The Nuprl Open Logical Environment

Stuart Allen, Robert Constable, Richard Eaton, Christoph Kreitz, Lori Lorigo

Department of Computer Science, Cornell University

Ithaca, NY 14853

The Nuprl Project

- **Computational Formal Logics**
  - Type Theory $\longmapsto$ logic + programming + data types
  - Meta-reasoning, reflection $\longmapsto$ extensibility, stability
  - Relating different logics
- **Proof / Program Development Systems**
  - The Nuprl open Logical Programming Environment $\longmapsto$ interoperability
  - The MetaPRL inference engine $\longmapsto$ speed, modularity
  - Proof search techniques $\longmapsto$ proof automation
  - Natural language generation from formal math $\longmapsto$ comprehensibility
  - Automated complexity analysis $\longmapsto$ efficient results
- **Applications:**
  - Formal mathematical textbook
  - Hardware verification
  - System verification and optimization

The NuPRL Type Theory
An Instance of Martin-Löf Type Theory

- **Constructive Higher-Order Logic**
  – Reasoning about types, members of types, propositions, functions ...
- **Functional Programming Language**
  – Polymorphic, partial recursive functions
  – Similar to core ML
- **Expressive Data Type System**
  – Function, Product, Disjoint Union, Π- & Σ-types, Void, Top
  – Integers, Atoms, Lists, Inductive Types
  – Subsets, Subtyping, Quotient Types
  – (dependent) Intersection, Union, Records
  – Equality Type, Propositions as types, Universes
- **Open-ended**
  – New types can be added

Terms of Nuprl's Type Theory

Function Space	$S \rightarrow T$, $x{:}S \rightarrow T$	$\lambda x.t$, $f\ t$
Product Space	$S \times T$, $x{:}S \times T$	$\langle s, t \rangle$, let $\langle x, y \rangle = e$ in u
Disjoint Union	$S + T$	inl(s), inr(t), case e of inl(x) $\mapsto u$ | inr(y) $\mapsto v$
Numbers	$\mathbb{Z}$	0,1,-1,2,-2,..., s+t, s-t, s*t, s/t, if a=b then s else t, *+ induction*
Atoms	Atom	"*token*", if a=b then s else t
Lists	S list	[], t::*list*, *+ induction*
Subset	$\{x{:}S \mid P[x]\}$	*— some members of S —*
Intersection	$\cap x{:}S.T[x]$	*— members that occur in all $T[x]$ —*
Union	$\cup x{:}S.T[x]$	*— members that occur in some $T[x]$ (consitency?) —*
Quotient	$x, y : S//E[x, y]$	*— members of S, new equality —*
Inductive Types	rectype $x = S[x]$	*— members of S, recursively unrolling —*
Empty Type	void	*— no members —*
Equality	$s = t \in T$	Ax
Universes	$\mathbb{U}_j$	*— types of level j —*
⋮	⋮	

Distinguishing Features of Nuprl's Type Theory

- **Uniform Internal Notation**
  - No syntactical distinction between types, members, propositions ...
  - Independent term display allows "free syntax" $\leadsto$ display forms
  - User-defined extensions possible $\leadsto$ abstractions
- **Separation between Expressions and Types**
  - No restriction on expressions that can be defined $\leadsto$ Y combinator
  - Expressions in proofs must be (top-level) typeable $\leadsto$ "total" functions
- **Refinement Calculus**
  - Top-down Sequent Calculus $\leadsto$ interactive proof development
  - Proof expressions linked to inference rules $\leadsto$ program extraction
  - Computation rules
  - User-defined inference rules $\leadsto$ tactics

The Nuprl System

- **Beginnings in 1984**
  - Nuprl 1 (Symbolics): proof & program refinement in Type Theory
  - Book: *Implementing Mathematics . . .* (1986)
  - Nuprl 2: Unix Version
- **Nuprl 3: Mathematical Problem Solving**
  - Machine proof for unsolved problems (Girard's paradox) (Howe 1987)
    (Higman's Lemma) (Murthy 1990)
- **Nuprl 4: System Verification and Optimization**
  - Verification of a logic synthesis tool (Aagaard & Leeser 1993)
  - Verification of the SCI cache coherency protocol (Howe 1996)
  - Optimization of the Ensemble group communication system (Kreitz, Hayden & Hickey 1999)
  - Verification of Ensemble protocol layers (Bickford 1999)

Nuprl 4 System Features

- **Interactive Proof Editor** ⇝ readable proofs
- **Flexible definition mechanism** ⇝ user-defined terms
- **Customizable Term Display** ⇝ flexible notation
- **Structure Editor for Terms** ⇝ no ambiguities
- **Tactics** ⇝ user-defined inferences
- **Decision Procedures**
- **Proof objects, Program Extraction** ⇝ program synthesis
- **Program Evaluation**
- **Library mechanism** ⇝ user-theories
  - Large mathematical libraries
  - Large tactics collection
- **HTML output generator** ⇝ web accessibility

