Foundations for the Management of Formal Mathematical Knowledge

Robert L. Constable

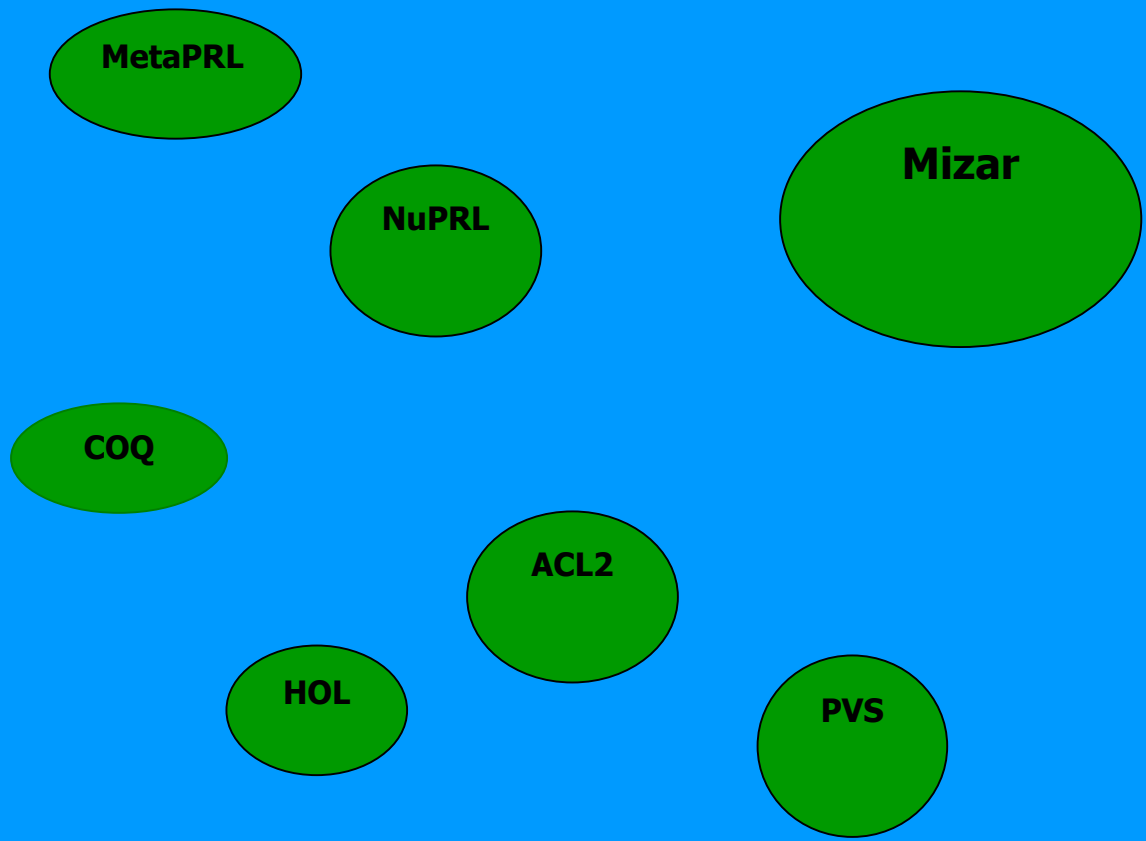
Cornell University

Small Types Workshop

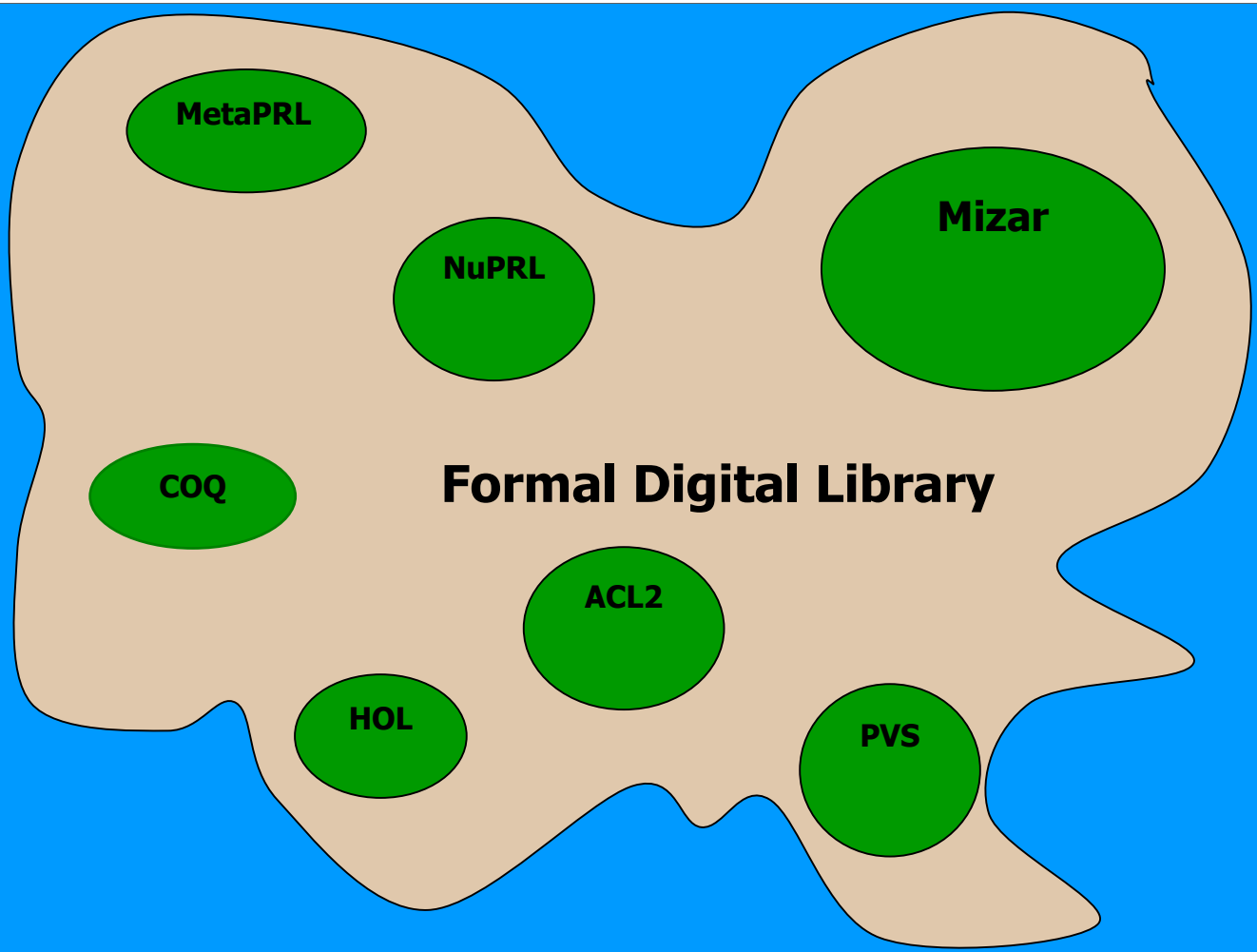
Nijmegen, November 2004

Vision

Very large libraries (databases) of formal mathematical knowledge, created and shared by the theorem proving community, will make interactive theorem provers indispensible tools in parts of mathematics, science, and education as they already have become in formal methods and software practice. In turn, the libraries will grow more rapidly and become interconnected with the intuitive mathematical literature, weaving threads of computational certainty into the oldest rigorous intellectual activity of human kind.



Theorem proving systems as isolated islands, reaching a small community



The FDL joins all the small islands into a single big island that can accommodate much more than the sum of all individual islands, and allows people to stand on the ground in between.

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

1. Relating theories and systems
2. The structure of theories
3. Aids to organization and search

Type Theories

There are at least eleven active important interactive provers, some coupled to fully automatic provers such as JProver, Otter, TPS, and Omega.

ACL2, Alf, Coq, HOL, Isabelle, MetaPRL, Minlog, Mizar, Nuprl, PVS, Twelf

Nine of these are based on higher-order typed logic. Why?

Will there ever be ONE system? ONE logic? ONE framework? ONE metalogic?

Will there ever be one Operating System? One Programming Language?

Central Characters – Higher-Order Logics and Type Theories

HOL/Isabel	Nuprl
$\mathbb{N}, \mathbb{B}$	$\mathbb{Z}$, *Atom*
$A \to B$	$x : A \to B$
$A \times B$	$x : A \times B$
$a = b$ *in* A	$a = b$ *in* A
$\forall x : A,\ \exists x : A$	$\forall x : A,\ \exists x : A$

Coq and Nuprl share many features, so Coq will be mentioned as well.

Comparing Representative Type Theories

Least Number Principle **(LNP):**

$$\forall P : N \to \mathbb{B}.\ \left(\exists y : \mathbb{N}.\ P(y) \Rightarrow \exists x : \mathbb{N}.\ \left(P(x)\ \&\ \forall x : \mathbb{N}.\ z < x \Rightarrow \neg P(z)\right)\right)$$

Law of Excluded Middle **(LEM):**

$$\forall P : \mathrm{Prop}_i.\ \left(P \vee \neg P\right)$$

Interpretations of LNP

In Nuprl LNP specifies a class of programs that includes:

```
least (P, y) =
    for i = 0 to y do
        if P (i) then return (i)
        else i := i + 1
    end; return (y).
```

In HOL and PVS the function $P \in \mathbb{N} \to \mathbb{B}$ is ANY function, and it represents any propositional function, including noncomputable ones, e.g.:

P(x,m) iff a Turing machine with *x* states can print the number *m* and halt starting with blank tape.

Interpretations of LEM

In Nuprl, LEM is not an axiom nor a theorem. There are models of Nuprl for which $\neg LEM$, and LEM is false in the domain theory of Nuprl.

In HOL and PVS, LEM is an axiom.

Adding LEM to Nuprl produces a theory we call Classical Nuprl. Doug Howe proved that this theory is consistent. We will examine that proof.

Type Theory as a Specification Language

Integer square root example

- Thm 1.2 $\forall n : \mathbb{N}.\ \exists! r : \mathbb{N}.\ \mathit{Root}(n, r)$
- Cor 1.1 $\exists rt : \mathbb{N} \to \mathbb{N}.\ \forall n : \mathbb{N}.\ \mathit{Root}(n, rt(n))$
- Cor 1.2 $\mathit{ext}(\mathit{Thm}1.1)\ \varepsilon\ \mathbb{N} \to \mathbb{N}$
- Cor 1.3 $\forall n : \mathbb{N}.\ \mathit{Root}(n, \mathit{ext}(\mathit{Thm}1.1)(n))$

Interpretations of Integer Square Root

These theorems could be true in Coq, HOL, Nuprl, and PVS if HOL and PVS supported extraction of functions from $\forall\exists$ statements according to the Semantics of Evidence; see my MOD 1982 lecture notes, *Assigning Meaning to Proofs;* and my *Handbook of Proof Theory* article [39], "Types in Logic, Mathematics and Programming."

In Coq and Nuprl, the function in Cor 1.1 is computable.

The Root Program Extract

The Working Material, Appendix A, p. 46, shows the extract term for this proof in ML notation:

let *rec* *sqrt* i =

if $i = 0$ then $< 0, pf_0 >$

else let $< r, pf_{i-1} > = sqrt(i - 1)$

in if $(r+1)^2 \leq n$ then $< r+1, pf_i >$

else $< r, pf_i' >$

Relating theories and systems

(early period)

CLASSICAL	CONSTRUCTIVE
Simple type theory – STT (HOL, Isabelle)	Intuitionistic type theory – ITT (Alf)
Extended STT – STT^+ (PVS)	Computational type theory – CTT (Nuprl, MetaPRL)
	Calculus of Inductive Constructions – CIC (Coq, MetaPRL)

Relating theories and systems

(current period)

CLASSICAL

Simple type theory – STT
(HOL, Isabelle)

Extended STT – STT^+
(PVS)

Classical – CTT (Howe 1997)
- Polymorphic
- Computational
- includes sets
- (Classical – Nuprl)

CONSTRUCTIVE

Constructive Simple type theory – CSTT
(Berghofer/Nipkow 2002)

Intuitionistic type theory – ITT

Computational type theory – CTT

Calculus of Inductive Constructions – ClC

Semantics of STT

Andy Pitts gives a clean semantics of STT in ZF set theory.

