Information Intensive Proof Technology

A series of lectures by

Robert L. Constable
Computer Science Department
Cornell University

for the **Marktoberdorf Summer School 2003**
An Advanced Study Institute of the NATO Science Committee
and the Institut für Informatik, Technische Universität München, Germany.
on
Proof Technology and Computation
July 29 to August 10, 2003

Series Summary

Working Material: www.nuprl.org

This series arises from two observations:

1. World-wide, provers are accumulating a significant body of general formal math.
   - Perhaps 25k to 30 k theorems
2. In relams where we have extensive holding, provers can "keep up with designers."

Lesson: TRY TO SHARE FORMAL MATH LIBRARIES!

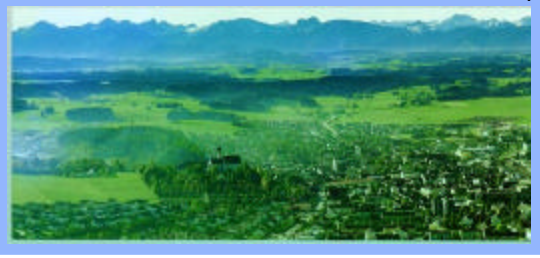
Lecture 1

Series Introduction

and

Type Theories

Marktoberdorf Summer School, 2003



Series Introduction

The topic of the summer school, "Proof Technology and Computation," lies in the intersection of several research areas; I list 5.

1. The foundations of mathematics and computing and information science
    - Type theory and higher-order logic – Principia onward
    - Set theory – ZF onward
    - Constructivity – Intuitionism onward

Series Introduction

2. Logic applied to the problems of philosophy, linguistics, mathematics, and science – applied logic

   - Modeling discrete systems
   - System verification
   - Semantics of language

Series Introduction

3. Programming languages and systems

   - Semantics – operational, denotational
   - Programming logics
   - Programming environments

Series Introduction

4. Formalized mathematics

   - Establishing landmarks, e.g. Fundamental Theorem of Algebra
   - Formal libraries (encyclopedias – “Digital Bourbaki”)
   - Mathematician’s assistant

