Nuprl’s Inductive Logical Forms

Mark Bickford, Robert L. Constable, Rich Eaton, and Vincent Rahli

http://www.nuprl.org

October 7, 2015
Nuprl Environment

- Distributed
- Runs in the cloud
- Structure editor
- Tactic language: Classic ML
- Shared library
- Database based
Getting access to Nuprl:

Virtual Machines: http://www.nuprl.org/vms/

MetaPRL: http://metaprl.org (dead?)

JonPRL: http://www.jonprl.org/
Nuprl Stack

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

\( \lesssim \) is a simulation relation

Greatest fixpoint of the following relation: \( t \ [R] u \) if whenever \( t \) computes to a value \( \theta(\overline{b}) \), then \( u \) also computes to a value \( \theta(\overline{b'}) \) such that \( \overline{b} \ R \overline{b'} \).

\( \sim \) is a bisimulation relation \((a \sim b \equiv a \lesssim b \land b \lesssim a)\)

Purely by computation:

\[
\text{map}(f, \text{map}(g, l)) \sim \text{map}(f \circ g, l)
\]
Howe’s Computational Equality

- Used for automated program optimization
- ≼ and ∼ are congruences
- Restricts the computation system
Howe’s Computational Equality

Type checking and type inference are undecidable

Proving that terms are well-formed can sometimes be cumbersome

Howe’s untyped equality saves us from having to prove well-formedness

It turned out that many equalities could be stated using Howe’s untyped equality
Constructive Domain Theory

Let $\perp$ be $\text{fix}(\lambda x. x)$. 
Constructive Domain Theory

Let $\bot$ be $\text{fix}(\lambda x.x)$.

Least element

$\forall t. \bot \preceq t$
Constructive Domain Theory

Let $\bot$ be $\text{fix}(\lambda x.x)$.

Least element

\[ \forall t. \bot \preceq t \]

Least upper bound principle

$G(\text{fix}(f))$ is the lub of the $\preceq$ chain $G(f^n(\bot))$ for $n \in \mathbb{N}$
Constructive Domain Theory

Let $\bot$ be $\text{fix}(\lambda x.x)$.

**Least element**

$$\forall t. \bot \preceq t$$

**Least upper bound principle**

$G(\text{fix}(f))$ is the lub of the $\preceq$ chain $G(f^n(\bot))$ for $n \in \mathbb{N}$

**Compactness**

if $G(\text{fix}(f))$ converges, then there exists a natural number $n$ such that $G(f^n(\bot))$ converges
Nuprl Types

Based on Martin-Löf’s extensional type theory

Equality: \( a = b \in T \)

Dependent product: \( a : A \to B[a] \)

Dependent sum: \( a : A \times B[a] \)

Universe: \( \mathbb{U}_i \)
Nuprl 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:** $t_1 \sim t_2$

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

**PER:** $\text{per}(R)$
Image type (Nogin & Kopylov)

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

**Union:** \( \bigcup a : A. B[a] \triangleq \text{Img}(a : A \times B[a], \pi_2) \)
PER type

\[ \text{Void} = \text{per}(\lambda _,_1 \preceq 0) \]

\[ \text{Top} = \text{per}(\lambda _,_0 \preceq 0) \]
**Nuprl Types**

**PER type**

\[
\begin{align*}
\text{Void} &= \text{per}(\lambda_,_1 \preceq 0) \\
\text{Top} &= \text{per}(\lambda_,_0 \preceq 0) \\
\text{halts}(t) &= Ax \preceq (\text{let } x := t \text{ in } Ax) \\
A \sqcap B &= \sqcap x:Base. \sqcap y:\text{halts}(x).\text{isaxiom}(x, A, B) \\
T//E &= \text{per}(\lambda x, y. (x \in T) \sqcap (y \in T) \sqcap (E x y))
\end{align*}
\]
Nuprl's proof engine is called a refiner (TB)

A generic goal directed reasoner:

- a rule interpreter
- a proof manager

Example of a rule

\[ H \vdash a: A \rightarrow B[a] \mid_{\text{ext}} \lambda x. b \]

BY [lambdaFormation]

\[ H, x : A \vdash B[x] \mid_{\text{ext}} b \]

\[ H \vdash A \in \mathbb{U}; \mid_{\text{ext}} Ax \]
Stuart Allen had his own meta-theory that was meant to be meaningful on its own and needs not be framed into type theory. We chose to use Coq and Agda.
Intuitionistic Type Theory

We’ve proved these rules correct using our Coq model:

Bar induction

- On free choice sequences of closed terms without atoms
- We can build indexed W types

Brouwer’s Continuity Principle for numbers

\[ \Pi F : B \rightarrow \mathbb{N}. \Pi f : B. \Sigma n : \mathbb{N}. \Pi g : B. f \equiv_{\mathbb{N}^n} g \rightarrow F(f) \equiv_{\mathbb{N}} F(g) \]
Verification of Distributed Systems

Vincent Rahli
Nuprl’s Inductive Logical Forms
October 7, 2015
Verification of Distributed Systems

A logic of events (LoE) and a general process model (GPM) implemented in Nuprl.

Specified, verified, and generated consensus protocols (e.g., 2/3-Consensus & Paxos) using EventML.