In that semantics:

$$\mathbb{N} = \{\phi, \{\phi\}, \{\phi, \{\phi\}\}, \ldots\}$$

$$\lambda x : N.b(x) = \{\langle x, b(x)\rangle \mid x \in \mathbb{N}\}$$

$\mathbb{N} \to \mathbb{N}$ is the set of functional relations on $\mathbb{N} \times \mathbb{N}$.

Semantics of CTT

Per Martin-Lof provided an intuitive axiomatic semantics based on his doctrine of canonical forms.

Stuart Allen gave a PER (partial equivalence relation) semantics in a neutral theory of inductive definitions.

In that semantics:

$\mathbb{Z} =$ all terms that evaluate to canonical integers

$\{n\}_{\mathbb{Z}} = \{x : \text{Term} \mid x \downarrow n\}$

$\mathbb{Z}$ is $\bigcup_{z \in \mathbb{Z}} \{z\}_z$

Semantics of CTT

$\lambda x.b(x)$ is defined by evaluation rules

think of $\{3\} \to \{f3\}$

$\mathbb{Z} \to \mathbb{Z}$ is $\bigcup_{f \in \mathbb{Z} \to \mathbb{Z}} \bigcap_{z:\mathbb{Z}} (\{z\} \to \{fz\})$

$\lambda x.x$ is in all $A \to A$!

Subtyping in CTT versus Set Theory

Notice Even $\sqsubseteq$ $\mathbb{N}$

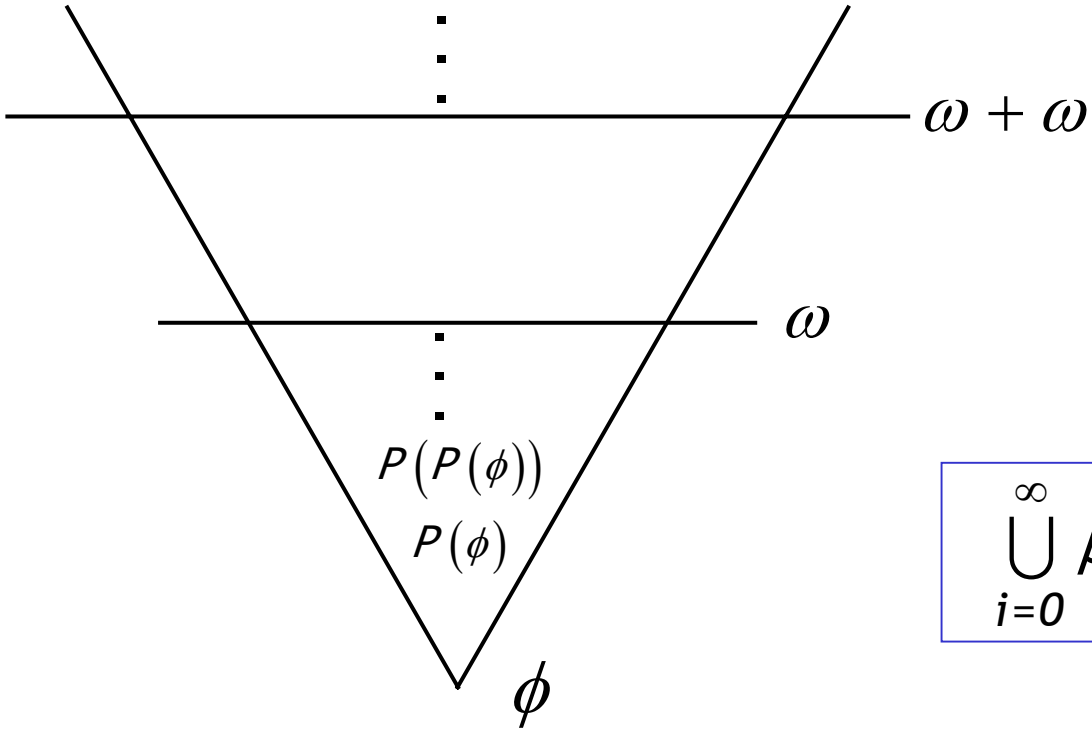
In CTT, $\mathbb{N} \to \mathbb{B}$ $\sqsubseteq$ Even $\to \mathbb{B}$

consider λx.rm(x,2)

In Set Theory, any f in Even $\to$ $\mathbb{B}$

is a subset of a function in $\mathbb{N} \to \mathbb{B}$

The Cumulative Hierarchy of Sets



$$\bigcup_{i=0}^{\infty} P^i_{(\phi)}$$

Captures the intuition of collecting stages and co-finality… unending stages.

Zermelo Fraenkel Set Theory (ZF)

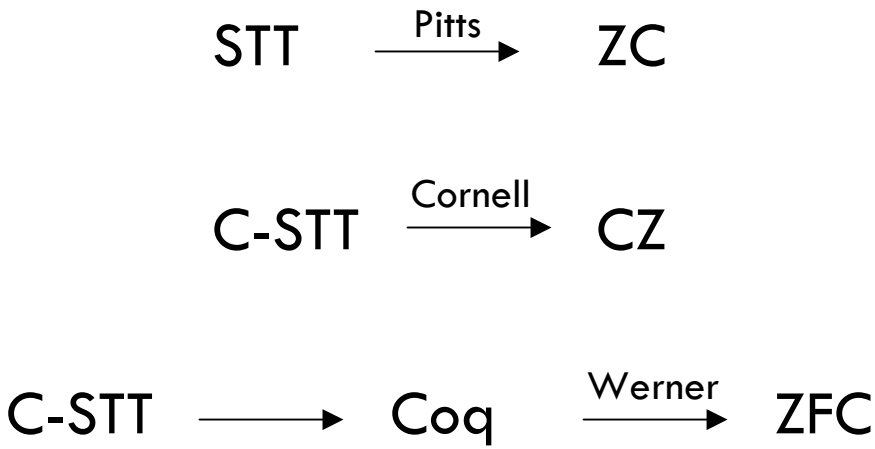
ZF can be axiomatized in 1st order logic by axioms, 2 axiom schemas (separation, replacement). Axiom of choice is a 9th axiom.

The hierarchy is defined as:

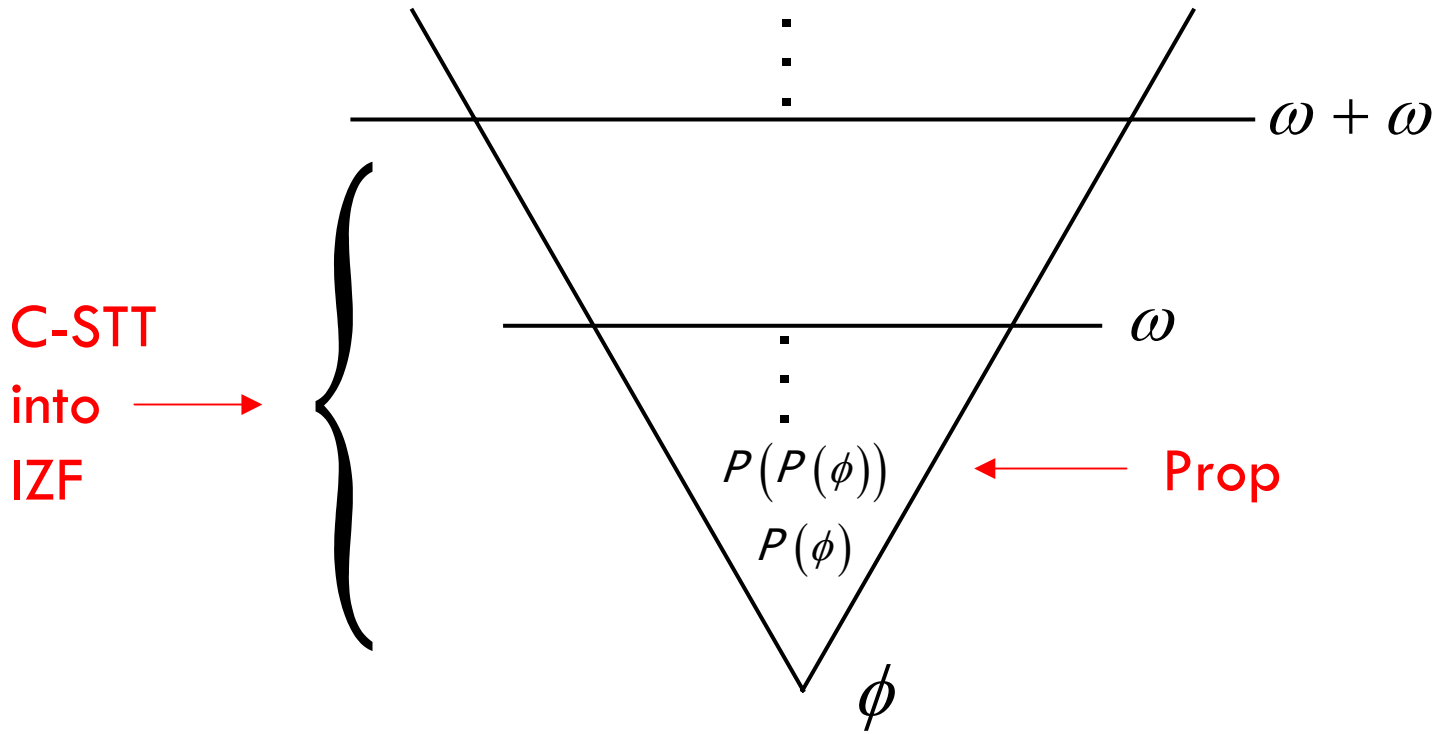
$$Z_0 = \phi$$
$$Z_\alpha = \bigcup_{\beta<\alpha} Z_\beta$$
$$Z_{\alpha+1} = P(Z_\alpha)$$
$$Z = \bigcup_{\alpha \in Ord} Z_\alpha$$

Every ZF set X is in some Z_α; its rank is the least such ordinal α.

