A Nominal Exploration of Intuitionism

Vincent Rahli and Mark Bickford
http://www.nuprl.org

January 19, 2016

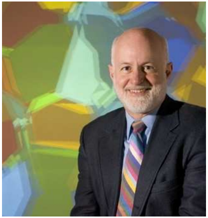
Overall Story

L.E.J. Brouwer

Mark Bickford

Stephen C. Kleene

Robert L. Constable



Nupri in a Nutshell

Similar to Coq and Agda

Extensional Intuitionistic Type Theory for partial functions

Consistency proof in Coq:
`https://github.com/vrahli/NuprlInCoq`

Cloud based & virtual machines: `http://www.nuprl.org`

JonPRL: `http://www.jonprl.org`

Nuprl Stack

Tactics

Refiner

Inference rules

Allen's PER semantics

Howe's computational equality

An untyped applied lambda-calculus

Nuprl Types

Based on Martin-Löf's extensional type theory

Equality: $a = b \in T$

Dependent product: $\mathbf{\Pi} a{:}A.B[a]$

Dependent sum: $\mathbf{\Sigma} a{:}A.B[a]$

Universe: $\mathbb{U}_i$

Nuprl Types

Less "conventional types"

Partial: $\overline{A}$

Disjoint union: $A+B$

Intersection: $\cap a{:}A.B[a]$

Union: $\cup a{:}A.B[a]$

Subset: $\{a : A \mid B[a]\}$

Quotient: $T//E$

Domain: `Base`

Simulation: $t_1 \preceq t_2$

Bisimulation: $t_1 \simeq t_2$

Image: $\mathtt{Img}(A, f)$

PER: $\mathtt{per}(R)$

Nuprl Types

Squashing/Truncation

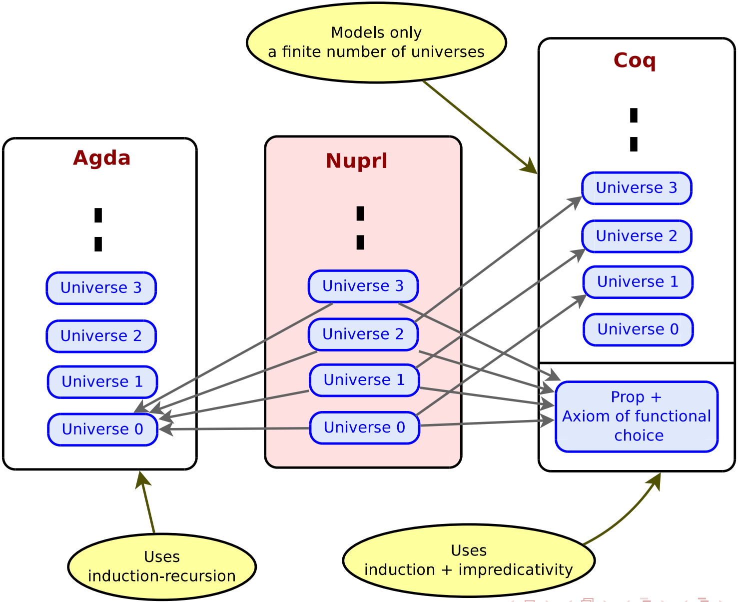
$\downarrow T$ $\{\texttt{Unit} \mid T\}$

$x = y \in \downarrow T$ iff x and y compute to $\star$
and T "is true"
...but we don't remember the reason why

$\downarrow T$ $T//\texttt{True}$

$x = y \in \downarrow T$ iff $x, y \in T$

Nuprl PER Semantics Implemented in Coq



The More Types & Inference Rules the Better!

All verified

Expose more of the metatheory

Encode Mathematical knowledge

Towards Intuitionistic Type Theory

We've proved this rule correct using our Coq model:

Brouwer's Continuity Principle for numbers

$$\boldsymbol{\Pi} F{:}\mathcal{B} \to \mathbb{N}.\boldsymbol{\Pi} f{:}\mathcal{B}.\downarrow\boldsymbol{\Sigma} n{:}\mathbb{N}.\boldsymbol{\Pi} g{:}\mathcal{B}.f =_{\mathcal{B}_n} g \to F(f) =_{\mathbb{N}} F(g)$$

$$(\mathcal{B} = \mathbb{N}^{\mathbb{N}} = \mathbb{N} \to \mathbb{N} \quad \& \quad \mathcal{B}_n = \mathbb{N}^{\mathbb{N}_n} = \mathbb{N}_n \to \mathbb{N})$$

Given a total function F on infinite sequences of numbers, for every sequence f, F only applies f to numbers up to some n.