**Aneris**: a total ordered broadcast service.

**ShadowDB**: a replicated database with 2 parametrizable replication protocols (PBR & SMR) built on top of Aneris.

Improved performance without introducing bugs. We get decent performance.
Combinators

EventML combinator

generate

Process combinator

generate

Logic of Events combinator

implements
Combinators

EventML for Paxos Synod:

\[
\text{agent } \text{Leader} = \text{SpawnFirstScout} \\
\quad \mid \mid ((\text{LeaderPropose} \mid \mid \text{LeaderAdopted}) \gg \gg \text{Commander}) \\
\quad \mid ((\text{LeaderPreempted} \gg \gg \text{Scout}) ;; \\
\text{main } \text{Leader} \mathbin{\&} \text{Idrs} \mid \mid \text{Acceptor} \mathbin{\&} \text{accpts}
\]
Inductive Logical Forms

We use causal induction + inductive logical forms (ILFs) + state machine invariants

\[
\text{decides (r1,s)} \quad \text{decides (r2,s)} \\
\text{ILF instance} \quad \text{ILF instance} \\
\text{States + Inputs} \\
\text{ILF instance} \\
\ldots \\
\text{Inconsistent states or inputs}
\]

\[
\text{decision on } p \text{ sent to } i \text{ at } e \\
\iff \\
\text{e happens at a leader location} \\
\land \text{decision is triggered by a p2b message} \\
\land \text{decision is sent to replica} \\
\land \text{commander has received a p2b message from a majority of acceptors} \\
\land \text{p comes from a proposal} \\
\lor \text{p comes from an acceptor} \\
\land \ldots
\]
Inductive Logical Forms

E.g., logical explanation of why decisions are made by Paxos:

\[
\forall [\text{Cmd}:\{T:\text{Type}| \text{valueall-type}(T)\}]. \forall [\text{accpts}, \text{ldrsls}: \text{bag}(\text{Id})]. \forall [\text{ldrsls_uid}: \text{Id} \rightarrow \text{Z}]. \forall [\text{reps}: \text{bag}(\text{Id})]. \\
\forall [\text{es}: \text{EO}']. \forall [\text{e}: \text{E}]. \forall [\text{i}: \text{Id}]. \forall [\text{p}: \text{Proposal}]. \\
(\text{decision'send}(\text{Cmd}) \ i \ p \in \text{pax_mb}\_\text{main}(\text{Cmd};\text{accpts};\text{ldrsls};\text{ldrsls_uid};\text{reps})(\text{e})) \Rightarrow \\
\text{decision of } p \text{ sent to } i \text{ at } e \\
\begin{align*}
\text{lo}c(\text{e}) \subseteq \text{ldrsls} & \text{ e happens at a leader location} \\
(\text{header}(\text{e}) = '\text{pax_mb p2b}') & \text{ the decision is triggered by a p2b message} \\
(\text{msgtype}(\text{e}) = \text{P2b}) & \text{ the recipient of the decision message is a replica} \\
i \subseteq \text{reps} \text{ } & \text{ proposal } p \text{ is extracted from a pvalue } z \\
\exists z: \text{PValue} \text{ } & \text{ either pvalue } z \text{ is made from a proposal and current ballot} \\
((\text{header}(\text{e}') = [\text{propose}]) & \text{ or either pvalue } z \text{ received in an adopted message or in leader state} \\
\wedge (\text{msgtype}(\text{e}') = \text{Proposal}) \wedge (\text{msgtype}(\text{e}') = \text{pax_mb}\_\text{AState}(\text{Cmd})) & \text{ this decision is the first output of the commander} \\
((\uparrow (\text{proposal_slot} (\text{proposal_cmd} \text{LeaderStateFun}(\text{e}')))) & \text{ the acceptor that sent the p2b message has accepted pvalue } z \\
\wedge (\uparrow (\text{in_domain} (\text{proposal_slot} \text{msgval}(\text{e}'))) (\text{proposal_cmd} (\text{proposal_cmd} \text{LeaderStateFun}(\text{e}'))))) & \text{ the commander has received a p2b messages from a majority of acceptors} \\
\wedge (z = (\text{mk_pvalue} (\text{proposal_slot} \text{LeaderStateFun}(\text{e}')) \text{msgval}(\text{e}')))) & \\
\wedge (\text{no commander_output}(\text{accpts};\text{reps}) z@\text{Loc} & \\
o (\text{Loc},\text{p2b}'\text{base}(), \text{CommanderState}(\text{accpts}) (\text{pval_balance} z) (\text{proposal_slot} (\text{pval_proposal} z))) & \\
between e' and e) \\
\wedge ((\text{pval_balance} z) = (\text{bl_balance} (\text{p2b_balance} \text{msgval}(\text{e}')))) \\
\wedge ((\text{proposal_slot} (\text{pval_proposal} z)) = (\text{p2b_slot} \text{msgval}(\text{e}'))) \\
\wedge ((\text{pval_balance} z) = (\text{p2b_balance} \text{msgval}(\text{e}')))) & \\
\wedge ((\text{pval_balance} z) = (\text{p2b_balance} \text{msgval}(\text{e}'))) & \\
\wedge ((\text{pval_balance} z) = (\text{p2b_balance} \text{msgval}(\text{e}'))) & \\
\wedge ((\text{pval_balance} z) = (\text{p2b_balance} \text{msgval}(\text{e}'))) & \\
\wedge ((\text{pval_balance} z) = (\text{p2b_balance} \text{msgval}(\text{e}'))) & \\
\wedge (\#(\text{CommanderStateFun}(\text{pval_balance} z;\text{proposal_slot} (\text{pval_proposal} z);\text{es}.e';\text{e})) < \text{threshold}(\text{accpts})) & \\
\wedge (p = (\text{pval_proposal} z)))) \\
\end{align*}
\]
We found bugs using our ILFS

Could be used for blame tracking

Translate to English explanations?