



Meta-logic : JProver Integration

- First Order Logic stand-alone prover
- Multi-prover environment, “hybrid” proofs

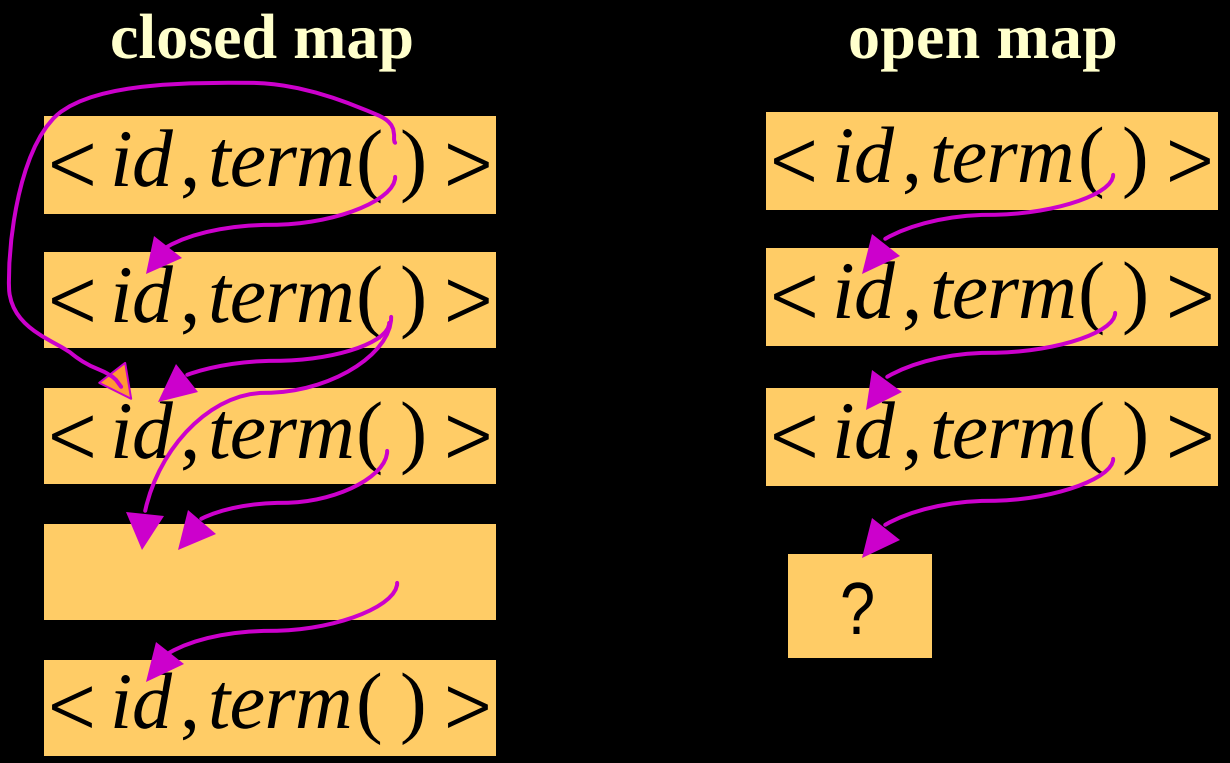


Nuprl Type Theory combined with JProver Logic

Translation operations via “maps”

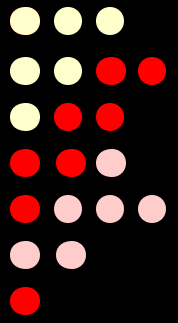
Working map: $D \rightarrow Term(D)$

closed under reference (no dangling pointers)

