Lectures on Computational Type Theory

From Proofs-as-Programs to Proofs-as-Processes

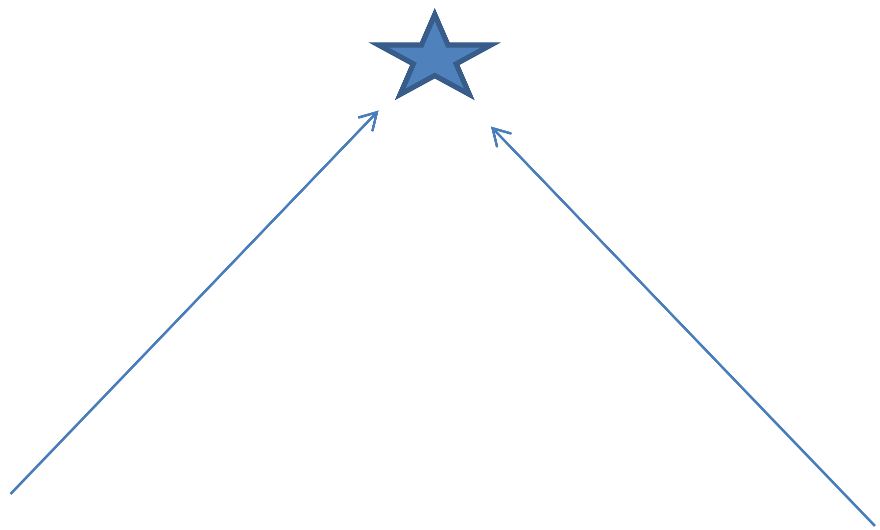
Robert L. Constable
Cornell University

Lecture Schedule

- Lecture 1: Origins and Introduction to Computational Type Theory (CTT)
- Lecture 2: Logic in CTT
- Lecture 3: Proofs as Programs
- Lecture 4: The Logic of Events and Proofs as Processes

Obsession

Since 1971 I've been obsessed with the connection between formal math and programming languages. I see a convergence.



formal computational math
PSEs

richly typed PLs
LPEs

Why Study CTT?

1. CTT is a very rich type theory, slightly older sibling of CIC as implemented in Coq. Here is a comparison:

CTT	CIC
grounded in semantics	grounded in proof theory
(partial equivalence)	(strong normalization)
implicitly typed	explicitly typed
extensional equality	intensional equality
predicative *	impredicative
Turing-complete	sub-Turing complete
elegant objects theory	objects ?
processes are primitive	no primitive processes
proofs as proof trees	proofs as proof scripts

*Deep insight of Poincaré.

Why Study CTT?

2. A "Revolution" in programming is coming.

Logical Programming Environments (LPEs) providing advanced formal methods including provers, like Coq , HOL, and Nuprl, are coming to industry, there will be room for many new ideas and a "race to the top."

Intel understood formal methods for hardware, will they get it for software? Will Microsoft or will they have the "best 1970s technology"?

Why Study CTT?

3. The "Idea Revolution" has happened, built on automated reasoning, constructive logics, correct-by-construction programming, large libraries of formal knowledge.

These ideas will be manifest broadly, from mathematics and physics to biology -- with exciting breakthroughs.

Selected Notable Examples

- Four Color Theorem formalization – Gonthier
- Kepler Conjecture Work – Halles with HOL team and INRIA team
- Constructive Higman’s Lemma – Murthy
- Prime Number Theorem – Harrison, Avigad
- Kruskal’s Theorem – Seisenberger
- Intel’s verified floating point arith -- Harrison
- POPLMark Challenge – Coq,Twelf,HOL
- Paris driverless Metro line 14 – Abrial, B-tool
- Mizar’s Journal of Formalized Mathematics

Selected Notable Examples

- Automatically Generated Correct-by-Construction Authentication Protocols – Bickford
- Verified ML Compiler – Dr. Who
- Other examples?

