Exercising Nuprl's Open-Endedness

Vincent Rahli

securityandtrust.lu

July 13, 2016

Collaborators — the ABC

Abhishek Anand

Mark Bickford

Robert L. Constable

Nuprl in a Nutshell

Similar to Coq and Agda

Extensional Intuitionistic Type Theory with partial types

Consistency proof in Coq:
`https://github.com/vrahli/NuprlInCoq`
We've proved the validity of about half the rules

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

Motivation

Correctness

Exploring type theory

Consistency of Nuprl relative to the one of Coq

Nuprl serves as a metatheory for Cubical Type Theory
(see Mark's talk)

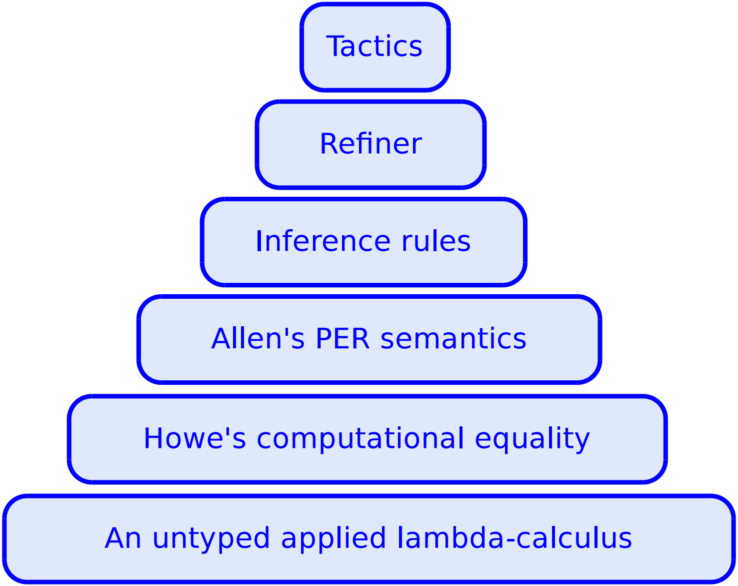
Other Verified Systems

Verified version of HOL light (Myreen et al.)

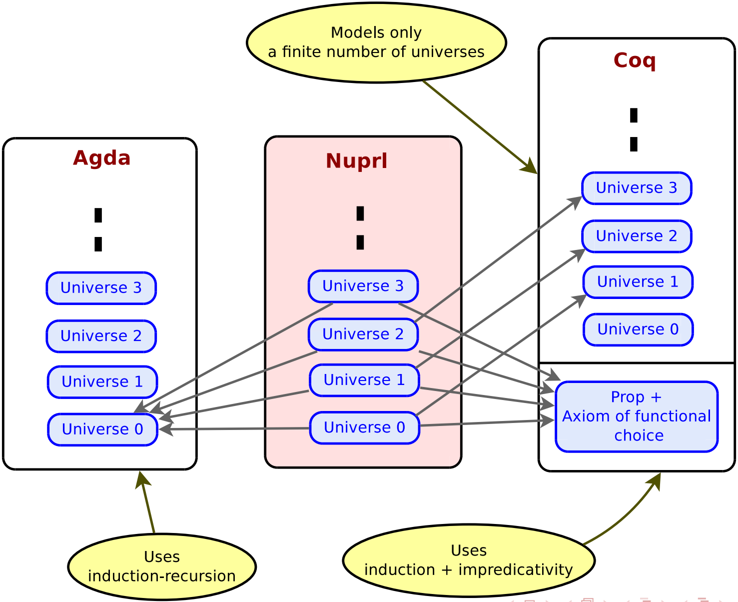
Milawa: Verified ACL2-like prover (Myreen & Davis)

Coq in Coq (Barras)

Nuprl Stack


Tactics
Refiner
Inference rules
Allen's PER semantics
Howe's computational equality
An untyped applied lambda-calculus

Nuprl PER Semantics Implemented in Coq



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: $\mathbb{U}_i$

Nupрl Types

Less "conventional types"

