# Formal Program Optimization in Nuprl Using Computational Equivalence and Partial Types

**Vincent Rahli**, Mark Bickford, Abhishek Anand

July 25, 2013

# Goals

**Long term goal:** Develop provably correct code.

**Current Goals:**

- Domain specific programming.
- Generate efficient code.

Work done as part of the CRASH project (**Correct-by-Construction Attack-Tolerant Systems**) funded by DARPA (Defense Advanced Research Projects Agency).

# Motivation

➲ **Formal specification, verification, and implementation of asynchronous fault-tolerant systems.**

# Motivation

➲ **Formal specification, verification, and implementation of asynchronous fault-tolerant systems.**

➲ How efficient is our generated code?

# Motivation

- **Formal specification, verification, and implementation of asynchronous fault-tolerant systems.**
- How efficient is our generated code?
- It was not!

# Motivation

- **Formal specification, verification, and implementation of asynchronous fault-tolerant systems.**
- How efficient is our generated code?
- It was not!
- **Formal program optimization in an untyped setting.**
  - More general
  - More efficient

# Nuprl

## Computation System

A **constructive type theory**: CTT13 an evolution of CTT84 closely related to ITT82 [CAB+86, Kre02, ABC+06].

**Untyped**, **deterministic**, **lazy**, applied $\lambda$-calculus with: natural numbers, pairs, injections, fix operator, $\perp$, call-by-value operator,. . . .

# Nuprl

## Computation System

2 meta-relations defined on top of the evaluation function [How96]:

- approximation $\preceq$
- computational equivalence $\sim$ (a congruence).
  $a \sim b \triangleq a \preceq b \land b \preceq a$.

# Nuprl

## Computation System

2 meta-relations defined on top of the evaluation function [How96]:

- approximation $\preceq$
- computational equivalence $\sim$ (a congruence).
  $a \sim b \triangleq a \preceq b \wedge b \preceq a$.

```
CoInductive approx: term -> term -> Prop :=
| approxc : forall t1 t2,
        (forall op terms1,
           computes_to t1 (Value op terms1)
           -> exists terms2,
                computes_to t2 (Value op terms2)
                /\ forall a b, In (a,b) (combine terms1 terms2)
                                   -> approx a b)
        -> approx t1 t2.
```

# Nuprl

## Computation System

For all terms $t$, $\bot \preceq t$.

$\langle \bot, 1 \rangle \preceq \langle 2, 1 \rangle$

$(\lambda x.x + 1)\ 2 \sim 3$.

$\bot \sim \texttt{fix}(\lambda x.x)$.

$\texttt{halts}(t) \triangleq 0 \preceq (\texttt{let}\ x := t\ \texttt{in}\ 0)$

# Nuprl

## Constructive evidence

Type system built on top of the untyped computation system.

A type is a **partial equivalence relation** on $\lambda$-terms [All87a, All87b].

➲ **2 equivalences**: computational and semantic.

Computational semantics: applied $\lambda$-terms provide **evidence** for the truth of propositions.

A sequent $H \vdash C \; \lfloor_{\textbf{ext}} \; t \rfloor$ means that $C$ has computational evidence (extract) $t$ in context $H$.

# Nuprl

## Environment

Distributed.

Runs in the cloud.

Structured editor.

Shared library.

Tactic language: Classic ML.

Replay tool.

# Nuprl

## ITT82 Types

**Equality**: $a = b \in T$
members: $\texttt{Ax}$.

**Dependent function**: $a{:}A \to B[a]$
members: $f$ such that $\forall a \in A,\ f(a) \in B[a]$
(Extensional function equality.)