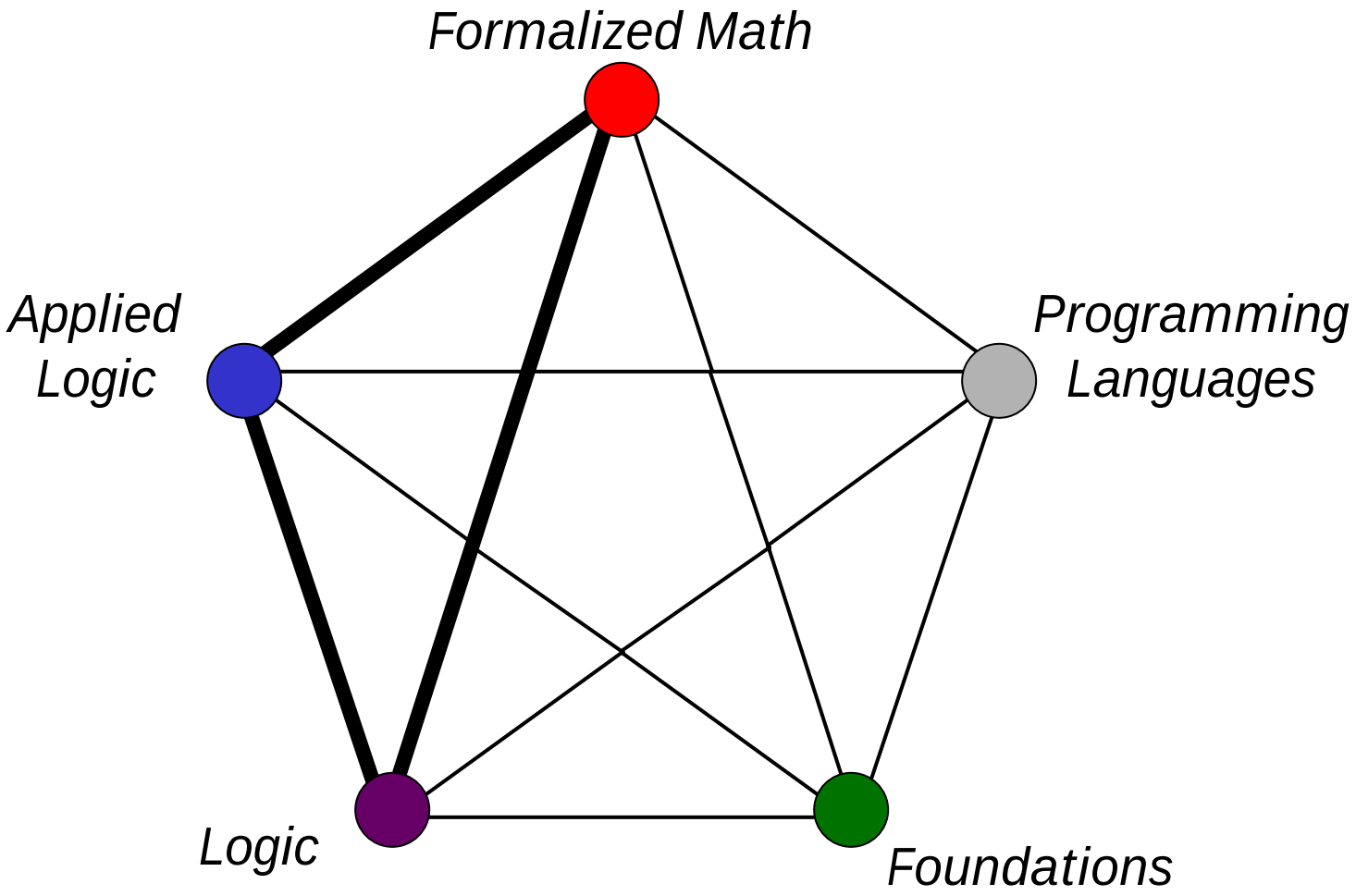
Series Introduction

5. Logic

   - Formal systems
   - Proof theory (and computability)
   - Model theory (and set theory)

These five areas are tightly connected, e.g. performing system verifications generates large amounts of general formal mathematics. Formalizing mathematics reveals foundational issues. Analyzing the foundational issues uses and extends the tools of logic, suggesting new approaches to applications.

Series Introduction



Series Introduction

I will stress the left side – the heavy lines – and the top node.

- Verifications create large amounts of formal knowledge
- Shared collections of formal knowledge accelerate applications
- Sharing raises new technical issues for logic

I will illustrate these ideas over the course of five lectures.

Series Outline

Lecture 1	The formal systems used to verify and formalize; Type Theories; stress variety; comparisons and means of reconciling
Lecture 2	A set-theoretic semantics for type theories; Howe's methods and their potential; Classical Nuprl
Lecture 3	Relating types and sets; preview of "logic in the large"

Series Outline

Lecture 4 — Example of formally developing a large theory and its applications; event systems and foundations of distributed computing

Lecture 5 — Logic in the large; building and modifying a large theory; conclusion, type theory as a peer to set theory

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	PVS
$\mathbb{N}, \mathbb{B}$	$\mathbb{Z}$, *Atom*	$\mathbb{N}, \mathbb{B}, \mathbb{R}$
$A \to B$	$x : A \to B$	$x : A \to B$
$A \times B$	$x : A \times B$	$x : A \times B$
$a = b \ in \ A$	$a = b \ in \ A$	$a = b \ in \ A$
$?x : A, \ ?x : A$	$?x : A, \ ?x : A$	$?x : A, \ ?x : A$

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

Translating Among Theories

$$[\![x : A_0 \rightarrow A_1[x]]\!]_p = [\![\Pi x : A_0 . A_1[x]]\!]_p$$

Note, this is a set of single-valued relations, or graphs.

The graphs are sets of ordered pairs,

$$\{< a_1, b_1 >, < a_2, b_2 >, \ldots\}.$$

Each graph has $[\![A_0]\!]$ as domain. The range depends on the a's, e.g. $b \in [\![A_1[a]]\!]_p$.

Comparing Representative Type Theories

Least Number Principle **(LNP)**:

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

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

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

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 $P\ ?\ \mathbb{N} \rightarrow \mathbb{B}$ 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 $\forall n : \mathbb{N}.\ \exists! r : \mathbb{N}.\ Root\ (n, r)$
- Cor 1.1 $\exists rt : \mathbb{N} \to \mathbb{N}.\ \forall n : \mathbb{N}.\ Root\ (n, rt\ (n))$
- Cor 1.2 $ext\ (Thm1.1)e\ \mathbb{N} \to \mathbb{N}$
- Cor 1.3 $\forall n : \mathbb{N}.\ Root\ (n, ext\ (Thm1.1)(n))$

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 " $ 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.