Lecture 1 Outline

Brief history of type theory from 1908 to 2010

Overview of Computational Type Theory (CTT)

- CTT Computation System
  - terms, evaluation, Howe's squiggle (~)
- CTT Type System
  - Martin-Löf's semantic method,
  - Allen's PER model

Exercises and Recommended Reading

Historical Backdrop

The research that led to modern type theories was done against the backdrop of a “crisis” in mathematics which caused logicians to look at ways to be more rigorous and precise about basic concepts. Key players in setting the stage were:

Frege	Cantor
Begriffsschrift	Set Theory
1879	1874

Origins

Russell & Whitehead Hilbert Brouwer Zermelo

Church Gentzen Herbrand Kolmogorov

McCarthy Kleene Kreisel Heyting de Bruijn Bishop

Milner Scott Girard Martin-Löf

Lisp (Algol68) ML

HOL Nuprl Coq Alf (Automath) Mizar

Origins

History

Greeks

Kronecker
Brouwer
Weyl
Baire
Borel
Bishop

Gentzen
Heyting
Kleene
M-L
de Bruijn

Girard
Coquand



Origins continued

Philosophical Issues

Logicism	Intuitionism	Formalism
Russell	Brouwer	Hilbert

Origins continued

Philosophical issues are harmonized in CTT.

-- CTT is formal but very abstract

-- CTT is a constructive logic, but is classically sensible and consistent

-- CTT uses propositions-as-types which relates logic and mathematics at a fundamental level

Foundational Criteria

What is required for a constructive theory to be an adequate foundation for computer science?

1. Proofs-as-programs works and the theory is a programming language and programming logic combined that can be well implemented.
2. Computational mathematics, e.g. numerical methods, computational geometry and algebra etc. is grounded in this theory.

Foundational Criteria continued

3. Can provide a semantics to any programming language.
4. All axioms and inference rules have a computational meaning, justified by propositions-as-types.
5. The theory explains and justifies the principles of computing as they unfold.
6. Reasoning can well automated well.
7. The theory can be read classically.

Reading for Lecture 1

All reading material can be found at
www.nuprl.org

Douglas Howe: Equality in Lazy Computation Systems, LICS 89

Stuart Allen: Non-type theoretic definition of Martin-Löf's types, LICS 87
Nax Mendler: *Inductive Definition in Type Theory*, PhD thesis, 1988 see Chapter 4

Robert W. Harper: Constructing Type Systems over an Operational Semantics, *J. of Symbolic Computation*, 14, 71-84, 1992

Christoph Kreitz: Nuprl 5 Reference Manual and User's Guide, 2002
www.nuprl.org/html/02cucs-NuprlManual.pdf see Appendix A

Computational type theory: Scholarpedia, 4(2):7618 2008

Lecture 2 Outline

CTT Inference System

- judgements and sequents
- functionality semantics of sequents
- propositions-as-types principle

Intuitionistic Propositional Calculus in CTT

Intuitionistic Predicate Calculus

Heyting Arithmetic (HA)

Proofs as programs

Exercises

Reading for Lecture 2

All reading material is at www.nuprl.org

Proofs as Programs by Joseph L. Bates and Robert L. Constable, ACM Transactions on Programming Languages and Systems, vol. 7, no. 1, pp. 53-71.

Christoph Kreitz: Nuprl 5 Reference Manual and User's Guide, 2002
www.nuprl.org/html/02cucs-NuprlManual.pdf see A.3 Inference Rules

Implementing Metamathematics as an Approach to Automatic Theorem Proving by Robert L. Constable and Douglas J. Howe, Formal Techniques in Artificial Intelligence: A Source Book, R.B. Banerji (ed.), pp. 45-76, Elsevier Science, North-Holland, 1990.

Lecture 3 Outline

Review and answers to exercises

Programming in CTT, efficient extracts

Universes and Higher-Order Logic

Object-oriented types

  subtyping, Top type, unit records

  records and intersection types

Exercises and Recommended Reading

Reading for Lecture 3

All reading material is at www.nuprl.org

Dependent Intersection: A New Way of Defining Records in Type Theory by Alexei Kopylov, Proceedings of 18th Annual IEEE Symposium on Logic in Computer Science, pp. 86-95, 2003.

Type Theoretical Foundations for Data Structures, Classes, and Objects by Alexei Kopylov, Cornell University Ph.D. Thesis, 2004.

