Representing Nuprl Proof Objects in ACL2:
towards a Nuprl Proof Checker

James Caldwell
John Cowles

Department of Computer Science
University of Wyoming
Laramie, Wyoming

8 April 2002

ACL2-2002
Third International Workshop on the ACL2 Theorem Prover and Its Applications
April 8 - 9, 2002
Grenoble, France
Introduction

□ (Philosophy) Guarantees on absolute correctness are unattainable, but increased confidence in correctness is justified by accounting and verification.

□ (Goal) To provide independent certification of proof objects produced by the Nuprl system.

□ (Approach) A proof checker for primitive Nuprl proofs in ACL2.
  ○ Primitive Nuprl proofs are output from within Nuprl in a form readable by ACL2.
  ○ These representations of Nuprl proofs read into ACL2 and are used to synthesize a term of the ACL2 logic is-proof.
  ○ The is-proof term is submitted to the ACL2 prover.

□ (Implementation) We are currently able to check gross structural properties of Nuprl proofs.
Core Nuprl

The core Nuprl type theory is a sequent presentation of a constructive type theory via type assignment rules. The Nuprl system implements a rich open environment to support reasoning about and computing with the Nuprl type theory.

- A term language – an untyped functional programming language with constructs denoting types and propositions.
- A computation system (an evaluator for the untyped lambda calculus – lazy evaluation.)
- Type assignment rules for proving terms inhabit types.
- An encoding of logic via the propositions-as-types encoding.
- An LCF style (tactic based) sequent prover, ML is the meta-language.
- A mechanism for extracting programs (λ-terms) from proofs (an implementation of the proofs-as-programs interpretation.).
- Term display is independent from term structure.
Nuprl Proof Rules

- Two classes of rules.
  - Uniform Rules – instances can be checked by a combination of pattern matching and well-formedness checks.
  - Nonuniform rules – they are instantiated in Nuprl by calls to the underlying Lisp implementation. These include decision procedures for linear arithmetic and equality reasoning.
- 152 Uniform rules, 15 nonuniform rules.

- Proof rules schemas are stored in the library as Nuprl terms.
A sample rule schema

* RULE functionExtensionality
  H ⊢ f = g ∈ x:A → B ext t
    BY functionExtensionality level{i} (y:C → D) (z:E → F) u
  H, u:A ⊢ f(u) = g(u) ∈ B[u/x] ext t
  H ⊢ A ∈ ℰ{i}
  H ⊢ f ∈ y:C → D
  H ⊢ g ∈ z:E → F

The name of the rule is functionExtensionality. The goal pattern is specified by the sequent H ⊢ f = g ∈ x:A → B. Here H is a hypothesis list meta-variable, f, g, A, and B are term meta-variables and x is variable meta-variable. The parameters to the rule include a universe level parameter level{i}, two term patterns (y:C → D) and (z:E → F) and a variable meta-variable u. There are four subgoals specified in terms of the meta-variables which appear in the goal pattern and the rule parameters.
Nuprl Proofs

- Nuprl proofs are interactively viewed and created by users via the tactic mechanism. A successful application of a tactic generates a partial proof tree.

- Proofs are a modified form of pure sequent proofs that allow views of the proof at both the tactic level and at the level of primitive rule applications.

  - At a proof node refined by a primitive rule, there are as many child nodes as there are premises of the rule.
  
  - At a proof node refined by a tactic:

    * The leftmost subtree is the (partial) proof tree generated by the tactic.

    * For each incomplete leaf of the left subtree, there is another child whose goal sequent matches the sequent on the incomplete branch.

- We use an addressing scheme to represent the structure of the proof.
Proof Checking

- Verifying the gross structure of the proof.
  - rule refined nodes
    - check the instance against the rule schema to verify the number of children.
  - tactic refined nodes:
    - Find a permutation matching sequents of unproved leaves of the left subtree with the roots of the n-but-first subtrees of the node.
    - Recursively check the left subtree.

- More refined checking using is-proof.
  - An ACL2 term to be submitted to the prover.
  - A conjunction of is-instance predicates, one for each rule refined node.
  - For instances of Arith, the ACL2 arithmetic procedures will be used to verify the correctness of the Nuprl decision procedure.
What if an error is found?

- Would motivate further investigations.
- Could be in Nuprl or in the checking system.
  - Nuprl – many possibilities – but the failure will provide information as to where to look.
  - Checker – error in the translation, error in the checking code, error in ACL2 itself.
- The checker serves to check ACL2 as well as Nuprl.
Conclusions and Future Work

□ Conclusions

○ Each new check adds to the confidence that Nuprl proofs are what the claim to be.
○ Accrued evidence supports an inductive (not mathematical) argument for the correctness of the Nuprl methods.

□ Future Work

○ Open-ended – in the sense that new criteria for correctness might be posed at any time – either about the checker itself or about the Nuprl proof objects.
○ Implement is-instance and is-proof.
○ Eliminate instances of Arith using Joe Smith’s (Wyoming Ph.D. student) new methods for arithmetic reasoning based on standard axioms.

□ Thanks

○ This work is supported by ONR grant N00014-01-1-0765.
○ We thank Ralph Wachter at ONR for his support of this project and Stuart Allen and Bob Constable at Cornell for their insights and suggestions.