Extraction from Proofs of Integer Square Root

In Nuprl and Coq, we can extract from the weaker theorem.

$$\forall n : \mathbb{N}.\ ?r : \mathbb{N}.\ Root\,(n, r).$$

We show a Nuprl proof of this result. See also Working Material, Appendix A, page 45-46.

Proof of Root Theorem

$\forall n : \mathbb{N}.\ \exists r : \mathbb{N}.\ r^2 \le n < (r+1)^2$

BY *allR*

$n : \mathbb{N}$

$\vdash \exists r : \mathbb{N}.\ r^2 \le n < (r+1)^2$

BY NatInd 1

.....basecase.....

$\vdash \exists r : \mathbb{N}.\ r^2 \le 0 < (r+1)^2$

BY *existsR* $\lceil 0 \rceil$ THEN *Auto*

.....upcase.....

$i : \mathbb{N}^+,\ r : \mathbb{N},\ r^2 \le i-1 < (r+1)^2$

$\vdash \exists r : \mathbb{N}.\ r^2 \le i < (r+1)^2$

BY *Decide* $\lceil (r+1)^2 \le i \rceil$ THEN *Auto*

Proof of Root Theorem (cont.)

.....Case 1.....

$$i : \mathbb{N}^+,\ r : \mathbb{N},\ r^2 \le i - 1 < (r+1)^2,\ (r+1)^2 \le i$$

$$\vdash \exists r : \mathbb{N}.\ r^2 \le i < (r+1)^2$$

BY *existsR* $\lceil r + 1 \rceil$ THEN *Auto* '

.....Case 2.....

$$i : \mathbb{N}^+,\ r : \mathbb{N},\ r^2 \le i - 1 < (r+1)^2,\ \neg\left((r+1)^2 \le i\right)$$

$$\vdash \exists r : \mathbb{N}.\ r^2 \le i < (r+1)^2$$

BY *existsR* $\lceil r \rceil$ THEN *Auto*

The Root Program Extract

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

$$
\begin{array}{l}
\text{let } rec \;\; sqrt \;\; i = \\
\quad \text{if } i = 0 \text{ then} < 0, pf_0 > \\
\qquad \text{else let } < r, pf_{i-1} > = sqrt\,(i - 1) \\
\qquad \text{in if } (r + 1)^2 \le n \text{ then} < r + 1, pf_i > \\
\qquad\quad \text{else } < r, pf_i' >
\end{array}
$$

Deduction Systems

HOL, Nuprl and PVS all use a version of Gentzen's sequents to organize proofs.

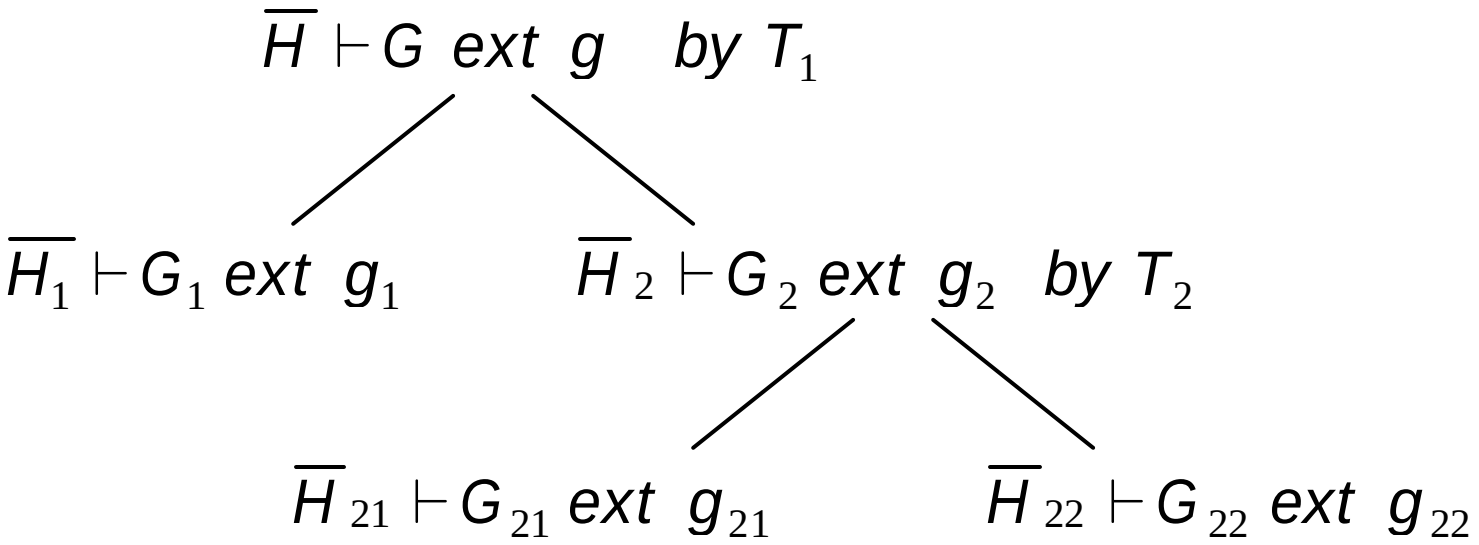
$$H_1, \ldots, H_n \vdash G \qquad \text{or} \qquad H_1, \ldots, H_n \vdash G_1, \ldots, G_m$$