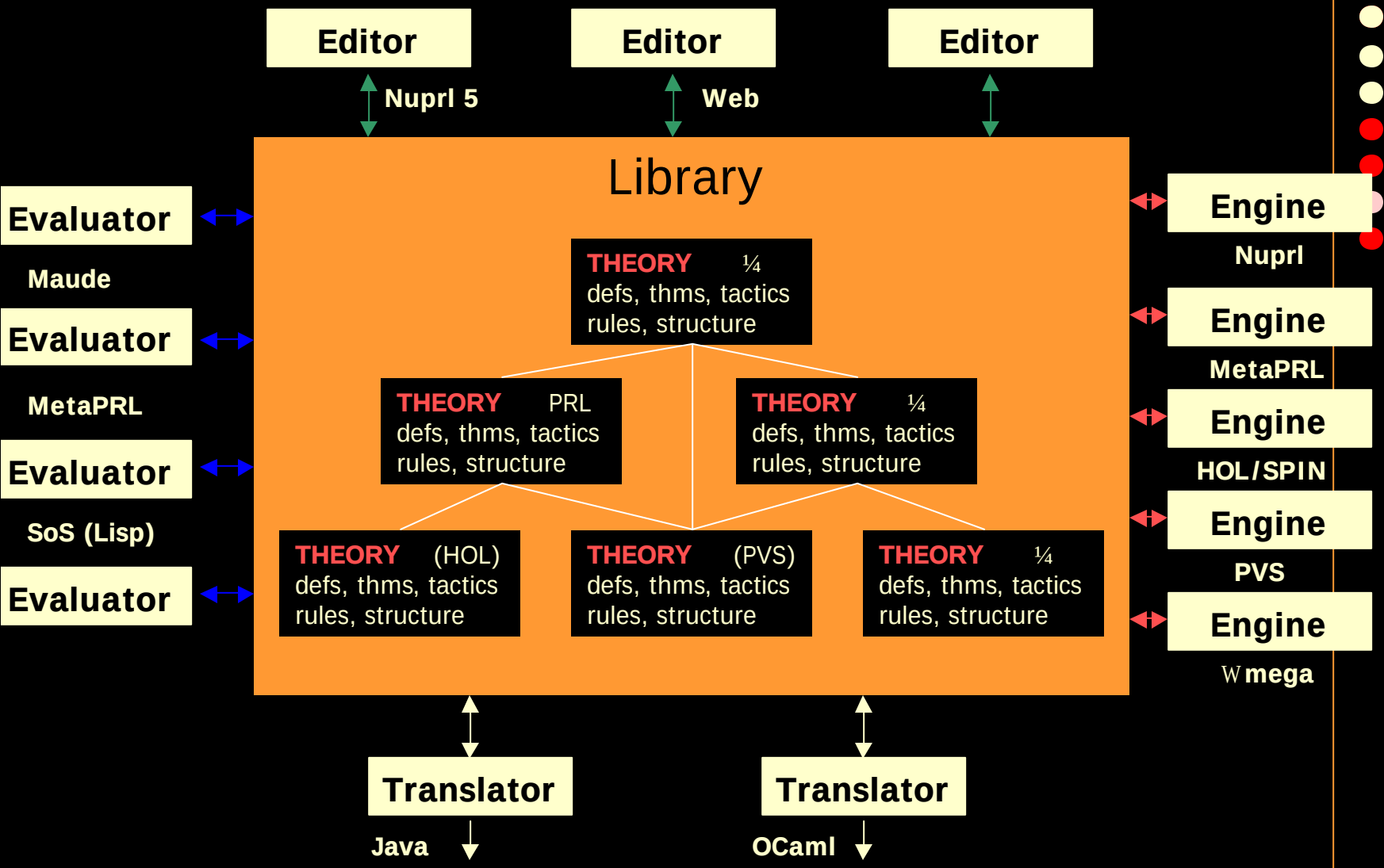


Outline of the Talk

- The ONR Digital Library Project
- Concepts for Designing the FDL
- Current Status of the FDL
- Questions and Issues