Lecture 4 Outline

Objectives of Proofs-as-Processes

Distributed Computing Model

Event Structures

A Logic of Events

Specifying Protocols

Extracting Processes from Proofs

General Process Model

Reading for Lecture 4

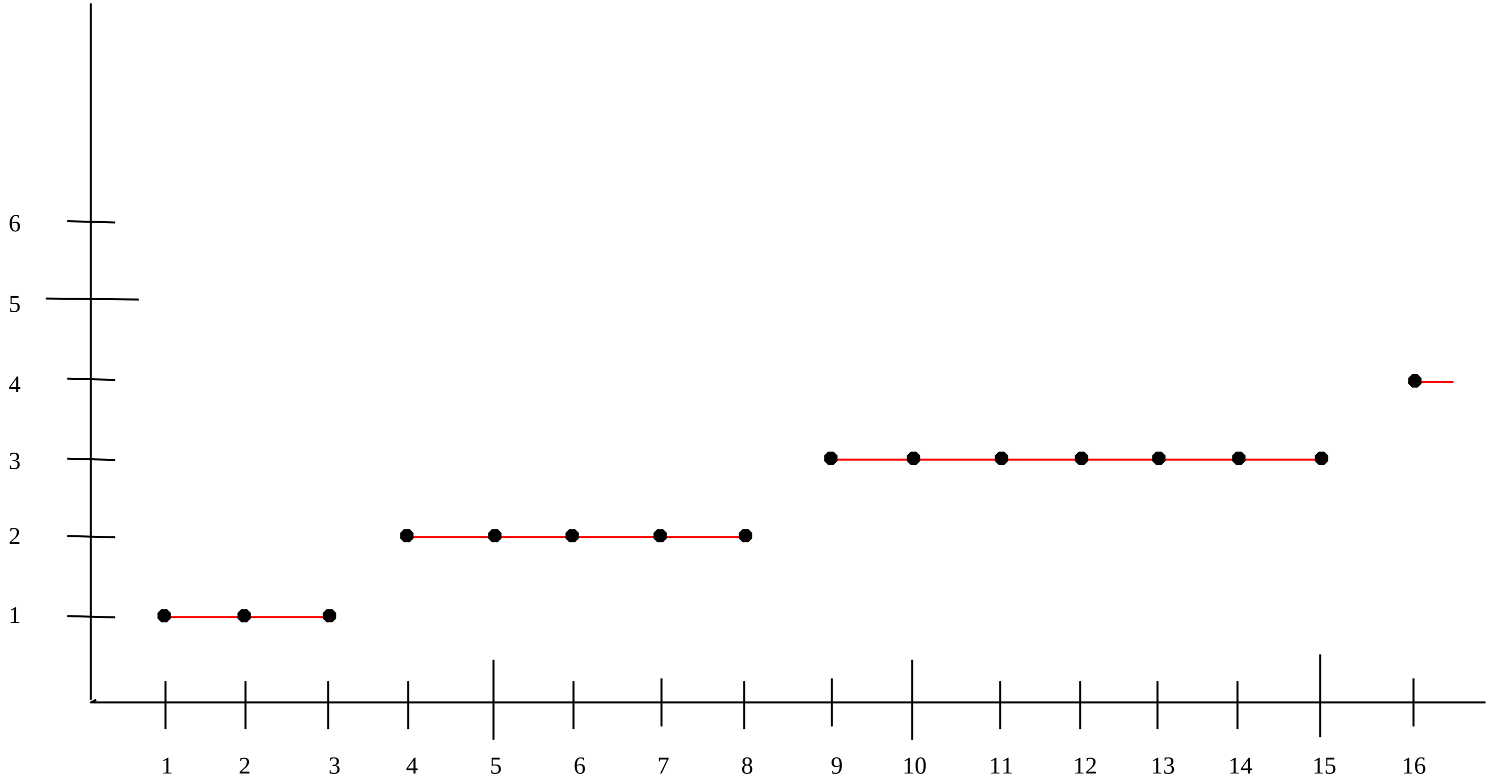
All reading material is located at www.nuprl.org

- *Formal Foundations of Computer Security* by Mark Bickford and Robert Constable, NATO Science for Peace and Security Series, D: Information and Communication Security, Vol. 14, pages 29 - 52, 2008., 2008.
- *Unguessable Atoms: A Logical Foundation for Security* by Mark Bickford, Verified Software: Theories, Tools, Experiments, Second International Conference, VSTTE 2008 Toronto, Canada, pp 30 - 53, 2008.

Lecture 2 Slides

Integer square root example

Integer Square Root



Proof of Root Theorem

$\forall n : \mathbb{N}.\ \exists r : \mathbb{N}.\ r^2 \leq n < r + 1\,^{2}$

BY allR

$n : \mathbb{N}$

$\vdash \exists r : \mathbb{N}.\ r^2 \leq n < r + 1\,^{2}$

BY <u>NatInd</u> 1

. . . . .induction case.....

$\vdash \exists r : \mathbb{N}.\ r^2 \leq 0 < r + 1\,^{2}$

BY existsR ⌈0⌉ THEN Auto

. . . . .induction case.....