$$\overline{H} \vdash G \qquad\qquad \overline{H} \vdash \overline{G}$$

Typically, hypotheses are named; in Nuprl we use:

$$x_1 : H_1, \ldots, x_n : H_n \vdash G\,(x_1, \ldots, x_n)\ ext\ g\,(x_1, \ldots, x_n)$$

Tactic-tree Proofs



The conclusions are functional in the hypotheses. The extracts g_i are synthesized bottom up, proofs are built top down.

How to Reconcile Theories?

Gödel's approach for number theory

$$A \not\vdash B$$

$$Ey : A.P$$

Bishop's approach for analysis: can read the mathematics both constructively AND classically.

Translating Among Theories

Types ? Sets

Sets ? Types

Example of types into sets (Troelstra, Dybjer, Pitts)

- Pairs
- Functions
- Dependent function spaces

Translating Among Theories

Dybjer's notations:

$$[\![x]\!]_p = p(x)$$

$$[\![\Pi x : A_0 . A_1[x]]\!]_p = \prod_{U ? [A_0]p} [\![A_1[x]]\!]_{p_x^u}$$

$$[\![I x : A . a[x]]\!]_p = \{ < u , [\![a[x]]\!]_{p_x^u} > | u ? [\![A]\!]_p \}$$

Dependent Types

Imagine the root example as:

$$x : \mathbb{N} \rightarrow \{y : \mathbb{N} \mid Root(x, y)\}$$

This expression is possible in Nuprl and PVS. The dependency is very expressive; the PVS functions in this type are sets of ordered pairs,

$$root\text{–}PVS = \{<0,0>, <1,1>, <2,1>, <3,1>, <4,2>, \ldots\}$$

In general, $x \in \mathbb{N}, r \in \{y : \mathbb{N} \mid Root(x, y)\}$.

Replies to Questions

1. Gödel's method of reconciling

$$A \;?\; B == \neg\left(\neg A \;\; \& \neg B\right)$$

2. Bishop's method relies on an open computation system and logic; we cannot say what we can in recursive mathematics; for example, there is no computable functional to decide numerical function equivalence:

$$\neg\exists\, Eq : \left(\mathbb{N} \to \mathbb{N}\right)^2 \to \mathbb{B}.$$

$$\forall f, g : \mathbb{N} \to \mathbb{N}.\ \left(Eq\left(f, g\right) = true\right.$$

$$\Leftrightarrow$$

$$\left.?x : \mathbb{N}.\ f\left(x\right) = g\left(x\right)\right)$$

Comments

3. Stefan Berghofer did program extraction in HOL!!

   Program Extraction in Simply-typed Higher Order Logic.

   (*In* H. Geuvers and F. Wiedijk, editors, *Types for Proofs and Programs,* International Workshop (TYPES 2002), LNCS 2646. Springer-Verlag, 2003.)

4. The Nuprl Web site, www.nuprl.org, has a vast collection of papers and formal proofs, including Howe's paper.