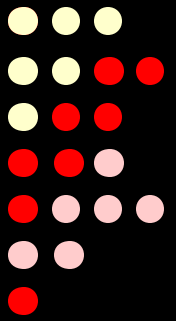


System Architecture



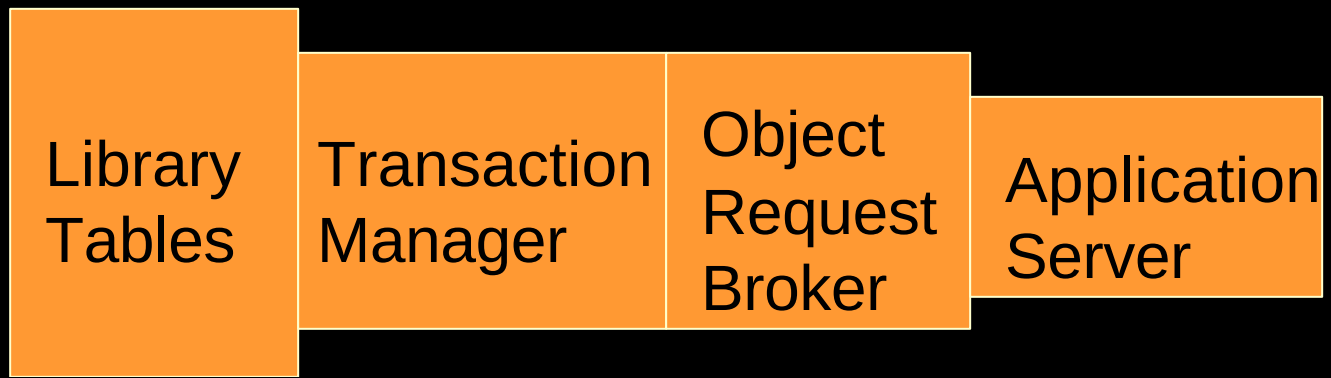
Library Contents

- 14 Nuprl theories (collections) + 408 PVS modules
- 796 Nuprl definitions + 803 PVS constants
- 2,764 Nuprl theorems + 4587 PVE lemmas
- 2,764 Nuprl proofs + 1103 PVS proofs

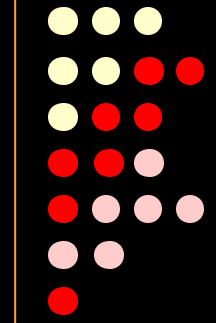


Library Implementation

- 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, and MetaPRL objects.

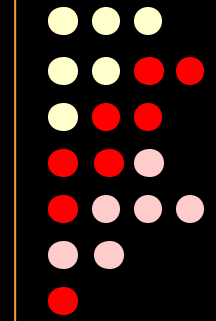


Basic FDL Operations



- lookup object
- create an object (empty)
- bind content to an object (save)
- unbind id from object
- activate object
- deactivate object
- allow garbage collection
- disallow garbage collection

Object Storage and Granularity



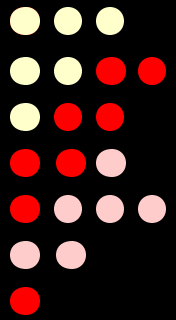
- Proof is a DAG of inferences
- Information links are stored as properties
- Names are stored as properties
- Transaction based model - objects are not deleted or stored only in memory, you can recover from crash or changes

Data Format: Term

$$Term = op \times Term\ List$$

$t = op(t; \mathrm{L}\ ; t)$ for t a term

- XML
- OMDoc
- MathBus
- Ascii, c/c++ interface



Data Format: Term

$$Term = op \times Term\ List$$

$t = op(t; \mathrm{L}\ ; t)$ for t a term

- XML
- OMDoc
- MathBus
- Ascii, c/c++ interface

Term may have bindings:

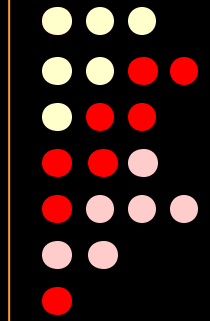
$op(\bar{v}_1.t_1; \mathrm{L}\ ; \bar{v}_n.t_n)$ $\bar{v}_i$ list of binding variables

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

i can be reference objects or values

Object Contributors

- MetaPRL import via MathBus
- XSLT is used to import formal objects in OMDoc (XML) format to the FDL
- OMDoc supported in ActiveMath, OMEGA, LambdaClam, INKA, TPS, PVS, Nuprl, more
- Import PVS directly using LISP data structures
- Import Larch from ORA using yacc/lex tool

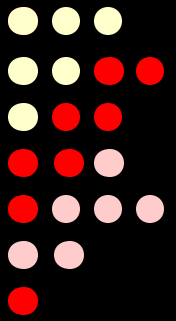


Outline of the Talk

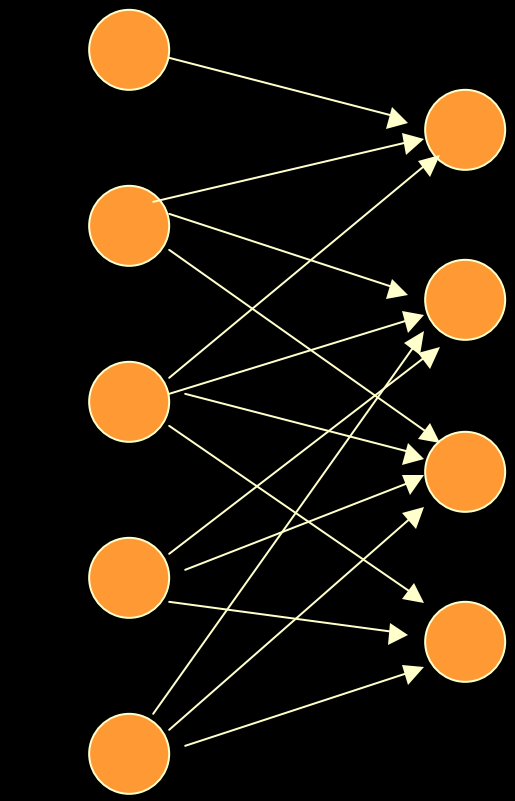
- The ONR Digital Library Project
- Concepts for Designing the FDL
- Current Status of the FDL
- Questions and Related Work

Questions

- What community to target?
- How to search?
- Data format justification?
- Can we learn from semantic knowledge?