Weak Continuity

False in Nuprl (Kreisel 62, Troelstra 77, Escardó & Xu 2015)

$$\mathbf{\Pi} F{:}\mathcal{B} \to \mathbb{N}.\mathbf{\Pi} f{:}\mathcal{B}.\mathbf{\Sigma} n{:}\mathbb{N}.\mathbf{\Pi} g{:}\mathcal{B}.f =_{\mathcal{B}_n} g \to F(f) =_{\mathbb{N}} F(g)$$

Easy in Coq model (almost purely by computation) because it doesn't have computational content

$$\mathbf{\Pi} F{:}\mathcal{B} \to \mathbb{N}.\mathbf{\Pi} f{:}\mathcal{B}.{\downarrow}\mathbf{\Sigma} n{:}\mathbb{N}.\mathbf{\Pi} g{:}\mathcal{B}.f =_{\mathcal{B}_n} g \to F(f) =_{\mathbb{N}} F(g)$$

Harder in Coq because it has computational content: uses named exceptions + ν (following Longley's method)

$$\mathbf{\Pi} F{:}\mathcal{B} \to \mathbb{N}.\mathbf{\Pi} f{:}\mathcal{B}.{\downarrow}\mathbf{\Sigma} n{:}\mathbb{N}.\mathbf{\Pi} g{:}\mathcal{B}.f =_{\mathcal{B}_n} g \to F(f) =_{\mathbb{N}} F(g)$$

Strong Continuity

Actually what we proved in Coq is essentially

$$
\begin{array}{l}
\mathbf{\Pi} F{:}\mathcal{B} \to \mathbb{N}. \\
\quad \downharpoonleft \mathbf{\Sigma} M{:}(\mathbf{\Pi} n{:}\mathbb{N}.\mathcal{B}_n \to \mathbb{N}+\mathtt{Unit}). \\
\qquad \mathbf{\Pi} f{:}\mathcal{B}.\mathbf{\Sigma} n{:}\mathbb{N}. \quad M\ n\ f =_{\mathbb{N}+\mathtt{Unit}} \mathtt{inl}(F(f)) \\
\qquad\qquad\qquad\qquad \wedge\, \mathbf{\Pi} m{:}\mathbb{N}.\mathtt{isl}(M\ m\ f) \to m =_{\mathbb{N}} n
\end{array}
$$

Given a total function F on infinite sequences of numbers, there exists a function M, that can tell us for every finite sequence f, whether f is long enough to apply F to f, and if it is, it returns $F(f)$.

For all infinite sequence f, there exists a number n such that applying M to f's initial segment of length n returns $F(f)$.

Strong Continuity

Actually what we proved in Coq is essentially

$$
\begin{array}{l}
\mathbf{\Pi} F{:}\mathcal{B} \to \mathbb{N}. \\
\quad \downharpoonright \mathbf{\Sigma} M{:}(\mathbf{\Pi} n{:}\mathbb{N}.\mathcal{B}_n \to \mathbb{N}+\texttt{Unit}). \\
\qquad \mathbf{\Pi} f{:}\mathcal{B}.\mathbf{\Sigma} n{:}\mathbb{N}. \quad M\ n\ f =_{\mathbb{N}+\texttt{Unit}} \texttt{inl}(F(f)) \\
\qquad\qquad\qquad\qquad \wedge\, \mathbf{\Pi} m{:}\mathbb{N}.\texttt{isl}(M\ m\ f) \to m =_{\mathbb{N}} n
\end{array}
$$

Every function F in $\mathcal{B} \to \mathbb{N}$ has a **neighborhood** function M.

which is equivalent to weak continuity because (standard)

$$\mathrm{AC}_{1,0\downharpoonright} \Rightarrow (\mathrm{WCP}_{\downharpoonright} \iff \mathrm{SCP}_{\downharpoonright})$$

(Digression) Axiom of Choice

Trivial

$$\mathbf{\Pi} a{:}A.\mathbf{\Sigma} b{:}B.P\ a\ b \ \Rightarrow\ \mathbf{\Sigma} f{:}B^{A}.\mathbf{\Pi} a{:}A.P\ a\ f(a)$$

Harder to prove ($AC_{0,0}$) in Coq: uses the axiom of choice and free choice sequences

$$\mathbf{\Pi} a{:}\mathbb{N}.\downarrow\mathbf{\Sigma} b{:}\mathbb{N}.P\ a\ b \ \Rightarrow\ \downarrow\mathbf{\Sigma} f{:}\mathbb{N}^{\mathbb{N}}.\mathbf{\Pi} a{:}\mathbb{N}.P\ a\ f(a)$$