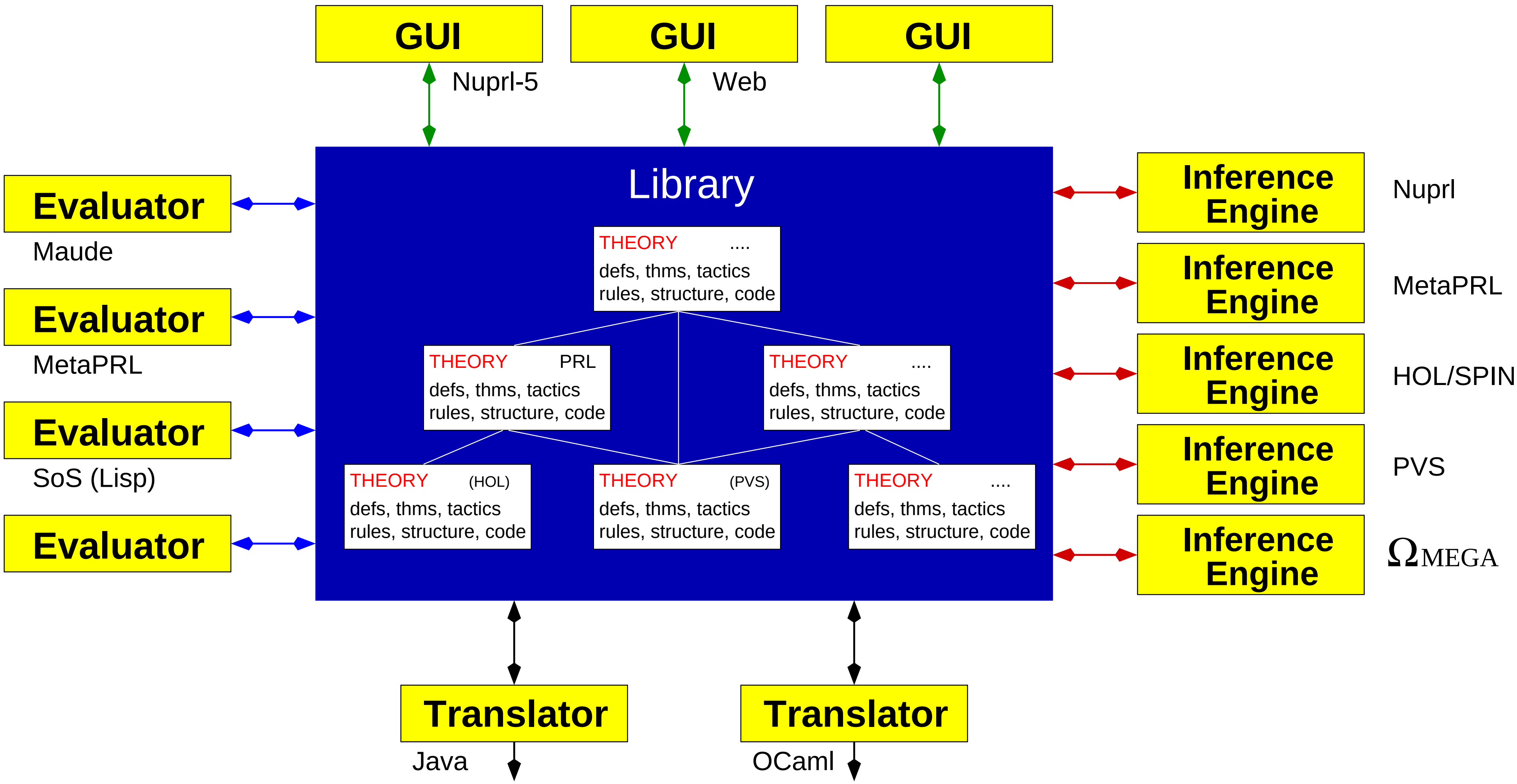
The Next Generation: An Open Logical Environment

Closed formal systems are not ready for future demands

- **Platform for Cooperating Reasoning Systems**
  - Proof assistants
  - Decision procedures
  - Fully automatic theorem provers
  - Proof planners
  - Rewrite engines
  - Model checkers
  - Computer Algebra systems

- **Nuprl 5 Design Objectives**
  - Interoperability
  - Optimization of system productivity
  - Accountability
  - Information preservation
  - Large scale object management

The Nuprl 5 Architecture



Key Features I

- **Collection of Cooperating Processes**
  - Centered around a common knowledge base
  - Refiners, interfaces, evaluators, etc. connect as independent processes
  - Processes can connect and disconnect at any time
  - Several users can work in parallel on the same formal theory
  - A user can start several refiners in parallel

- **Ability to Connect to External Systems**
  - MetaPRL (modularized PRL, multiple logics) (Hickey & Nogin, 1999)
  - Jprover (a matrix-based intuitionistic theorem prover) (Schmitt & Lorigo, 2000)
  - HOL (classical higher order logic) (Howe, 1998, Stehr & Naumov, 1999)
  - Mathematica (Benzinger, 2000)