**Dependent product**: $a : A \times B[a]$
members: $\langle a, b \rangle$

**Disjoint union**: $A + B$
members: $\texttt{inl}(a)$, $\texttt{inr}(b)$

**Universe**: $\mathbb{U}_i$
A hierarchy of universes to avoid Girard's paradox

# Nuprl

## Types

**Subtype**: $A \sqsubseteq B$

**Quotient**: $T//E$

**Intersection**: $\cap a : A.B[a]$

$\star$**Image**: $\mathtt{Img}(T, f)$

Subset: $\{a : A \mid B[a]\} \triangleq \mathtt{Img}(a : A \times B[a], \pi_1)$

Union: $\cup\, a : A.B[a] \triangleq \mathtt{Img}(a : A \times B[a], \pi_2)$

**Recursive type**: $\mathtt{rec}(F)$
where $F$ is a monotone function on types [Men88].

# Nupri

## Types

Constructive domain theory:

**Domain**: `Base`

closed terms of the computation system quotiented by $\sim$

$\star$**Approximation**: $a \preceq b$

members: `Ax`

**Computational equivalence**: $a \sim b$

members: `Ax`

$\star$**Partial types**: $\overline{T}$

contains all members of $T$ as well as all divergent terms

# Nuprl
## Types

$\texttt{True} \triangleq 0 \preceq 0$

$\texttt{Void} \triangleq \texttt{False} \triangleq 0 \preceq 1$

$\texttt{Top} \triangleq \cap a : \texttt{Void}.\texttt{Void}$

$(\texttt{Type}, \sqsubseteq, \cap, \cup, \texttt{Top}, \texttt{Void})$ is a complete bounded lattice.

# Computational equivalence

**A simple example:**

$$\texttt{let}\ x, y = \bot\ \texttt{in}\ x \sim \bot?$$

# Computational equivalence

**A simple example:**

$$\texttt{let } x, y = \bot \texttt{ in } x \sim \bot?$$

They have the same observable behavior.

How can we prove this equivalence?

# Computational equivalence

**A simple example:**

$$\texttt{let}\ x, y = \bot\ \texttt{in}\ x \sim \bot?$$

They have the same observable behavior.

How can we prove this equivalence?

We have to prove:

$$\texttt{let}\ x, y = \bot\ \texttt{in}\ x \preceq \bot$$

$$\bot \preceq \texttt{let}\ x, y = \bot\ \texttt{in}\ x$$

# Computational equivalence

$\bot \preceq \texttt{let } x, y = \bot \texttt{ in } x$ is trivial.

How about:

$$\texttt{let } x, y = \bot \texttt{ in } x \preceq \bot$$

By definition of $\preceq$ we can assume:

$$\texttt{halts}(\texttt{let } x, y = \bot \texttt{ in } x)$$

**We added a rule that says**:

$$\text{if } \texttt{halts}(\texttt{let } x, y = t \texttt{ in } F) \text{ then } t \sim \langle \pi_1(t), \pi_2(t) \rangle$$

(And similarly for all destructors.)

# Computational equivalence

➲ **We added rules to reason about the computation system**

# Computational equivalence

$$\forall t:\texttt{Top}.\ \texttt{map}(f,\texttt{map}(g,t)) \sim \texttt{map}(f \circ g, t)?$$

# Computational equivalence

$$\forall t : \texttt{Top}.\ \texttt{map}(f, \texttt{map}(g, t)) \sim \texttt{map}(f \circ g, t)?$$

$$\begin{aligned}&\texttt{map}(f, t)\\ &= \texttt{fix}\left(\lambda R.\lambda t.\texttt{ispair}\left(\begin{array}{l} t, \\ \texttt{let}\ x, y = t\ \texttt{in}\ (f\ x) \bullet R\ y, \\ \texttt{isaxiom}(t, \texttt{nil}, \bot) \end{array}\right)\right)\ t\end{aligned}$$

$$\texttt{List}(T) = \texttt{rec}(L.\texttt{Unit} \cup T \times L)$$

a list: $\langle 1, \langle 2, \langle 3, \texttt{Ax}\rangle\rangle\rangle$

# Computational equivalence

➲ **We added the following least upper bound property [Cra98]**

$$H \vdash G[\mathtt{fix}(f)/x] \preceq t$$
$$\quad \text{BY } \mathtt{[least\text{-}upper\text{-}bound]}$$
$$\quad H, n : \mathbb{N} \vdash G[f^n(\bot)/x] \preceq t$$

We prove

$$\mathtt{map}(f \circ g, t) \preceq \mathtt{map}(f, \mathtt{map}(g, t))$$

using `[least-upper-bound]` and then by induction on $n$.