Non-trivial to prove ($AC_{0,n}$ and $AC_{1,n}$) in Nuprl

$$\mathbf{\Pi} a{:}\mathbb{N}.\downarrow\mathbf{\Sigma} b{:}B.P\ a\ b \ \Rightarrow\ \downarrow\mathbf{\Sigma} f{:}B^{\mathbb{N}}.\mathbf{\Pi} a{:}\mathbb{N}.P\ a\ f(a)$$

$$\mathbf{\Pi} a{:}\mathcal{B}.\downarrow\mathbf{\Sigma} b{:}B.P\ a\ b \ \Rightarrow\ \downarrow\mathbf{\Sigma} f{:}B^{\mathcal{B}}.\mathbf{\Pi} a{:}\mathcal{B}.P\ a\ f(a)$$

How to Compute Moduli of Continuity?

$$
\begin{array}{l}
\Pi F{:}\mathcal{B} \to \mathbb{N}.\\
\quad \downarrow \Sigma M{:}(\Pi n{:}\mathbb{N}.\mathcal{B}_n \to \mathbb{N}{+}\texttt{Unit}).\\
\qquad \Pi f{:}\mathcal{B}.\Sigma n{:}\mathbb{N}.\quad M\ n\ f =_{\mathbb{N}+\texttt{Unit}} \texttt{inl}(F(f))\\
\qquad\qquad\qquad\qquad \wedge\ \Pi m{:}\mathbb{N}.\texttt{isl}(M\ m\ f) \to m =_{\mathbb{N}} n
\end{array}
$$

We want to be able to test whether a finite sequence f of length n is long enough. Following Longley's method of using effectful computations:

```
let exception e in
(F (fun x => if x < n then f x else raise e);
 true) handle e => false
```

How to Compute Moduli of Continuity?

```
let exception e in
(F (fun x => if x < n then f x else raise e);
 true) handle e => false
```

We want exceptions & a try/catch operator

F should not be able to catch exception *e*

How to Compute Moduli of Continuity?

```
let exception e in
(F (fun x => if x < n then f x else raise e);
 true) handle e => false
```

We want "unguessable" names

(Have been around in Nuprl for a long time)

If F does not have the name of the exception e then it cannot catch e

We want to be able to generate fresh "unguessable" names

Let's Extend Nuprl With These New Operators

```
let exception e in
(F (fun x => if x < n then f x else raise e);
 true) handle e => false
```

$$
\begin{array}{lll}
v & ::= \cdots \mid \boldsymbol{a} & \text{(name value)} \\
e & ::= \mathtt{exc}(t_1, t_2) & \text{(exception)} \\
vt & ::= \ldots & \\
 & \mid \mathtt{Name} & \text{(name type)} \\
 & \mid \mathtt{Exc}(t_1, t_2) & \text{(exception type)} \\
t & ::= \ldots & \\
 & \mid e & \text{(exception)} \\
 & \mid \mathtt{if}\ \boxed{t_1} = \boxed{t_2}\ \mathtt{then}\ t_3\ \mathtt{else}\ t_4 & \text{(name equaliy)} \\
 & \mid \boldsymbol{\nu} x.\boxed{t} & \text{(fresh)} \\
 & \mid \mathtt{try}_n\ \boxed{t}\ \mathtt{with}\ x.c & \text{(try/catch)}
\end{array}
$$

The way our $\boldsymbol{\nu}$ operator works is similar to Odersky's $\boldsymbol{\nu}$ operator in his $\lambda\nu$-calculus.

Mostly Computational Proof

We've proved that this effectful test function
inhabits the continuity principle

Proof mostly done by computation

For example:

In Coq, we compute an over-approximation of the modulus of continuity of F at f by computing $F(f)$ to a number k, and returning the largest number occurring in the sequence:

$$F(f) \mapsto \cdots \mapsto k$$

This proves that the modulus of continuity of F at f $\downarrow$-exists.

Consistency

We've added these terms to Nuprl's computation system and proved that Nuprl's meta-theoretical properties are preserved

Including:

The congruence of Howe's computational equivalence relation

The validity of Nuprl's inference rules

Questions

Can we prove continuity for sequences of terms instead of $\mathcal{B}$?

Exception mechanism doesn't seem to play well with a parallel operator?

What properties of names and exceptions do we have to make available as inference rules so that we can do the proof directly in Nuprl?