Constructive Proofs and Program Extraction

Christoph Kreitz

1. Type Theory vs. Set Theory
2. Overview of the Nuprl System
3. Proofs of the Integer Square Root Problem

What distinguishes Type Theory from Set Theory?

What is the meaning of $\forall n\, \exists r\;\; r^2 \leq n \,\wedge\, n < (r+1)^2$

- **Set Theory**
  - Integer square roots exist for all natural numbers
  - A proof shows that the notion $\lfloor\sqrt{n}\rfloor$ is well-defined
  - Different proofs lead to the same insight
- **Type Theory**
  - There is a method to construct integer square roots
  - Algorithms for computing $\lfloor\sqrt{n}\rfloor$ can be extracted from proofs
  - Different proofs lead to different algorithms

Proofs for $\forall n\, \exists r\; r^2 \leq n \wedge n < (r+1)^2$

- **Nonconstructive**
  - Assume r wouldn't exist for some n
  - Then for all r: $r^2 > n$ or $(r+1)^2 \leq n$
  - Consider $r_1 = \min\{r | r^2 > n\}$-1
  - Then $r_1^2 \leq n$ and $(r_1+1)^2 > n$ – a contradiction

  Unnecessary indirect approach - it masks a constructive argument

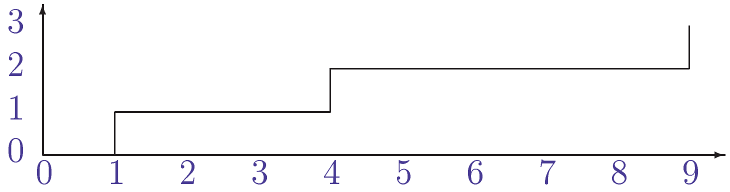
- **Mathematical rigor: Induction on n**
  - Base Case: prove $\exists r\; r^2 \leq 0 \wedge 0 < (r+1)^2$

    Choose $r = 0$, prove $0^2 \leq 0 \wedge 0 < (0+1)^2$ using standard arithmetic
  - Step case: assume $\exists r_n\; r^2 \leq n \wedge n < (r_n+1)^2$ and

    prove $\exists r\; r^2 \leq n+1 \wedge n+1 < (r+1)^2$
  - If $(r_n+1)^2 \leq n+1$

    then choose $r = r_n+1$

    otherwise choose $r = r_n$



  - Proof obligation follow using standard arithmetic
  - Proof leads to algorithm that constructs $\lfloor \sqrt{n} \rfloor$ inductively

How to extract algorithms from proofs?

- **Use formal logic to express proof**
  - First-Order Logic + Induction + Basic Arithmetic $\subseteq$ **Type Theory**
  - Proof rules tie proof steps to algorithm fragments
- **Use computerized proof assistant to formalize proof**
  - **Nuprl** proof development system supports formal proofs in type theory
  - Use proof tactics to keep formalization “simple”
- **Extract algorithm from computerized proof**
  - Nuprl composes algorithm fragments of rules used in proof
  - Algorithm can be executed in Nuprl

