CS 671 Automated Reasoning

Introduction to NuPRL

1. NuPRL Features
2. NuPRL Architecture
3. Interactive Theorem Proving in NuPRL
   - Decomposition & Computation
   - Defining new constructs
THE NUPRL SYSTEM

● Beginnings in 1984
  – Nuprl 1 (Symbolics): proof & program refinement in Type Theory
  – Book: Implementing Mathematics ...
  – 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
Nuprl 5: An Open Logical Environment

Platform for Cooperating Reasoning Systems
Nuprl 5: **Additional System Features**

- **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

- **Ability to Connect to External Systems**
  - MetaPRL, JProver, HOL, ... 

- **Library Organized as Persistent Data Base**
  - Transaction model + Version control + Dependency tracking

- **Reflective System Structure**
  - System designed within the system’s library \(\sim\) customizable structure

- **Cooperating Inference Engines**
  - Asynchronous and distributed theorem proving

- **Multiple User Interfaces**
  - Structure editor, Web front end, ...
Refinement Rules for Natural Numbers

\[ H \vdash \mathbb{N} \text{ type} \]

\[ H \vdash 0 = 0 \in \mathbb{N} \]

\[ H \vdash \text{suc}(e) = \text{suc}(e') \in \mathbb{N} \]

\[ H \vdash e = e' \in \mathbb{N} \]

\[ H \vdash \text{ind}(e; \text{base}; n, x. \text{up}) = \text{ind}(e'; \text{base}'; n', x'. \text{up'}) \in T \]

\[ H_1, x : T, H_2 \vdash x = x \in T \]

\[ H \vdash \text{ind}(0; \text{base}; n, x. \text{up}) = e' \in T \]

\[ H \vdash \text{base} = e' \in T \]

\[ H \vdash \text{ind}(\text{suc}(e); \text{base}; n, x. \text{up}) = e' \in T \]

\[ H \vdash \text{up}[e, \text{ind}(e; \text{base}; x.n, \text{up}) / n, x] = e' \in T \]
Refinement Rules for Function Spaces

\[ H \vdash S \rightarrow T \text{ type} \]  \hspace{1cm} \text{funR}

\[ H \vdash S \text{ type} \]
\[ H \vdash T \text{ type} \]

\[ H \vdash \lambda x. e = \lambda x'. e' \in S \rightarrow T \]  \hspace{1cm} \text{lamR}

\[ H, x : S \vdash e = e'[x/x'] \in T \]
\[ H \vdash S \text{ type} \]

\[ H \vdash f e = f' e' \in T \]  \hspace{1cm} \text{appR } S \rightarrow T

\[ H \vdash f = f' \in S \rightarrow T \]
\[ H \vdash e = e' \in S \]

\[ H \vdash (\lambda x. e) e' = e^* \in T \]  \hspace{1cm} \text{compute 1}

\[ H \vdash e'[e/x] = e^* \in T \]

Note: \( e = e \in T \) is usually abbreviated by \( e \in T \)
## Refinement Rules for First-Order Logic

<table>
<thead>
<tr>
<th>left</th>
<th>right</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>andL i</strong></td>
<td>( H, A \land B, H' \vdash G )</td>
</tr>
<tr>
<td></td>
<td>( H, A, B, H' \vdash G )</td>
</tr>
<tr>
<td><strong>orL i</strong></td>
<td>( H, A \lor B, H' \vdash G )</td>
</tr>
<tr>
<td></td>
<td>( H, A, H' \vdash G )</td>
</tr>
<tr>
<td><strong>impL i</strong></td>
<td>( H, A \implies B, H' \vdash G )</td>
</tr>
<tr>
<td></td>
<td>( H, A \implies B, H' \vdash A )</td>
</tr>
<tr>
<td><strong>notL i</strong></td>
<td>( H, \neg A, H' \vdash G )</td>
</tr>
<tr>
<td></td>
<td>( H, \neg A, H' \vdash A )</td>
</tr>
<tr>
<td><strong>exL i</strong></td>
<td>( H, \exists x : T . B, H' \vdash G )</td>
</tr>
<tr>
<td></td>
<td>( H, x : T, B, H' \vdash G )</td>
</tr>
<tr>
<td><strong>allL i t</strong></td>
<td>( H, \forall x : T . B, H' \vdash G )</td>
</tr>
<tr>
<td></td>
<td>( H, \forall x : T . B, B[t/x], H' \vdash G )</td>
</tr>
</tbody>
</table>

*Note: an unlabelled hypotheses A is an abbreviation for \( \% : A \)*