Semantics for HOL

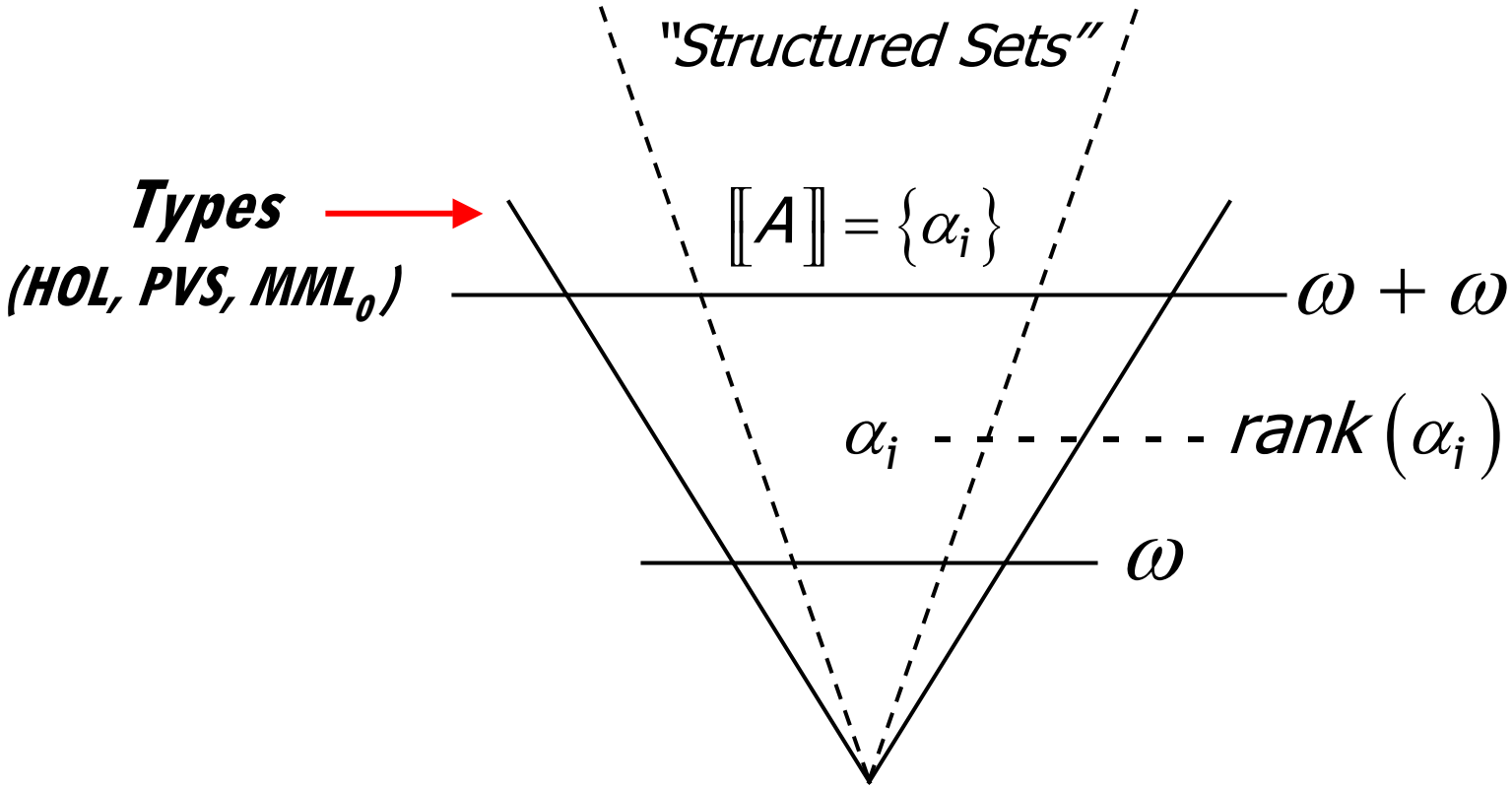


An Intuitionistic Cumulative Hierarchy of Sets



First Order Logic (FOL) Base with proof terms (proofs-as-programs.)

Review of Set Theoretic Semantics



The Cumulative Hierarchy of Sets

Goal of Howe's Approach

Pure Type Theory

$$\boxed{a_1, a_2, \ldots, a_i \ldots} \quad f(a)$$

$$f \Downarrow \lambda x.b \quad p \Downarrow < a, b >$$

$$\lambda x.b \quad < a, b > \quad c_i(\overline{a}) \quad \mathbb{U}_i$$

Type Theory with Set Terms

$$\boxed{a_1, a_2, \ldots, a_i \ldots} \quad \boxed{\alpha_1, \alpha_2, \ldots, \alpha_i \ldots}$$

$$A \quad \hat{\gamma}_A \quad f(a) \quad \hat{\varphi}(\hat{\alpha})$$

$$f \Downarrow \lambda x.b \quad f \Downarrow \hat{\varphi} \quad c_i(\overline{\alpha}) \Downarrow c_i(\overline{\alpha})$$

$$\lambda x.b \quad a \mapsto b \quad < a, b > \quad < \hat{\alpha}, \hat{\beta} > \quad c_i(\overline{a}) \quad c_i(\hat{\alpha})$$

Howe's Meaning Functions

Howe's Approach to Set-theoretic Semantics:

like Pitts, Dybjer, Troelstra, and others, Doug Howe maps types into sets; for A, a type, $[\![A]\!]$ is a set.

However, he introduces new ideas to handle functionality, polymorphism, subtyping, quotient types, and types as objects (universes). For instance,

$[\![\lambda X.X]\!]_{A\to A}$ is an element φ of $[\![A \to A]\!]$

and he writes that $\varphi \lhd \lambda X.X.$

Limitations of Set Embedding

Unlike with PVS and HOL, no embedding can cover all of Nuprl because

- Recursive types allow $T = \left(T \rightarrow \mathbb{B}\right) \rightarrow \mathbb{B}$
- Domains all absolutely unsolvable problems

But Howe gets a very substantial part and Moran extends to more.

Goal of Howe's Approach

- Extend the family of "structured sets"
- Create constants for all "structured sets"
- Extend evaluation to all terms
- Define approximation (covering)

$$\alpha_i \lhd a_i$$

to relate sets α_i to terms a_i

Howe's Method

1. Encode type terms and types into sets in the cumulative hierarchy W

$< ty, \gamma >$	γ
$< fn, \varphi >$	φ
$< c_i < \alpha_1, \ldots, \alpha_n >>$	$c_i\left(\overline{\alpha}\right)$

2. Define a subset of W with a unique meaning property; call it V.

3. Define a term model for all V sets; call it T_0.

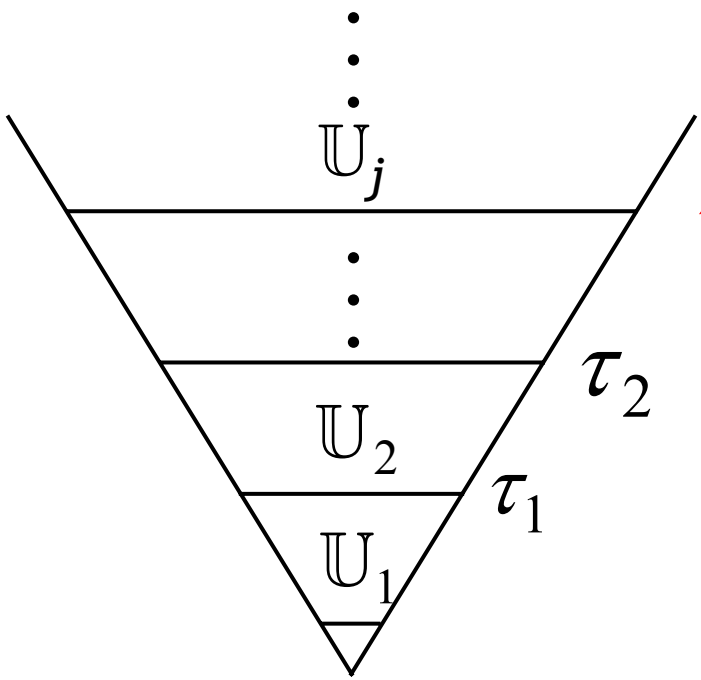
4. Define evaluation and approximation on T_0; this provides a computational type theory with set oracles.

I present a variant used by Evan Moran to extend Howe's results.

Extending Set Theoretic Semantics to ML-82, Alf, Nuprl

Can we account for these ideas in set theory?

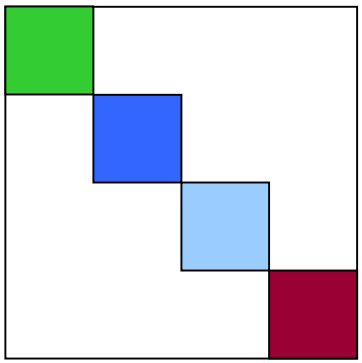
Can we embed CTT into the cumulative hierarchy?



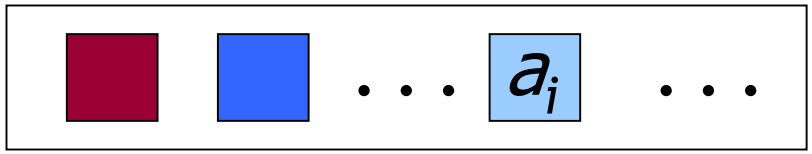
τ_i are inaccessible cardinals

Howe's Semantics

We think of types as collections of equivalence classes of terms:



or



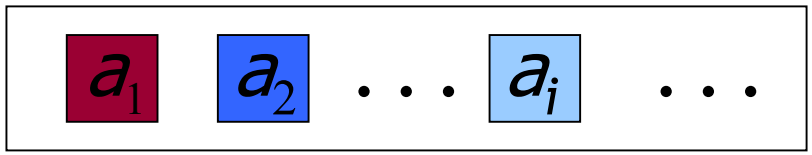
We need to allow sets representing equivalence classes α_i into the collection of types.

$$\boxed{\alpha_1, \alpha_2, \ldots, \alpha_n} \quad \text{written } \gamma\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$$

where α_i covers a_i $\left(\alpha_i \text{ approximates } a_i\right)$ $\alpha_i \lhd a_i$

Disjointness and Incompatibility

The equivalence classes are disjoint, of course.

$A =$ 

if $a_i \cap a_j \neq 0$ then

$a_i = a_j$

The set elements must also be incompatible:

$$\gamma_A = \{\alpha_1, \alpha_2, \ldots, \alpha_i\}$$

$con(\alpha_i, \alpha_j)$ implies $\alpha_i = \alpha_j$

For each $a_i \in A$ there will be a unique $\alpha_i \in \gamma_A$, such that $\alpha_i \lhd a_j$.

Disjointness and Incompatibility

The key condition for functions is:

$$\varphi \quad \boxed{<\alpha_1,\beta_1>,<\alpha_2,\beta_2>,\ldots,<\alpha_i,\beta_i>,\ldots}$$

$$\varphi' \quad \boxed{<\alpha_1',\beta_1'>,<\alpha_2',\beta_2'>,\ldots,<\alpha_i',\beta_i'>,\ldots}$$

If for all $<\alpha,\beta>,<\alpha',\beta'>$

$con(\alpha_i,\alpha_i')$ implies $con(\beta_i,\beta_i')$

then

$con(\varphi,\varphi')$

Compatibility (Consistency) Examples

- $0 \mapsto 0$ and $1 \mapsto 1$ are compatible (consistent) since $\varphi\{<0,0>\}$ and $\varphi\{<1,1>\}$ have a non-empty intersection, $\varphi\{<0,0>,<1,1>\}$.

- $0 \mapsto (0 \mapsto 0)$ and $0 \mapsto (1 \mapsto 1)$ are compatible, i.e. $\varphi\{<0,\varphi\{<0,0>\}>\}$ *con* $\varphi\{<0,\varphi\{<1,1>\}>\}$ since 0 *con* 0 and $\varphi\{<0,0>\}$ *con* $\varphi\{<1,1>\}$.

Membership and Approximation

If the type A is interpreted as the set γ_A, then for each $a \in A$, there is a unique α such that

$$\alpha \lhd a.$$

We will say $[\![A]\!] = \gamma_A$ and $[\![a]\!]_A = \alpha$.

Uniqueness Theorem

For any term t in T_0,

1. If $\gamma_1 \lhd t$ and $\gamma_2 \lhd t$, then $\gamma_1 = \gamma_2$
2. For all $\gamma \in V$ and $\alpha_1, \alpha_2 \in \gamma$,

   if $\alpha_1 \lhd t$ and $\alpha_2 \lhd t$, then $\alpha_1 = \alpha_2$

The Term Language T_0

Howe defines a term language, T_0, that includes the Nuprl terms with binding, such as

$$x : A \to B,\ \ x : A \times B,\ \ \lambda x.b,\ \ x : A \cap B,$$
$$less\left(n; m; a; b\right),\ \ decide\left(p; x.a; y.b\right), \ldots$$

T_0 includes the constructors and constants, such as

$$\mathbb{U}_i,\ nat\left\{n\right\},\ pair\left(a; b\right),\ inl\left(a\right),\ inr\left(b\right),\ ap\left(f; a\right),\ \ldots$$

T_0 includes constants for all of the $\gamma, \varphi, \xi, c_i\left(\overline{\alpha}\right)$ sets. Howe uses $\hat{\gamma}, \hat{\varphi}, \hat{\xi}$ to denote the set terms.