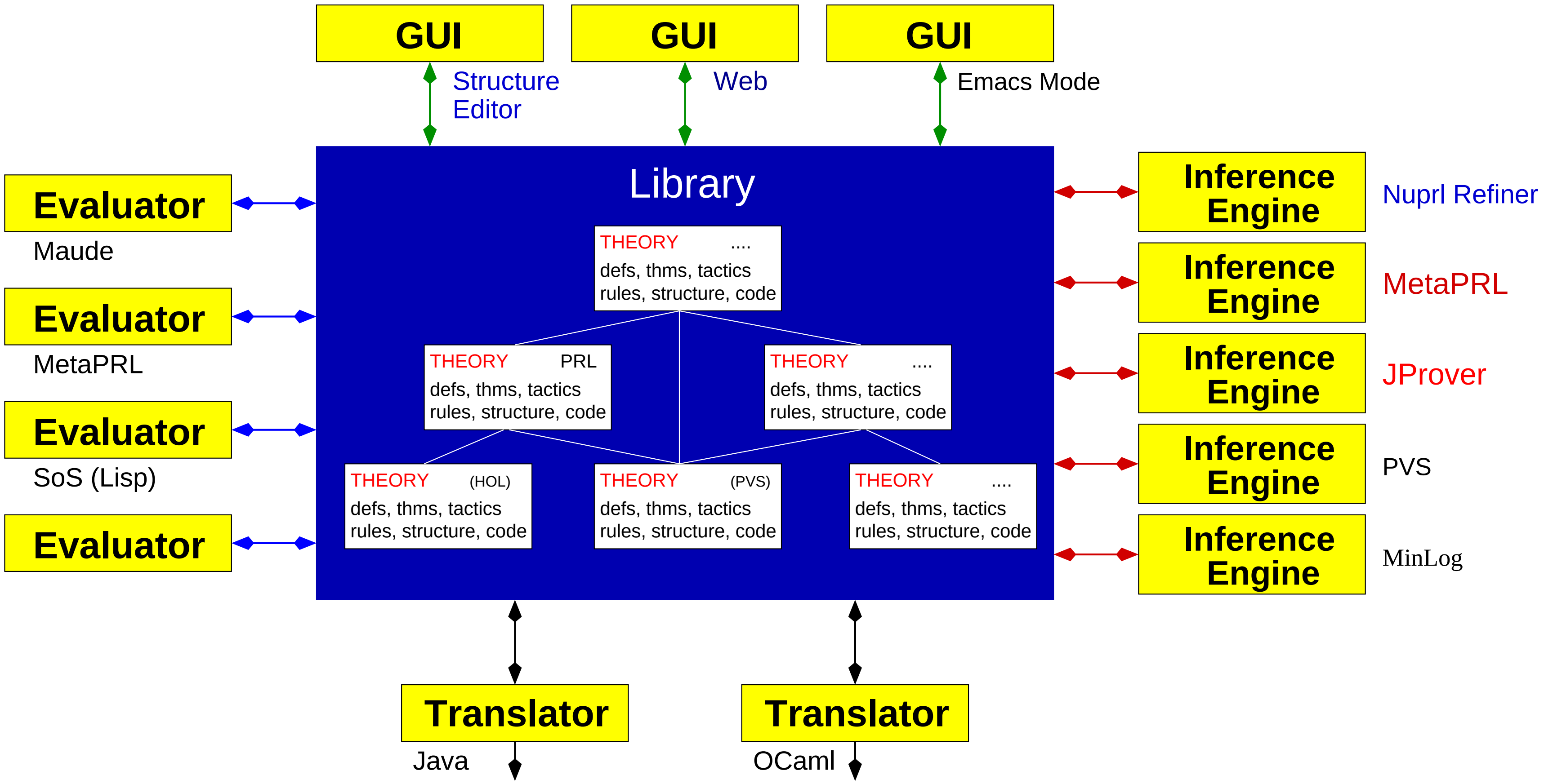
The Nuprl System

Proof & program refinement in Type Theory

- **Interactive Proof Editor** $\rightsquigarrow$ readable proofs
- **Proof Tactics** $\rightsquigarrow$ user-defined inferences
- **Decision Procedures** $\rightsquigarrow$ proof automation
- **Flexible definition mechanism** $\rightsquigarrow$ user-defined terms
- **Customizable Term Display** $\rightsquigarrow$ flexible notation
- **Structure Editor for Terms** $\rightsquigarrow$ no ambiguities
- **Library mechanism** $\rightsquigarrow$ user-theories
  - Large mathematical libraries and tactic collections
- **Program Extraction and Evaluation** $\rightsquigarrow$ program synthesis

Nuprl: System Architecture

A Platform for Cooperating Reasoning Systems



Basic System uses Library, Editor, and Nuprl Refiner

Tactics: User-defined inference rules

- **Meta-level programs built using**
  - Basic inference rules, standard tactics, predefined tacticals
  - Meta-level analysis of the proof goal and its context

  $\mapsto$ Applying a tactic always results in a valid proof

- **Basic Tactics** $\mapsto$ *Nuprl Manual*
  - `Hypothesis`: Prove $\ldots C \ldots \vdash\ C'$ where C' α-equal to C
  - `D` c: Decompose the outermost connective of clause c
  - `EqD` c: Decompose immediate subterms of an equality in clause c
  - `EqTypeD` c: Decompose type subterm of an equality in clause c
  - `Assert` t: Assert (or cut) term t as last hypothesis
  - `Auto`: Apply trivial reasoning, decomposition, decision procedures

  also rules tailored for: Logic, Induction, ...