Partial: $\overline{A}$

Disjoint union: $A+B$

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

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

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

Quotient: $T//E$

Domain: $\texttt{Base}$

Simulation: $t_1 \preceq t_2$

Bisimulation: $t_1 \simeq t_2$

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

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

Two Equalities

Propositional/Definitional equality

$$(\lambda x.x + 1) = (\lambda x.1 + x) \in \mathbb{N}^{\mathbb{N}}$$

Howe's computational equivalence relation

$$\mathtt{fix}(\lambda x.x) \simeq \mathtt{fix}(\lambda x.x(x))$$

$$\mathtt{fix}(\lambda x.\langle 0, x\rangle) \simeq \mathtt{fix}(\lambda x.\langle 0, \langle 0, x\rangle\rangle)$$

Allen's PER semantics

Inductive (type equality/$\_{\equiv}\_$)
- Recursive (equality in a type/$\_{\equiv}\_{\in}\_$)

$$x_1{:}A_1 \rightarrow B_1 {\equiv} x_2{:}A_2 \rightarrow B_2$$

$$\begin{array}{l} A_1 {\equiv} A_2 \\ \wedge\ \forall a_1, a_2.\ a_1 {\equiv} a_2 {\in} A_1 \Rightarrow B_1[x_1 \backslash a_1] {\equiv} B_2[x_2 \backslash a_2] \end{array}$$

$$f_1 {\equiv} f_2 {\in} x{:}A \rightarrow B$$

$$\begin{array}{l} \mathtt{type}((x{:}A \rightarrow B)) \\ \wedge\ \forall a_1, a_2.\ a_1 {\equiv} a_2 {\in} A \Rightarrow f_1(a_1) {\equiv} f_2(a_2) {\in} B[x \backslash a_1] \end{array}$$

Allen's PER semantics—Fun Facts

These are Nuprl types:

$$n{:}\mathbb{N} \to \mathbb{U}_n$$

$$\bigcup n{:}\mathbb{N}.\mathbb{U}_n$$

The More Types & Inference Rules the Better!

All verified (our goal)

Expose more of the metatheory

Encode Mathematical knowledge

Open-Endedness

Nuprl was **designed to be open-ended**:
Theorems about computations and types hold for a broad class of extensions to the system

For example:

Howe characterized the computations that can be added to Nuprl in order to preserve the congruence of its computational equivalence relation

Allen characterized the types that can be added to Nuprl in order to preserve the fact that it is a type system

Open-Endedness

New computations:

- named exceptions

$$1 + \mathtt{exc}(\boldsymbol{a}, 0) \mapsto^* \mathtt{exc}(\boldsymbol{a}, 0)$$

- a fresh operator

$$\boldsymbol{\nu} x.\mathtt{if}\ x{=}\boldsymbol{a}\ \mathtt{then}\ 0\ \mathtt{else}\ 1 \mapsto^* 1$$

- choice sequences

$$\mathtt{seq}(\mathtt{f})(1) \mapsto^* \mathtt{f}(1)$$

Where `f` is a Coq function on numbers.

Only used on the metatheory, never get exposed in the theory.

Open-Endedness

New types:

- generalized Π types

$$\bot \in \mathbb{N}^{\texttt{False}}$$

- generalized equality types

$$(\lambda x.x \in B^{A}) \in \texttt{Type}$$

- added PER types

$$T//E = \texttt{per}(\lambda x, y.x \in T \wedge y \in T \wedge (E\ x\ y))$$

Open-Endedness

New inference rules:

- continuity
- bar induction
- **axiom of choice**

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 $x, y \in T$

Axiom of Choice

Trivial

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

Proved in Nuprl (non-trivial): $AC_{0,n}$ and $AC_{1,n}$

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

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

where $\mathcal{B} = \mathbb{N}^{\mathbb{N}}$

Axiom of Choice

Proved in Coq model: $AC_{0,n}$

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

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

where $\mathtt{NBase} = \{t : \mathtt{Base} \mid (t : \mathtt{Base})\#\}$
(name-free members of $\mathtt{Base}$)

Uses the axiom of choice and choice sequences

Axiom of Choice

Negation of $\mathrm{AC}_{2,n}$ in Nuprl:

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

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

Using continuity

Open-ended?

How can we know to what extent Nuprl is open-ended?

Parameterize! Enough?

How can we know we haven't made decisions that unexpectedly limit Nuprl's open-endedness?

By requiring some properties of the computation system.

By adding operators that preclude us from adding others.