The Term Language T_0

Howe extends the evaluation relation to all terms, for example:

$$ap(\lambda x.b; a) \Downarrow c \quad \text{if } b[a/x] \Downarrow c$$
$$less(0; 1; a; b) \Downarrow c \quad \text{if } a \Downarrow c$$

For instance, $ap(\lambda x.x; \hat{\varphi}) \Downarrow \hat{\varphi}$.

Evaluation Rules

$$\frac{f \Downarrow \hat{\phi} \quad (\alpha, \beta) \in \phi \quad \alpha \lhd a}{f(a) \Downarrow \hat{\beta}} (ap_{\phi}) \qquad \frac{f \Downarrow \lambda x.b \quad b[a/x] \Downarrow v}{f(a) \Downarrow v} (ap_{\lambda})$$

$$\frac{}{\hat{\gamma} \Downarrow \hat{\gamma}} \qquad \frac{}{\lambda x.b \Downarrow \lambda x.b} \qquad \frac{}{c_i(\overline{a}) \Downarrow c_i(\overline{a})}$$

Approximation Rules

$$\frac{e \Downarrow v \quad \alpha \lhd v}{\alpha \lhd e} \qquad \frac{\forall j \quad \alpha_j \lhd a_j}{c_i\left(\hat{\alpha}_1, \ldots, \hat{\alpha}_n\right) \lhd c_i\left(\hat{a}_1, \ldots, \hat{a}_n\right)}$$

$$\frac{\forall (\alpha, \beta) \in \phi \quad \beta \lhd b\left[\hat{\alpha} / x\right]}{\phi \lhd \lambda x.b}\ \dagger$$

† Howe does not use this rule because it is unproven when the non-determinism of functions on quotient types is allowed.

Examples

Let $\varphi = \{<0,4>,<1,5>\}$ $\varphi' = \{<0,2>\}$

$\psi = \{<\varphi,17>,<\varphi',18>\}$

$\widehat{\varphi}(0+0) \Downarrow 4$ because $0 \lhd 0+0$

$\widehat{\varphi} \lhd \lambda x.x+4$ but not $\widehat{\varphi}' \lhd \lambda x.x+4$

$\widehat{\psi}(\lambda x.x+4) \Downarrow 17$

The Substitution Lemma

If $\alpha \lhd a$ then for any term e,

$\beta \lhd e[\alpha / x]$ implies $\beta \lhd e[a / x]$

The intuition is that for any demonstration of $\beta \lhd e[\alpha / x]$, if it does not examine α, then it also establishes $\beta \lhd e[a / x]$.

If the derivation depends on α, then since $\alpha \lhd a$ and a is a term, there is more information in a than in any approximation α, and that information will establish the facts used about α to show $\beta \lhd e[\alpha / x]$ using a in place of α.

Rationale for the Substitution Lemma

Assume $\alpha \lhd a$ and $\beta \lhd e\left[\hat{\alpha} / x\right]$

To say $\beta \lhd e\left[\hat{\alpha} / x\right]$ means $\alpha \mapsto \beta \lhd \lambda x.e$.

Thus, $\lambda x.e \in \gamma\{\alpha \mapsto \beta\}$, a singleton set, and

(1) $\lambda x.e \in \gamma\{\alpha\} \to \gamma\{\beta\}$

(2) From $\alpha \lhd a$ we know $a \in \gamma\{\alpha\}$

From (1) and (2) we have $(\lambda x.e)a \in \gamma\{\beta\}$,

thus, $e\left[a / x\right] \in \gamma\{\beta\}$, and hence

$$\beta \lhd e\left[a / x\right].$$

Soundness – the Meaning of $x : A \to B$

$[\![x : A \to B]\!]$ is the set of functions φ with domain $[\![A]\!]$ such that for all $< \alpha, \beta > \in \varphi$, $\beta \in [\![B[\alpha / x]]\!] = \gamma_\alpha$.

Soundness of Function Elimination – Conclusion

We need to know that $[\![f(a)]\!]_{B[a/x]}$ is defined, i.e.

there is a β in $[\![B[a/x]]\!]$

$$\beta \lhd [\![f(a)]\!].$$

We know that for any $< \alpha, \beta > \in \varphi_f$, $\beta \in [\![B[\hat{\alpha}/x]]\!]$,

by the premise for f.

To say $[\![a]\!]_A = \alpha$ means $\alpha \lhd a$. (Recall "Membership and Approximation.")

Soundness of Function Elimination –Conclusion

By the Substitution Lemma,

$$\text{if } \gamma \lhd B\left[\widehat{\alpha} / x\right], \text{ then } \gamma_{\alpha} \lhd B\left[a / x\right],$$

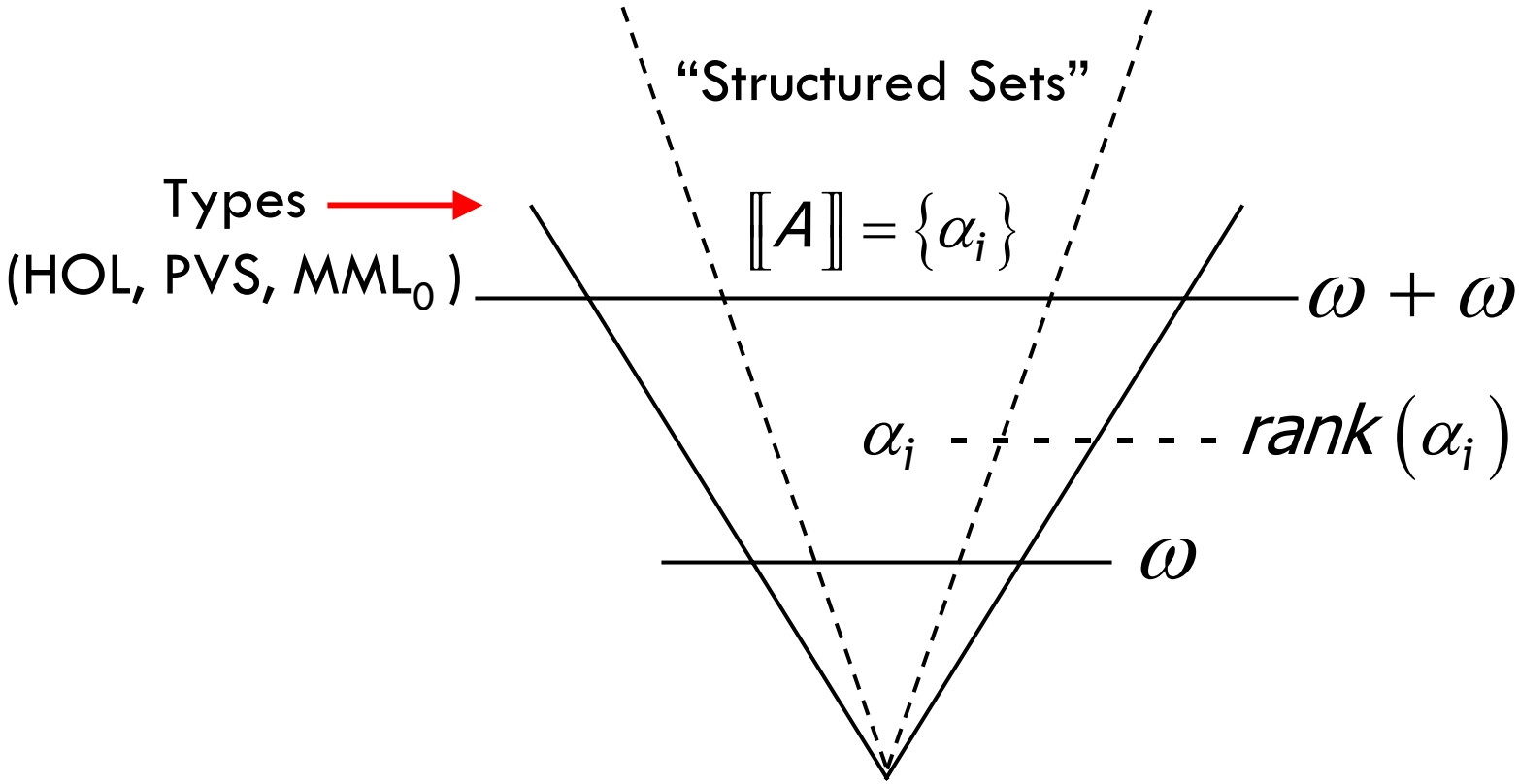
$$\text{thus, } \gamma = \left[\!\left[B\left[\widehat{\alpha} / x\right] \right]\!\right] = \left[\!\left[B\left[a / x\right] \right]\!\right].$$

Likewise, if $\beta \lhd f\left(\widehat{\alpha}\right)$ then $\beta \lhd f\left(a\right)$.

But by the Substitution Lemma, we know $\beta \lhd f\left(\widehat{\alpha}\right)$, since $\varphi_f \lhd f$ and $\beta \lhd \varphi_f\left(\widehat{\alpha}\right)$.

Hence $\left[\!\left[f\left(a\right) \right]\!\right]_{B[a/x]} \in \left[\!\left[B\left[a / x\right] \right]\!\right]$ as required.

Review of Set Theoretic Semantics



Cumulative Hierarchy of Sets

Issues with Nuprl Type Theories

Some Nuprl theorems contradict classical set theory and classical logic.

(a) Nuprl can solve this recursion equation on types:

$$T = (T \to \mathbb{B}) \to \mathbb{B}$$

using $\mu X.(X \to \mathbb{B}) \to \mathbb{B}.$

(b) Nuprl's domain theory (bar types) includes this result:

$$\neg\exists h : \overline{\mathbb{N}} \to \mathbb{B}.\ \ \forall x : \overline{\mathbb{N}}.\ (h(x) = t \ \textit{iff}\ x \downarrow)$$

Nuprl has Union and Intersection

The types $A \cup B$, $\bigcup_{x \in A} B_x$, $A \cap B$, $\bigcap_{x \in A} B_x$, $x : A \cap B$

all violate Howe's construction as outlined.

$$\left(\{0\} \to \{0\}\right) \cup \left(\{1\} \to \{1\}\right)$$

includes $0 \mapsto 0$ and $1 \mapsto 1$ but no

$$\alpha \mapsto \beta$$

can approximate both.