Formal proof of Integer Square Root Theorem

```
∀n:ℕ. ∃r:ℕ. r² ≤ n < (r+1)²
BY  allR

  n:ℕ
  ⊢ ∃r:ℕ. r² ≤ n < (r+1)²
  BY  NatInd 1

  .....basecase.....
      ⊢ ∃r:ℕ. r² ≤ 0 < (r+1)²
  √  BY  existsR ⌈0⌉ THEN Auto

  .....upcase.....
      i:ℕ⁺,  r:ℕ,  r² ≤ i-1 < (r+1)²
      ⊢ ∃r:ℕ. r² ≤ i < (r+1)²
      BY  Decide ⌈(r+1)² ≤ i⌉ THEN Auto

      .....Case 1.....
          i:ℕ⁺,  r:ℕ,  r² ≤ i-1 < (r+1)²,  (r+1)² ≤ i
          ⊢ ∃r:ℕ. r² ≤ i < (r+1)²
      √  BY  existsR ⌈r+1⌉ THEN Auto'

      .....Case 2.....
          i:ℕ⁺,  r:ℕ,  r² ≤ i-1 < (r+1)²,  ¬((r+1)² ≤ i)
          ⊢ ∃r:ℕ. r² ≤ i < (r+1)²
      √  BY  existsR ⌈r⌉ THEN Auto
```

ALGORITHM EXTRACTED FROM THE PROOF

- **In raw Type Theory**

```
let rec sqrt i
= if i=0 then <0,pf_0>
  else let <r,pf_{i-1}>  = sqrt (i-1)
       in
         if (r+1)^2≤n  then <r+1,pf_i>
         else <r,pf_i'>
```

- **In SML notation** (after stripping proof components)

```
fun sqrt n = if n=0 then 0
             else let val r = sqrt (n-1)
                  in
                    if n<(r+1)^2  then r
                    else r+1
                  end
```

Are there better proofs?

- **Mathematically**
  - Proof is short and "elegant" – why change it?

- **Computationally**
  - Extracted algorithm for $\lfloor\sqrt{n}\rfloor$ is linear in size of input n $\boldsymbol{\mathcal{O}(n)}$

    Proof uses standard induction on n

    $\forall$P:$\mathbb{N}\to\mathbb{P}$. (P(0) $\wedge$ ($\forall$i:$\mathbb{N}^+$. P(i-1) $\Rightarrow$ P(i))) $\Rightarrow$ ($\forall$i:$\mathbb{N}$. P(i))
  - A better algorithm would increase r until $(r{+}1)^2 > n$ $\boldsymbol{\mathcal{O}(\sqrt{n})}$

    Corresponding proof needs schema for bounded search

    $\forall$P:$\mathbb{N}\to\mathbb{P}$. $\forall$n:$\mathbb{N}$. P(n) $\Rightarrow$ ($\exists$k:{0..n}. P(k) $\wedge$ ($\forall$j:{0..k-1}. $\neg$P(j)))
  - An even better algorithm computes $\lfloor\sqrt{n}\rfloor$ bit for bit $\boldsymbol{\mathcal{O}(\log_2 n)}$

    Proof almost identical to first one, but needs 4-adic induction

    $\forall$P:$\mathbb{N}\to\mathbb{P}$. (P(0) $\wedge$ ($\forall$i:$\mathbb{N}$. P(i$\div$4) $\Rightarrow$ P(i))) $\Rightarrow$ ($\forall$i:$\mathbb{N}$. P(i))