⋮

Key Features II

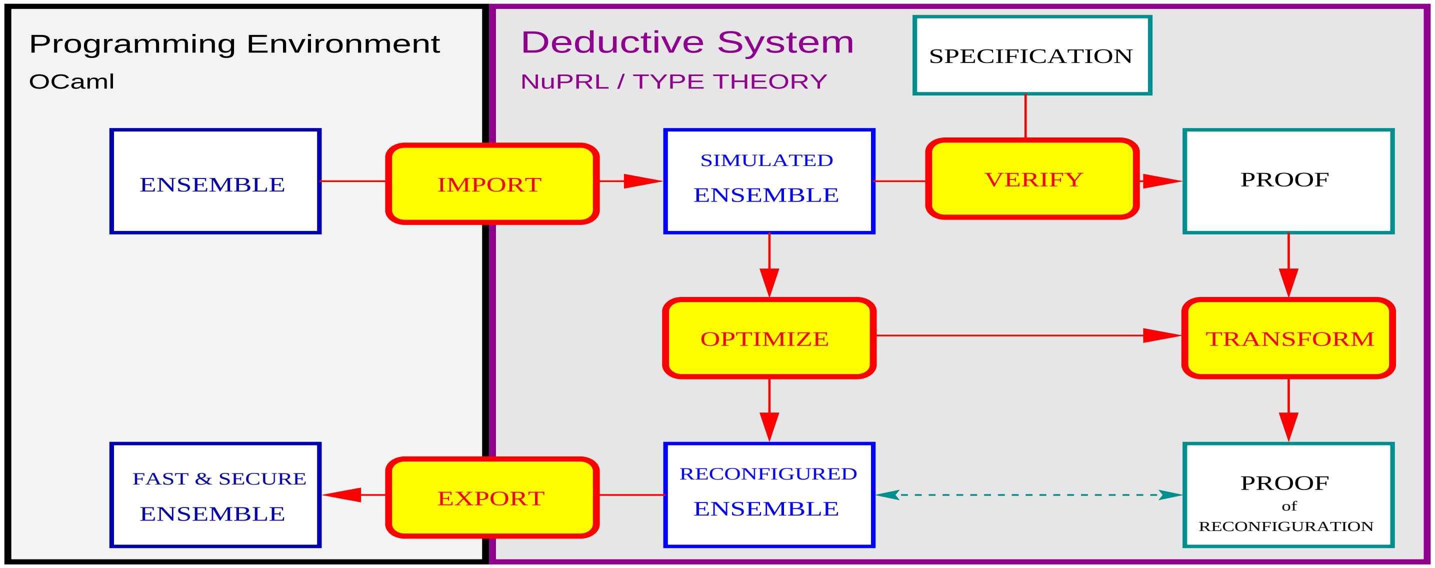
- **Library Organized as Persistent Data Base**
  - Transaction model (preserves data even in case of crashes)
  - Version control mechanism
  - Dependency tracking
- **Reflective System Structure**
  - System designed within the system's library
  - Customizable structure
- **Cooperating Inference Engines**
  - Asynchronous refinement
  - Distributed theorem proving
- **Multiple User Interfaces**
  - Structure editor for proofs, terms, and library navigation
  - Collaborative proving while using favorite editor
  - Web front end will allow external users to browse the library

\+ **Preservation of Nuprl 4 Capabilities and Libraries**

Applications of NuPRL

- **Mathematics**
  - Number theory, real analysis, calculus
  - Group theory, algebra
  - Automata theory, computing theory
  - Formal mathematical textbook (Constable, Allen 1999)
- **Hardware Verification**
  - A logic synthesis tool (Aagaard & Leeser 1993)
  - SCI cache coherency protocol (Howe 1996)
- **Program Verification, Synthesis, and Optimization**
  - Synthesis of elementary algorithms: square-root, sorting, ...
  - Programming semantics & complexity analysis (Benzinger, 2000)
  - Optimization and Verification of the Ensemble group communication system (Kreitz, Bickford, Hayden, Hickey, Liu, van Renessee 1998–)

Application: Building Reliable, High-Performance Systems



- **Apply Formal Tools to Real Code**
  - Type-theoretical semantics for OCAML subset
  - Automatic import and export of OCAML-code into NUPRL
- **Reliability**
  - I/O Automata model of Ensemble protocol specification
  - Verification of total order protocol helped detect and fix subtle bug
- **High-Performance**
  - Auotmatic optimization of Ensemble for common execution paths
  - Performance improved by factor 3
  - Guarantee for same functionality

Conclusion

- **Open Proof Environments are the way of the future**
  - No individual system is strong enough
  - Many systems now connect external inference engines
- **Nuprl 5 goes one step further**
  - Cooperating systems
  - Support for joint formal theories
- **Future Plans:**