Outline

1. Relating theories and systems
2. The structure of theories
3. Aids to organization and search

Theory Structure

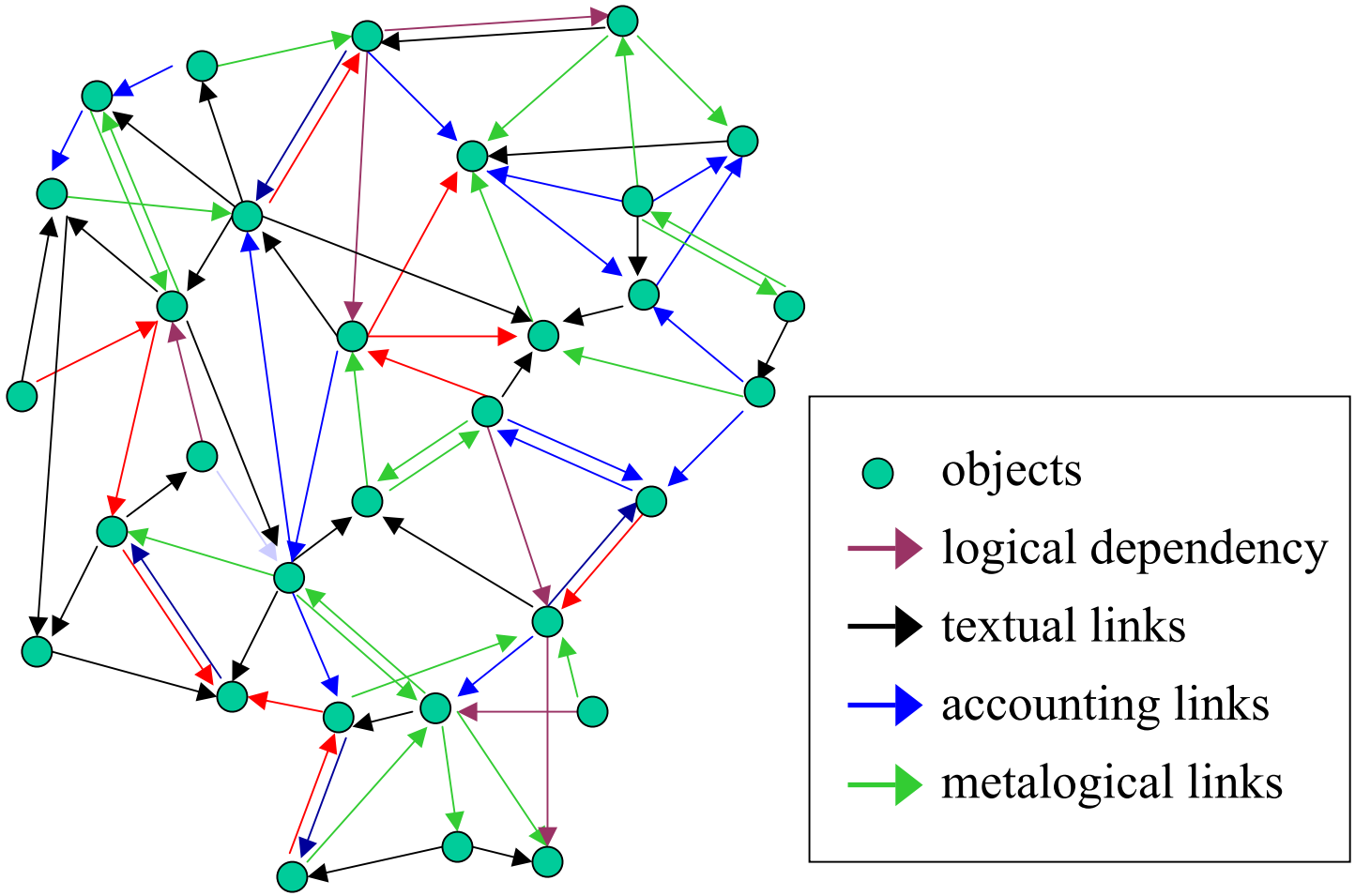
Should theories be directory-like structures, as in Automath contexts (a tree of knowledge)?

These can be internalized as dependent records.

{G:Type, op:GxG $\longrightarrow$ G; id:G, ax1: Assoc(G, op);
ax2: Identity(G, op, id); …}

The FDL takes a different approach, more like a database than a file system.

Information Graph of the FDL



FDL contains formal objects

rules

definitions

algorithms, code

conjectures

specifications

theorems

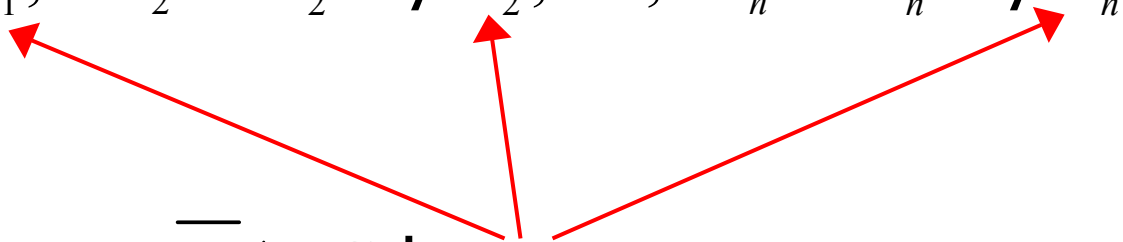
inferences

proofs, partial proofs

certificates

Inferences

$\overline{H_1} \vdash G_1$ by J_1, $\overline{H_2} \vdash G_2$ by J_2, $\cdots$, $\overline{H_n} \vdash G_n$ by J_n



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

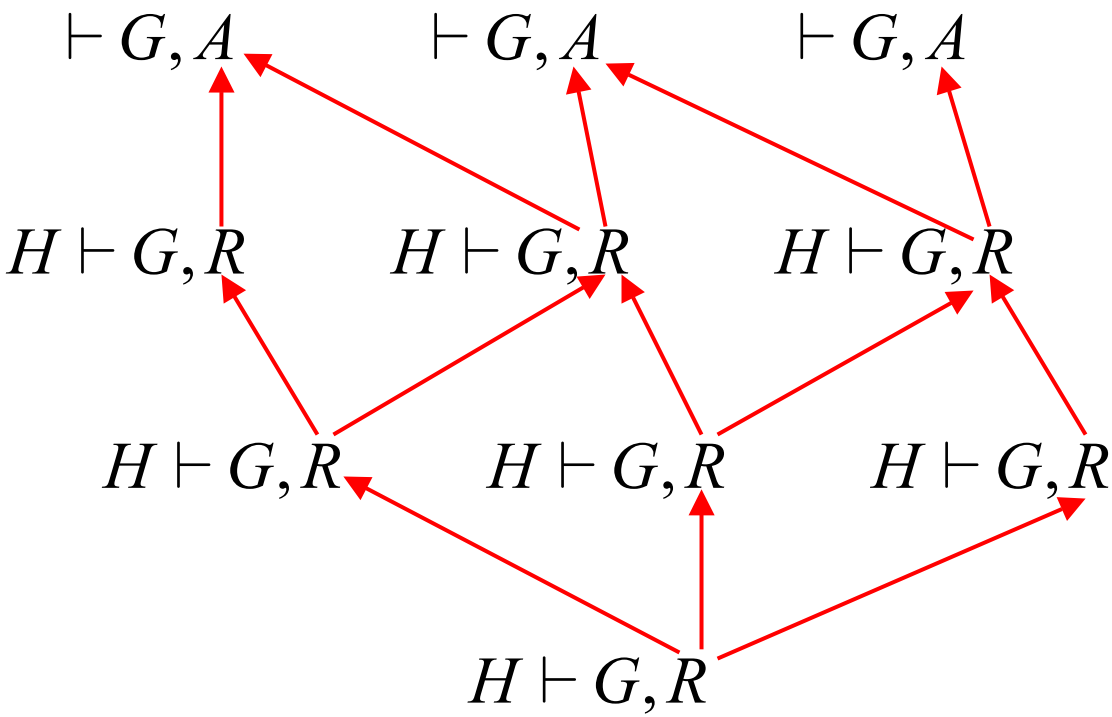
$\overline{H_i}$ a list of formulas (terms)

G_i a formula (term)

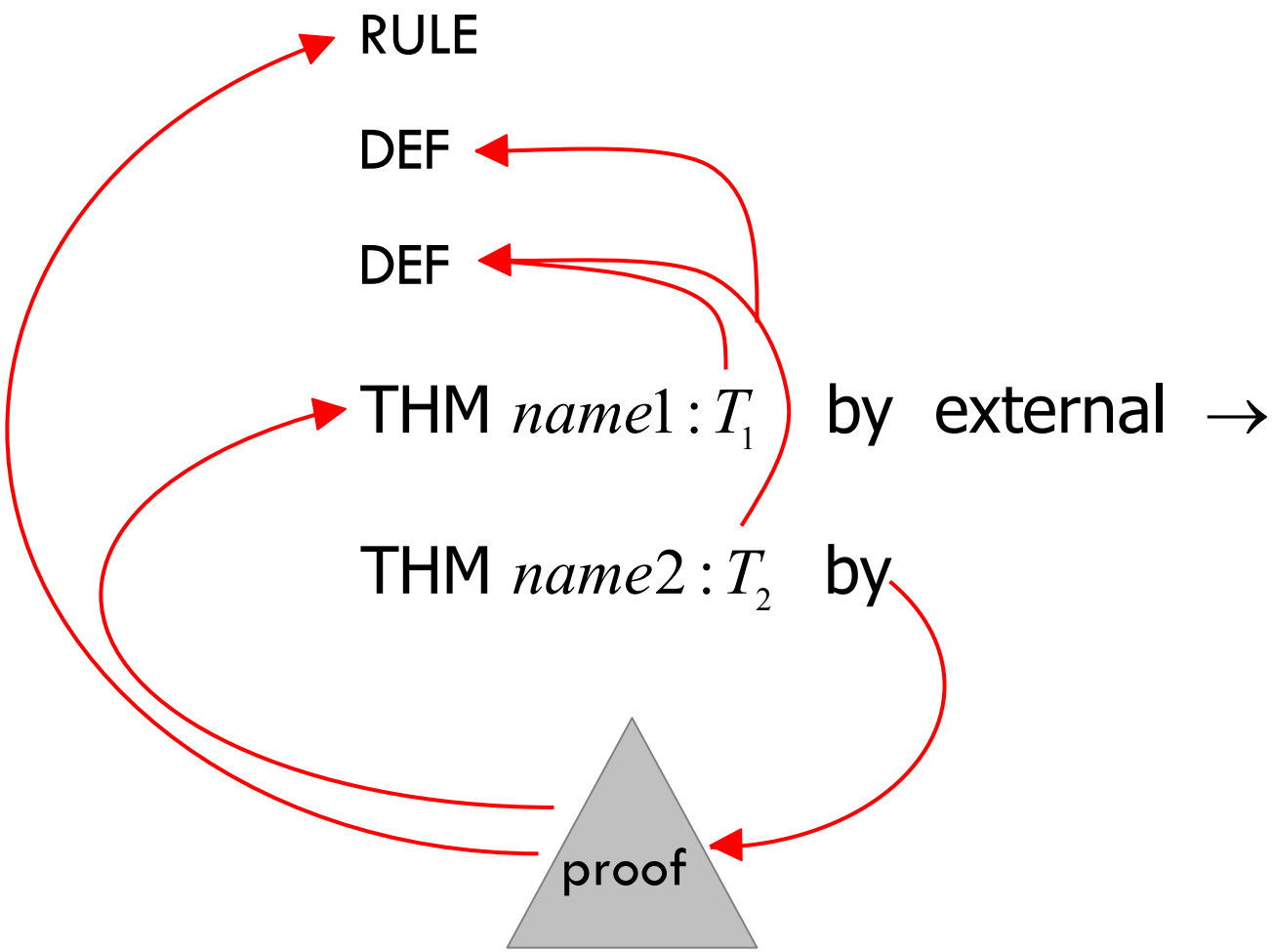
J_i a justification (rule, tactic)