$\boldsymbol{i} : \mathbb{N}^{+},\ r : \mathbb{N},\ r^2 \leq \boldsymbol{i} - 1 < r + 1\,^{2}$

$\vdash \exists r : \mathbb{N}.\ r^2 \leq \boldsymbol{i} < r + 1\,^{2}$

BY Decide ⌈ $\underline{r} + 1\,^{2} \leq \boldsymbol{i}$ ⌉ THEN Auto

Proof of Root Theorem (cont.)

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

$\boldsymbol{i} : \mathbb{N}^{+},\ \mathtt{r} : \mathbb{N},\ \mathtt{r}^{2} \leq \boldsymbol{i} - 1 < \ \mathtt{r} + 1\ ^{2},\ \ \mathtt{r} + 1\ ^{2} \leq \boldsymbol{i}$

$\vdash \exists \mathtt{r} : \mathbb{N}.\ \mathtt{r}^{2} \leq \boldsymbol{i} < \ \mathtt{r} + 1\ ^{2}$

BY `existsR` $\lceil \underline{\mathtt{r}} + 1 \rceil$ THEN `Auto'`

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

$\boldsymbol{i} : \mathbb{N}^{+},\ \mathtt{r} : \mathbb{N},\ \mathtt{r}^{2} \leq \boldsymbol{i} - 1 < \ \mathtt{r} + 1\ ^{2},\ \neg\ \ \mathtt{r} + 1\ ^{2} \leq \boldsymbol{i}$

$\vdash \exists \mathtt{r} : \mathbb{N}.\ \mathtt{r}^{2} \leq \boldsymbol{i} < \ \mathtt{r} + 1\ ^{2}$

BY `existsR` $\lceil \underline{\mathtt{r}} \rceil$ THEN `Auto`

The Root Program Extract

Here is the extract term for this proof in ML notation with proof terms (pf) included:

```
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 ≤ n then < r + 1, pf_i >
            else < r, pf_i' >
```

A Recursive Program for Integer Roots

Here is a very clean functional program

$$
\begin{aligned}
r(n) := \textbf{if } & n = 0 \textbf{ then } 0 \\
& \textbf{else } \text{let } r_0 = r\,(n-1) \textbf{ in} \\
& \textbf{if } (r_0 + 1)^2 \le n \textbf{ then } r_0 + 1 \\
& \qquad\qquad\qquad\quad \textbf{else } r_0 \textbf{ fi} \\
& \textbf{fi}
\end{aligned}
$$

This program is close to a declarative mathematical description of roots given by the following theorem.

Efficient Root Program

The interactive code and the recursive program are both very inefficient. It is easy to make them efficient.

$$
\begin{aligned}
&\mathrm{root}(n) := \ \mathbf{if}\ n=0\ \mathbf{then}\ 0\\
&\mathbf{else}\ \mathbf{let}\ r_0 = \mathrm{root}\,(n/4)\ \mathbf{in}\\
&\mathbf{if}\ (2\cdot r_0+1)^2 \le n\\
&\qquad\qquad \mathbf{then}\ 2\cdot r_0+1\\
&\qquad\qquad \mathbf{else}\ 2\cdot r_0\ \mathbf{fi}\\
&\qquad\qquad \mathbf{fi}\\
&\qquad\qquad \text{since if } n \ne 0,\ n/4 < n
\end{aligned}
$$

This is an efficient recursive function, but why is it correct?

A Theorem that Roots Exist
(Can be Found)

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

Pf by efficient induction

Base $n = 0$ let $r = 0$

Induction case assume $\exists r: \mathbb{N}.$Root (r, n/4)

Choose r_0 **where** $r_0^{\,2} \le n/4 < (r_0 + 1)^2$

note $4 \cdot r_0^{\,2} \le n < 4 \cdot (r_0 + 1)^2 = 4 \cdot r_0^{\,2} + 8 \cdot r_0 + 4$

thus $2 \cdot r_0 \le \text{root}(n) < 2 \cdot (r_0 + 1)$

if $(2 \cdot r_0 + 1)^2 \le n$ **then** $r = 2 \times r + 1$

since $(2 \cdot r_0)^2 = 4 \cdot r_0^{\,2} + 8 \cdot r_0 + 4$

else $r = 2 \times r$ since $(2 \cdot r_0)^2 \le n < (2 \cdot r_0 + 1)^2$

Qed

Correctness of the Recursive Program

Using this “efficient induction principle”.

We can give a nice proof of the principle by ordinary induction.

$$P(0) \;\&\; \forall n{:}\ \mathbb{N}.(P(n/4) \Rightarrow P(n)) \Rightarrow \;\forall n{:}\ \mathbb{N}.P(n)$$