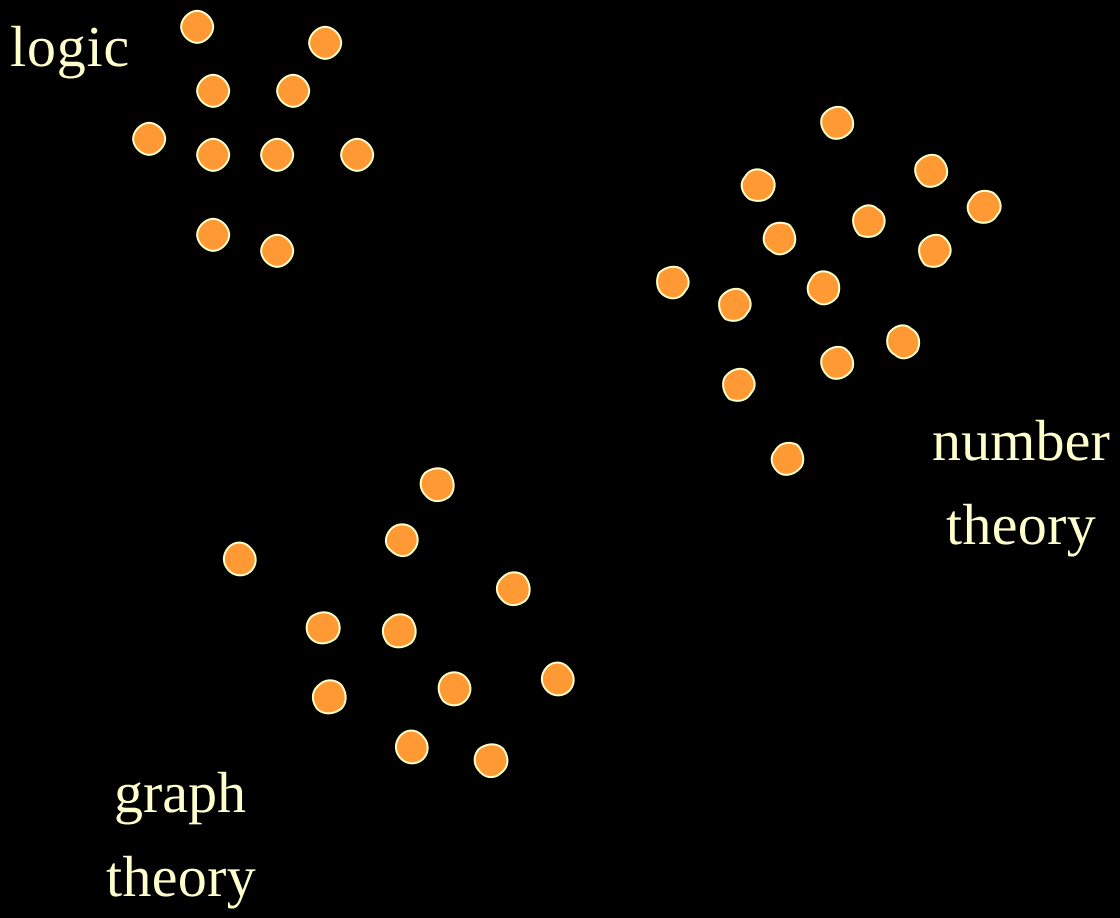
Hubs and Authorities, Kleinberg

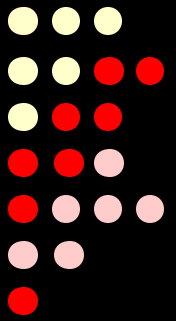


hubs

authorities

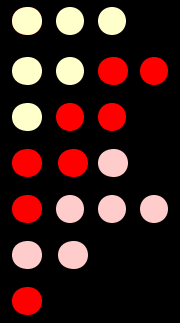
Classifying by Eigenvectors





Related Work

- HELM http://helm.cs.unibo.it/
- MathWeb http://www.mathweb.org
- Mathematical Knowledge Management (MKM) http://imps.mcmaster.ca/na-mkm-2002/presentation



FDL Manual

FDL Manual

http://www.nuprl.org/html/Digital_Libraries.html

Thank you!

lolorigo@cs.cornell.edu