Proof

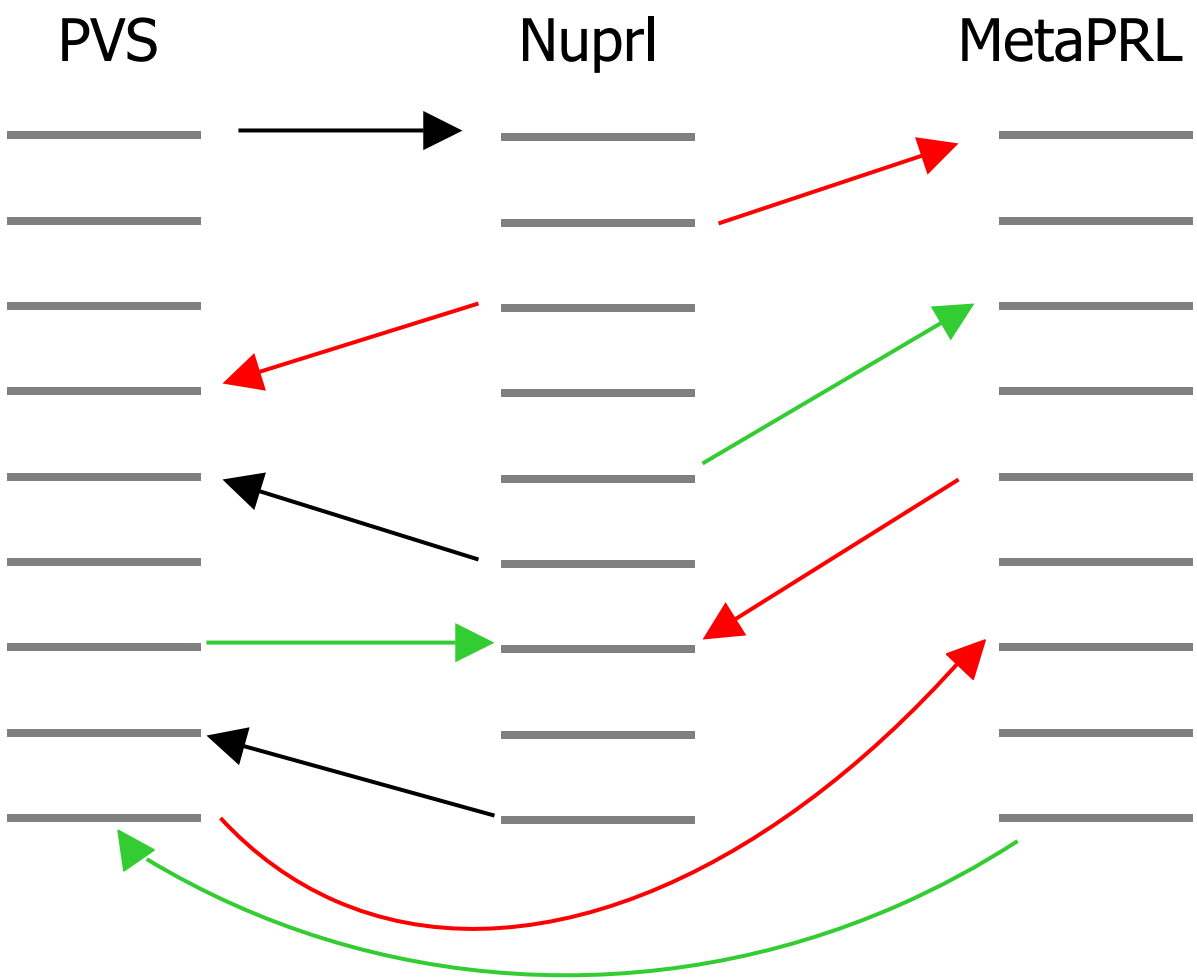
A proof is a dag of inferences



Example of Dependencies



FDL allows sharing among collections



FDL performs archival functions

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

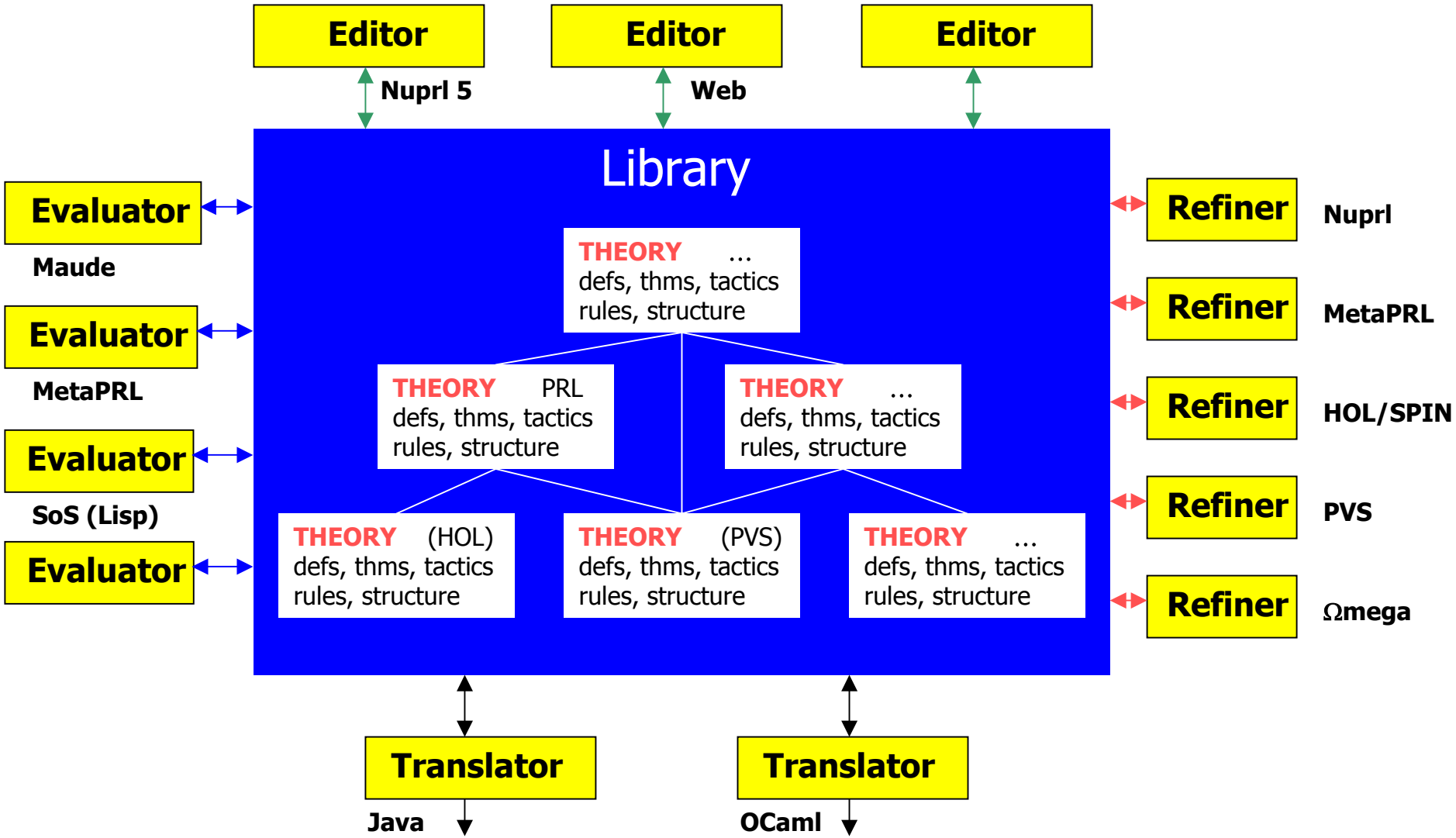
Coq 5.0 created the following extract for the Fundamental Theorem of Algebra (June 14, 2003).

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).

Formal Digital Library



Prototype FDL – Data (Manual 3.1)

Organized to eventually support closed maps

$$D \rightarrow Term(D)$$

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.

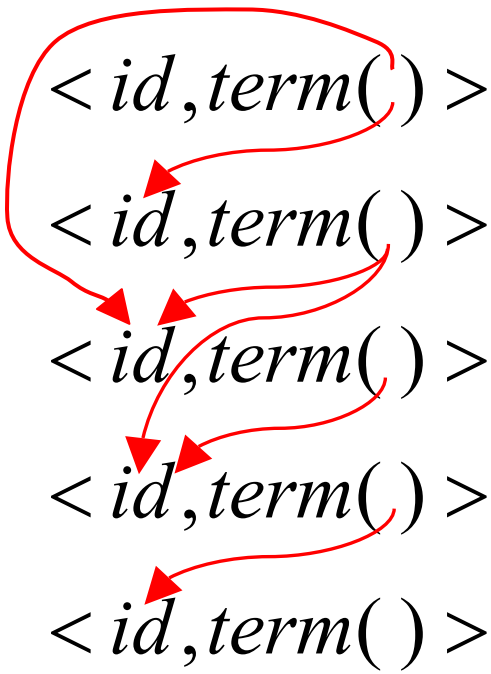
Basic data structure is the library table.

Closed Maps

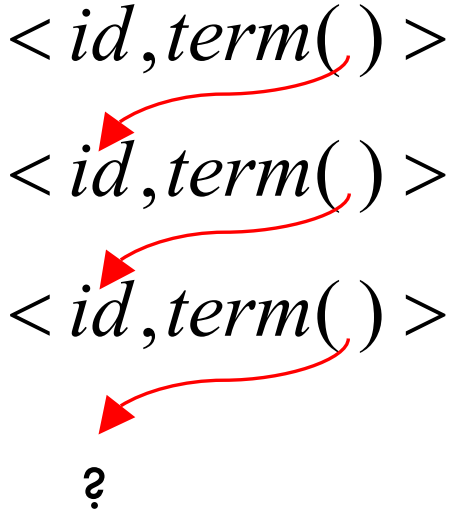
$$D \rightarrow Term(D)$$

closed under reference (no dangling pointers)

closed



open



Abstract Object Identifiers

$$D \cong D'$$

implies

$$\text{Term}\,(D) \equiv \text{Term}\,(D')$$

i.e. $d \leftrightarrow d'$ implies

$$\text{def}(d) = \text{def}(d')$$

$$\text{thm}(d) = \text{thm}(d')$$

$$\text{pf}(\text{thm}(d)) = \text{pf}(\text{thm}(d'))$$

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

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

Outline

1. Relating theories and systems
2. The structure of theories
3. Aids to organization and search

Objective

- Organize digital formal math by **automated** and **adaptable** means
- Exploratory analysis of formal structure

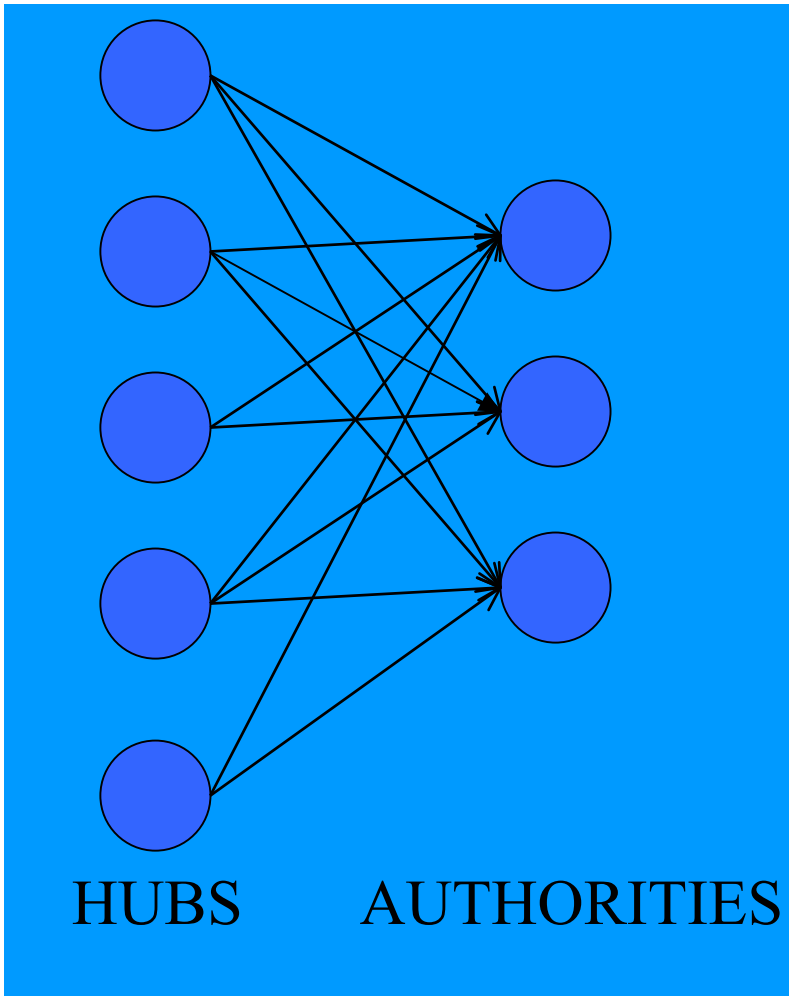
Motivation

- Collections are growing large; need for better automation, search, visualization, personalization <*very broad*>
- Can we exploit it in the same was as is done with the structure of the web?
- Formal Mathematics is rich with structure; can we exploit this as some kind of metadata?