# Computational equivalence

In the induction case, we end up with:

$$\texttt{ispair}\begin{pmatrix} t, \\ \texttt{let}\ x, y = t\ \texttt{in}\ (f\ \ x) \bullet R\ \ y, \\ \texttt{isaxiom}(t, \texttt{nil}, \bot) \end{pmatrix} \preceq X$$

➲ **We added the following rule:**

$$\begin{array}{l} H \vdash C\ \lfloor \textbf{ext}\ \texttt{ispair}(t, a, b)[x\backslash\texttt{Ax}]\rfloor \\ \quad \text{BY}\ \texttt{[ispairCases]} \\ \quad H \vdash \texttt{halts}(t) \\ \quad H \vdash t \in \texttt{Base} \\ \quad H, x : t \sim \langle \pi_1(t), \pi_2(t)\rangle \vdash C\ \lfloor \textbf{ext}\ a\rfloor \\ \quad H, x : (\forall[u, v : \texttt{Base}].\ \texttt{ispair}(z, u, v) \sim v)[z\backslash t] \vdash C\ \lfloor \textbf{ext}\ b\rfloor \end{array}$$

# Computational equivalence

Process type:

$$\texttt{corec}(\lambda P.A \to P \times \texttt{Bag}(B))$$

where

$$\texttt{corec}(G) = \cap n : \mathbb{N}.\texttt{fix}\begin{pmatrix}\lambda P.\lambda n.\texttt{if}\ n =_{\mathbb{Z}} 0\ \texttt{then Top} \\ \texttt{else}\ G\ (P\ (n-1))\end{pmatrix}\ n$$

$$P = \texttt{buffer}((\lambda n.\lambda buf.\{n + buf\})\ \texttt{o}\ \texttt{base}(\lambda m.\{m\}), \{0\})$$

$$\Downarrow$$

$$P' = \texttt{fix}(\lambda F.\lambda s.\lambda m.\texttt{let}\ x ::= m + s\ \texttt{in}\ \langle F\ x, \{x\}\rangle)\ 0$$

# Computational equivalence

- $P$ vs. $P'$:
  - 100/200 computation steps for $P$
  - less than 10 computation steps for $P'$

# Computational equivalence

- $P$ vs. $P'$:
  - $100/200$ computation steps for $P$
  - less than 10 computation steps for $P'$

- ShadowDB (replicated database implemented by Nicolas Schiper):
  - non-optimized code: 127 milliseconds
  - optimized code: 60 milliseconds
  - Lisp code: 5 milliseconds
  - reference implementation: 1 millisecond

# Current and future work

- Performance
  - Identify more optimizations.
  - Prove that our optimizations improve the runtime.
- Nuprl
  - Prove that our new types and rules are valid.

# References I


Stuart F. Allen, Mark Bickford, Robert L. Constable, Richard Eaton, Christoph Kreitz, Lori Lorigo, and Evan Moran.
Innovations in computational type theory using Nuprl.
*J. Applied Logic*, 4(4):428–469, 2006.
http://www.nuprl.org/.

Stuart F. Allen.
A non-type-theoretic definition of martin-löf's types.
In *LICS*, pages 215–221. IEEE Computer Society, 1987.

Stuart F. Allen.
*A Non-Type-Theoretic Semantics for Type-Theoretic Language*.
PhD thesis, Cornell University, 1987.

R. L. Constable, S. F. Allen, H. M. Bromley, W. R. Cleaveland, J. F. Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, J. T. Sasaki, and S. F. Smith.
*Implementing mathematics with the Nuprl proof development system*.
Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1986.

Karl Crary.
*Type-Theoretic Methodology for Practical Programming Languages*.
PhD thesis, Cornell University, Ithaca, NY, August 1998.

Douglas J. Howe.
Proving congruence of bisimulation in functional programming languages.
*Inf. Comput.*, 124(2):103–112, 1996.

# References II


Christoph Kreitz.
*The Nuprl Proof Development System, Version 5, Reference Manual and User's Guide.*
Cornell University, Ithaca, NY, 2002.
www.nuprl.org/html/02cucs-NuprlManual.pdf .

P.F. Mendler.
*Inductive Definition in Type Theory.*
PhD thesis, Cornell University, Ithaca, NY, 1988.