Method

- Graph-based approach to exploit meta-information from the structure
  - relationships between theories are hidden, mass of objects, ordered by humans
- Internet search engines seems really smart; they exploit structure
  - Rank results according to "authority," Google's Page Rank
  - Teoma (www.teoma.com) allows you to "refine" search based on clusters
- *Equate hyperlinks with dependencies, use "HITS"*

Kleinberg's "HITS" Algorithm



- *Hubs* point to good *authorities* & *authorities* point to good *hubs*
- Sparse adjacency matrix, A
  - Hub vector = 1st eigenvector of AA^T
  - Authority vector = 1st eigenvector of A^TA
- Communities
  - Strongest hubs and authorities from respective non-principle eigenvectors
  - Currently there are improved techniques relying on the same graph

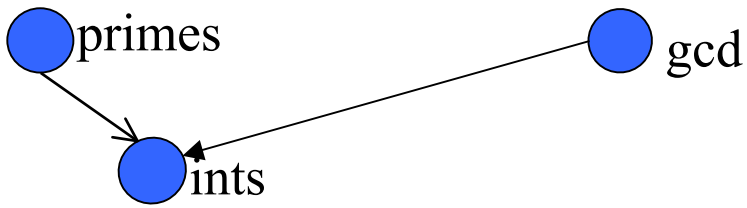
Iterative Algorithm

- Define $x^{<n>}$, $y^{<n>}$ authority, hub weight for node n in graph G with edges E.
- $x^{<n>} \leftarrow \sum_{q:(q,n)\in E} y^{<q>}$
- $y^{<n>} \leftarrow \sum_{q:(n,q)\in E} x^{<q>}$

- Iterate ~ 20 times, normalizing after each.

Design Decisions

- What are analogous to hubs and authorities in formal math?
- Dependency Graphs
  - Theorems and Definitions (Rules, Tactics?)
  - Definitions have no (logical) dependencies
- Direct Forward Dependencies
- Control over size and components of Graph, "Seeding"



Graph Statistics

- Nuprl5 Standard & Num Thy Subset
- Bickford's Event Systems

Nodes	Thm	Def	Max Outlinks	Max Inlinks	Edges	Assortativity
811	**646**	**165**	**58**	**637**	**8765**	**-0.2949**
328	**260**	**68**	**56**	**257**	**3397**	**-0.2848**
1795	**1306**	**489**	**210**	**708**	**16648**	**-0.15896**

Nuprl5 Number Theory

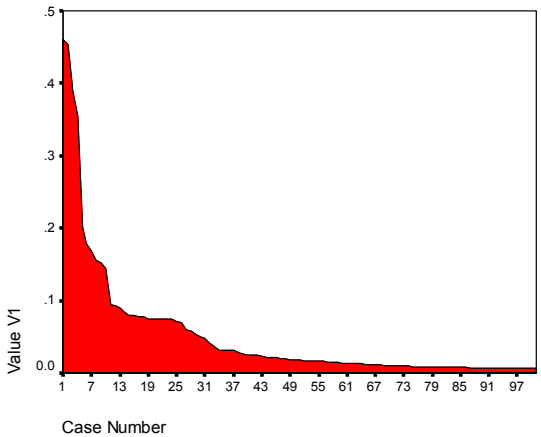
Standard and Numthy Communities

1	listify_length, select_listify_id, int_seg_ind, select_append_front, decidable__ex_int_seg, or, decidable, so_apply1
2	fincr_formation, fincr_wf, fincr_wf2, equiv_rel_functionality_wrt_iff
3	fib_coprime, gcd_sat_gcd_p, gcd_sat_pred, fib_wf, gcd_wf ycomb, not_wf
4	atomic_char, assert_of_eq_int, prime_elim, assert_of_eq_atom, le_wf

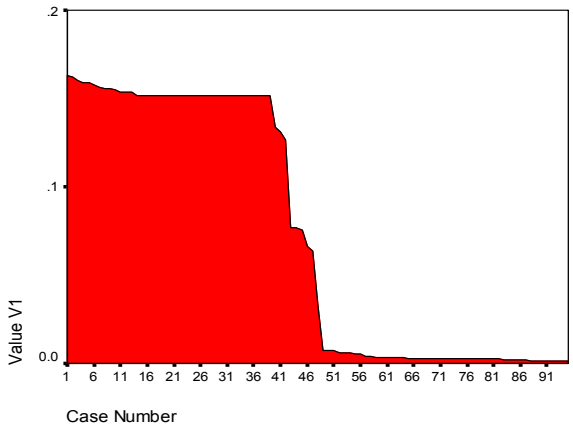
1	divides_of_absvals, absval_assoced, absval_wf
2	chrem_exists_aux_a, gcd_ex_n, chrem_exists_aux, atomic_char, prime_elim, gcd_exists_n, bezout_ident_n, chrem_exists
3	div_3_to_1, div_2_to_1, div_4_to_1, divide_wf, nequal
4	eqff_to_assert, eqtt_to_assert, assert_of_bnot, assert_of_band, prop

Weight Distributions Nuprl5

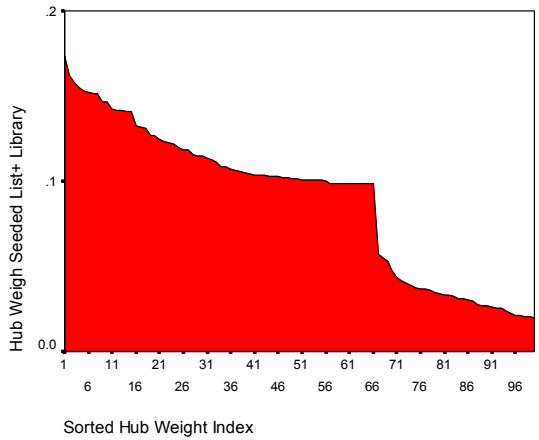
AUTHORITIES



HUBS



HUBS*

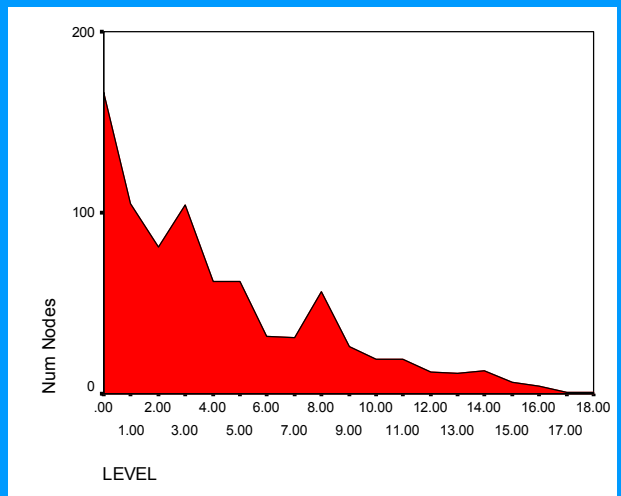


- Step-like Weight Distributions
  - Booleans, lower plateau on authority graph
  - Stratified development of hubs, Bickford's extended list theory

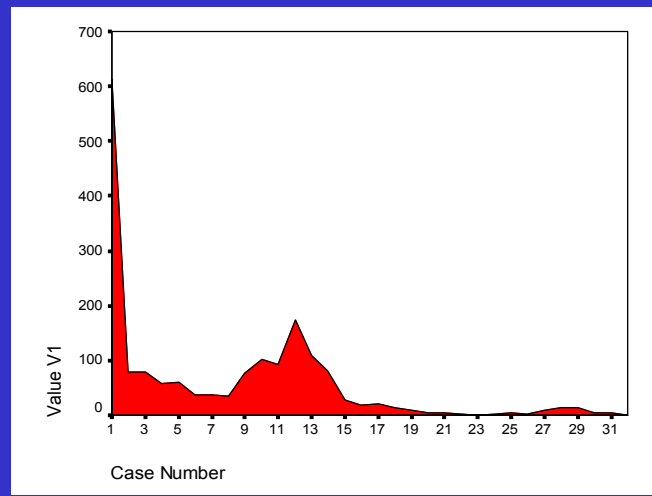
Level Distribution

- Def: A node, *n*, is in level, *i*, iff *n* depends only on nodes in levels < *i*, and *i* is the smallest value for which this is the case.
- Characteristic peaks found, suggest characteristic size/complexity of proofs

NUPRL5



EVENT SYSTEMS



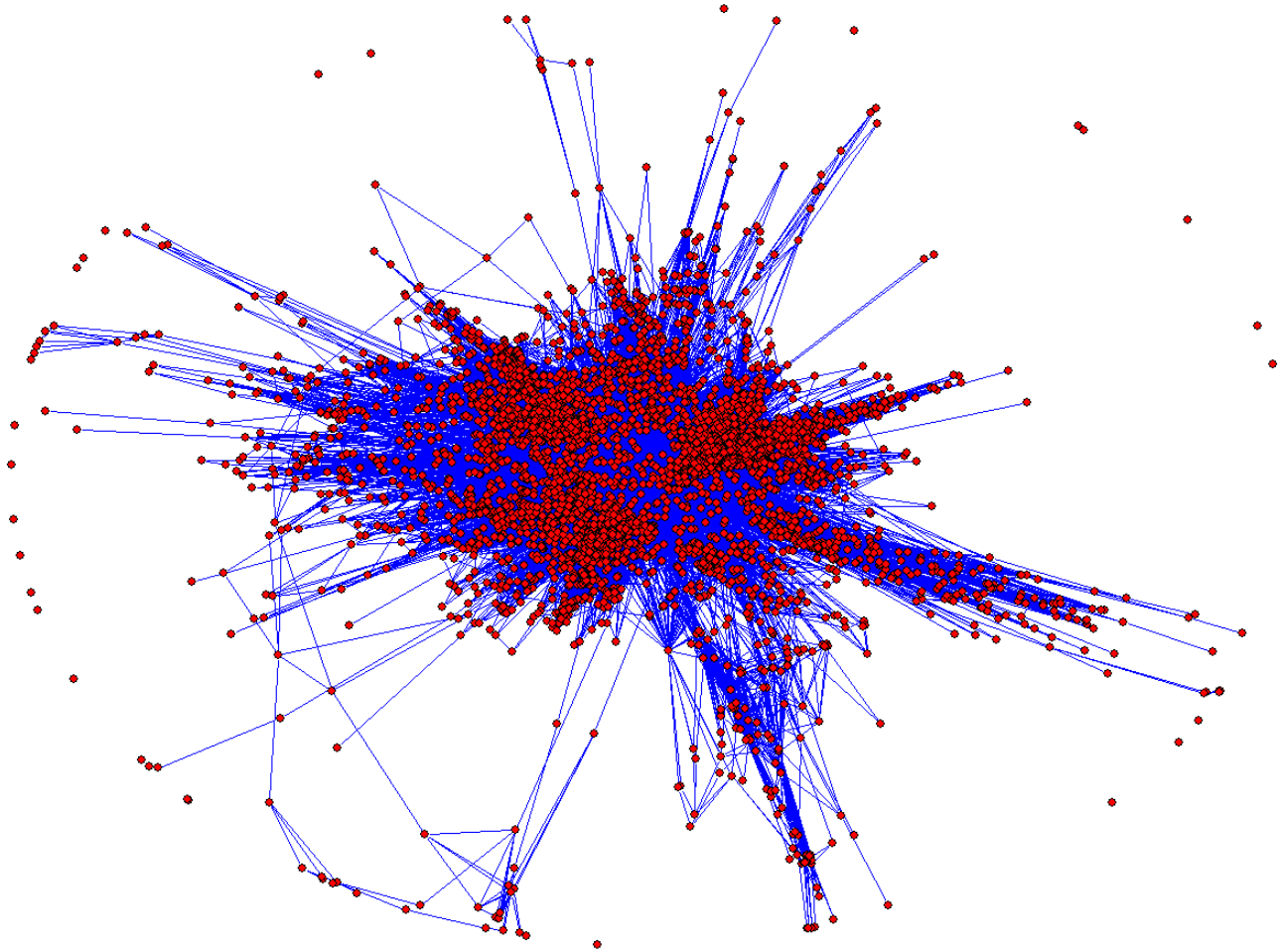
HELM Coq Library Comparison

- Nuprl5 Event Systems
- Coq in HELM

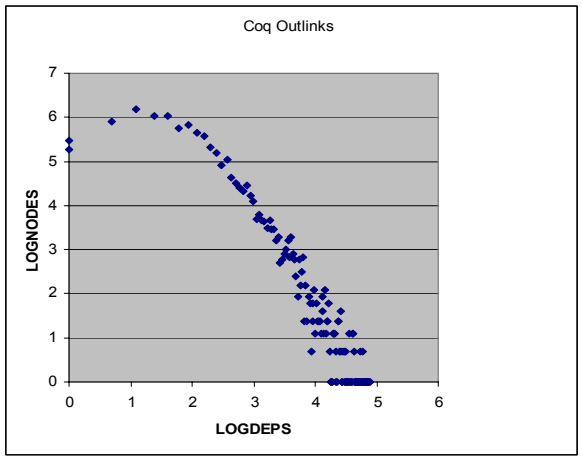
Nodes	Thm	Def	Max Outlinks	Max Inlinks	Edges	Assortativity
811	646	165	58	637	8765	-0.2949
1795	1306	489	210	708	16648	-0.15896
5346	5170	176	132	3093	63093	-0.1884

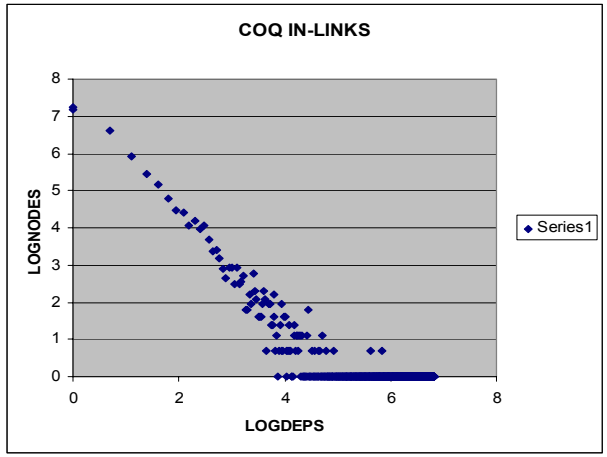
132 was /Coq/Reals/RiemannInt/RiemannInt_P28

Coq Graph

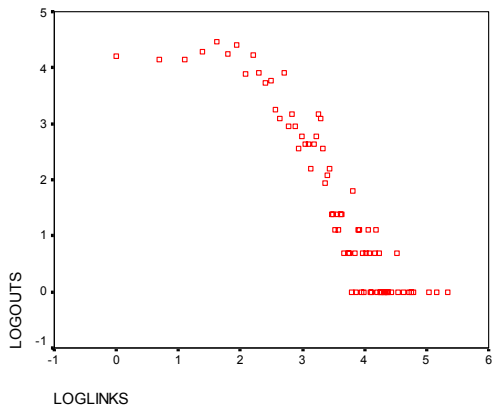


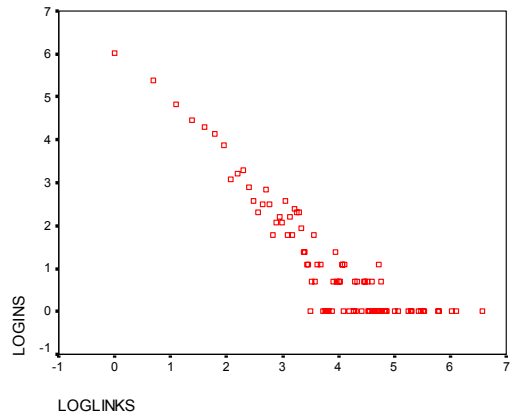
Coq Link Analysis





- Similar link distribution pattern





Coq HITS

AUTHORITIES	"HUBS"
/Init/Logic/eq.ind /Init/Logic/eq_ind /Reals/Rdefinitions/R /Init/Logic/eq_ind_r /Reals/Rdefinitions/R0 /Init/Datatypes/bool.ind /Reals/Rdefinitions/Rplus /Reals/Rdefinitions/R1 /Reals/Rdefinitions/Rmult	/Reals/Rtrigo_alt/cos_bound /Reals/SeqProp/ cv_speed_pow_fact /Reals/RiemannInt/ RiemannInt_P12 RiemannInt_P6 RiemannInt_P28 /Reals/Rtopology/Heine /Reals/Alembert/Alembert_C2 /Reals/RiemannInt/ RiemannInt_P25 RiemannInt_P8

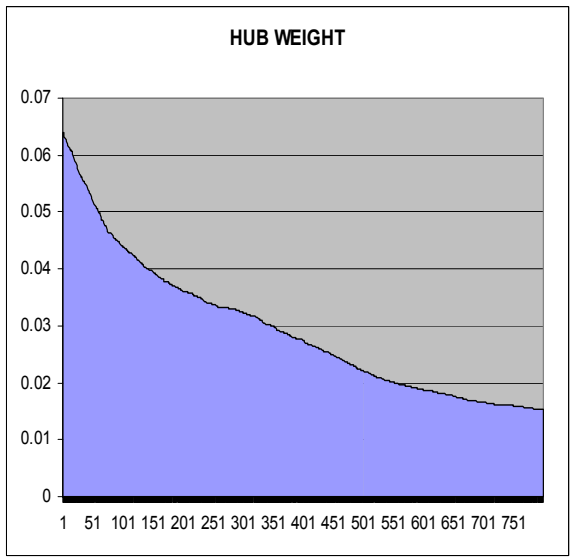
Coq Communities

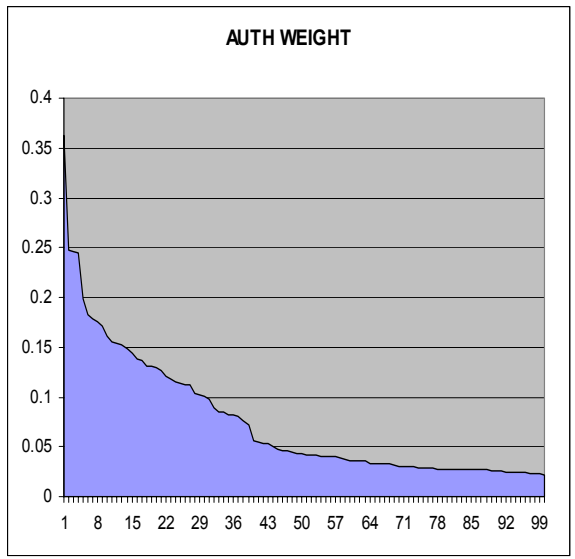
- Logic/Berardi/inv Arith/Wf_nat/induction_gtof1 Arith/Wf_nat/lt_wf_rec1 IntMap/Mapaxioms/MapMerge_assoc Logic/Berardi/i Logic/Berardi/j Bool/Bvector/Vconst
- Bool/Bvector/vector_rec Bool/Bvector/vector_ind Init/Datatypes/bool_ind Init/Datatypes/bool_rec
- Logic/ClassicalFacts/f1_o_f2 IntMap/Mapaxioms/FSetUnion_idempotent IntMap/Mapc/MapMerge_idempotent_c Logic/Diaconescu/proof_irrel Logic/Berardi/retract_rect IntMap/Mapaxioms/MapMerge_assoc Logic/Eqdep/eq_dep_dep1 Logic/Eqdep/eq_dep1_eq
- Relations/Newman/Newman Init/Datatypes/bool_rect Bool/Bvector/vector_rect IntMap/Mapaxioms/MapMerge_assoc

Coq Communities II

- Logic/Hurkens/lemma2 Logic/Hurkens/U Logic/Hurkens/sb Logic/Hurkens/le Logic/Hurkens/U
- Init/Logic_Type/identity_rect_r Init/Logic_Type/identity_rec_r Init/Logic_Type/identity_ind_r Init/Datatypes/identity_rect
- romega/ReflOmegaCore/Tred_factor3_stable omega/OmegaLemmas/fast_Zred_factor3 ZArith/auxiliary/Zred_factor3 romega/ReflOmegaCore/Tred_factor3
- ring/Ring_abstract/minus_varlist_insert_ok ring/Ring_abstract/minus_varlist_insert ring/Ring_theory/Th_plus_zero_left2 ring/Ring_theory/Th_plus_comm

Coq HITS





- Smoother weight degradation
- Very small hub values

References

P. Aczel. Notes on the simply typed lambda calculus. In Berger and Schwichtenberg. 1999.

P. Aczel. The type theoretic interpretation of constructive set theory: Inductive definition. In Logic, Methodology and Philosophy of Science VII.1986.

P. Aczel and M. Rathjen. Notes on Constructive Set Theory. Mittag-Leffler Technical Report No.40, 2000/2001.

P. Aczel. The type theoretic interpretation of constructive set theory. In Logic Colloquium '77.

P. Aczel. The type theoretick interpretations of constructive set theory: Choice principles. In The L.E.J. Brouwer Centenary Symposium. 1982.

S. Allen. Abstract identifiers, intertextual reference and a computational basis for recordkeeping. First Monday, 9(2), 2004. www.firstmonday.org

S. Allen. A Non-type-theoretic Definition of Martin-Lof's Types. In Gries. 1997.

H. Barendregt and H. Geuvers. Proof Assistants using dependent type systems. In Handbook of Automated Reasoning. 2001.

S. Berghofer. Proofs, Programs and Executable Specifications in Higher Order Logic. PhD thesis, Technische Universitat Munchen, 2003.

S. Berghofer and T. Nipkow. Executing higher order logic. In Types for Proofs and Programs: TYPES 2000. 2002.d

R. Constable. Types in logic, mathematics and programming. In S. R. Buss, editor, Handbook of Proof Theory. 1998.

D. Howe. Importing mathematics from HOL into Nuprl. In Theorem Proving in Higher Order Logics, volume 1125, of Lecture Notes in Computer Science. 1996.

D. Howe. Semantic foundations for embedding HOL in Nuprl. In volume 1101 of Lecture Notes in Computer Science. 1996.

E. Moran. Adding Intersection and Union Types to Howe's Model of Type Theory [working title]. PhD thesis, Cornell University, 2003.