
Steve Roach       Fares Fraij
Department of Computer Science
The University of Texas at El Paso

Fifth International Workshop on the ACL2 Theorem Prover and Its Applications (ACL2-2004)

November 18, 2004
Goal

Develop models and techniques using ACL2 to prove the correctness of HATS transformation rules and apply them to a high-consequence system.
Formal Approaches for Software Assurance

<table>
<thead>
<tr>
<th>Approach</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformation-Oriented Programming (TOP)</td>
<td>Incremental refinement of formal specifications to implementations</td>
<td>HATS, Maude, ELAN, Stratego, and ASF+SDF</td>
</tr>
<tr>
<td></td>
<td>- Correctness by construction</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Examples: <strong>HATS</strong>, Maude, ELAN, Stratego, and ASF+SDF</td>
<td></td>
</tr>
<tr>
<td>Automated theorem provers</td>
<td>Model computing systems and their desired properties in the language of the theorem prover and prove the correctness of these properties using inference rules, axioms, and theorems</td>
<td><strong>ACL2</strong>, HOL, PVS, Isabelle</td>
</tr>
<tr>
<td></td>
<td>- Correctness by verification</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Examples: <strong>ACL2</strong>, HOL, PVS, Isabelle</td>
<td></td>
</tr>
</tbody>
</table>
HATS Goals

- Create a language-independent program transformation system
- Perform program transformation in a provably correct fashion
- Provide framework for experimenting with transformation techniques
HATS High-Level Overview

- Transforms input programs written in abstract languages to output programs in concrete languages.
- Transformation language program (TLP) consists of sequence of transformation rules and a control strategy.
### HATS Transformation Language Program

<table>
<thead>
<tr>
<th>Transformation rules</th>
<th>Combinators</th>
<th>Control strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>• General form</td>
<td>• Types:</td>
<td>Control the application of transformation rules to the input file</td>
</tr>
<tr>
<td></td>
<td>– Seq (;)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Left-biased (&lt;+)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Right-biased (-&gt;)</td>
<td></td>
</tr>
<tr>
<td>• Two types of</td>
<td></td>
<td></td>
</tr>
<tr>
<td>transformation rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>– First-Order</td>
<td></td>
<td></td>
</tr>
<tr>
<td>– High-Order</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Combinators:**

- Seq (;)
- Left-biased (<+)
- Right-biased (->)

**Control strategies:**

- Once
- Fix
- Transient
- Hide
Example: Once VS. Fix

Given the following table, $T$, the goal is to resolve the pointers in the second column to their respective string values.

$$T = \begin{pmatrix} (1 \text{ "Hello"}) \\ (2 \text{ "World"}) \\ (3 2) \\ (4 3) \end{pmatrix}$$

To resolve the pointers in the table $T$, the following first-order transformation rules are needed:

- $TR-1.0 = (x 1) \rightarrow (x \text{ "Hello"})$
- $TR-1.1 = (x 2) \rightarrow (x \text{ "World"})$
- $TR-1.2 = (x 3) \rightarrow (x 2)$
- $TR-1.3 = (x 4) \rightarrow (x 3)$
Example: Once VS. Fix

Rule-list

TR-1.0 = (x 1) → (x “Hello”)
TR-1.1 = (x 2) → (x “World”)
TR-1.2 = (x 3) → (x 2)
TR-1.3 = (x 4) → (x 3)

T = ((1 “Hello”) (2 “World”) (3 2) (4 3))

NEW-T = ((1 “Hello”) (2 “World”) (3 “World”) (4 2))

FINAL-T = ((1 “Hello”) (2 “World”) (3 “World”) (4 “World”))
Verification Challenge

How do we know transformations are correct?
High-Consequence Application: Sandia Secure Processor (SSP)

- A general-purpose computational infrastructure suitable for use in high-consequence embedded systems
- A simplified Java processor designed to be small and analyzable
- Closed system
SSP-classloader and HATS

- HATS is used to implement the SSP-classloader
- Functionality of the SSP-classloader is decomposed into five canonical forms
  - \( TLP_1 \): index resolution
  - \( TLP_2 \): static fields address calculation
  - \( TLP_3 \): instance field offset calculation
  - \( TLP_4 \): method table construction
  - \( TLP_5 \): inter-class absolute address and offset address distribution
Methodology

• Model the HATS TLP\textsubscript{1} in ACL2
  – Modeling the control strategies and the combiners, model_{TLP\textsubscript{1}}
  – Defining semantic function, S_0

• Prove that the application of the transformation rules preserves the semantics
## Methodology

- **Model the behavior of TLP₁**
  
  \(\text{fix-strategy (C}_{CF}\text{, rule-list)}\)
  
  - Applies the rule-list to \(C_{CF}\) exhaustively

- **Construct a semantic function \(S₀\) for TLP₁**
  
  \(\text{get-constant (n C}_{CF}\text{)}\)
  
  - Chases a pointer \(n\) down in a table \(C_{CF}\)

- **Main conjecture:**
  
  \(\forall (C_{CF}) \ S₀ (\text{model}_{TLP₁} (C_{CF})) = S₀ (C_{CF}), \ i.e.,\)

  \(\forall (C_{CF}), \ \text{get-constant (n, (fix-strategy (C}_{CF}\text{, rule-list)))} = \)

  \(\text{get-constant (n C}_{CF}\text{)}\)
Simplified ACL2 Model of TLP₁

- **Put-in-place** (new-node, classfile)
- **apply-rule-to-node** (rule, i, classfile)
- **apply-rule-list-to-node** (rule-list, i, classfile)
- **once-strategy** (rule-list, tail, classfile)
- **fix-strategy1** (rule-list, classfile)
- **generate-rules** (classfile)
- **fix-strategy** (classfile)
Verification

• Proof of termination of *fix-staregyl*

• Proof of the main conjecture
Proof of Termination

\[
\text{(deftm sum-addr-once-strategy-strictly-<}\n\text{(implies}\n\text{(and (well-formed-classfilep classfile)}\n\text{(some-matchp rule-list tail classfile))}\n\text{(< (sum-addr-to-resolve}\n\text{(once-strategy rule-list}\n\text{tail classfile))}\n\text{(sum-addr-to-resolve classfile))})\n\]
Proof of The Main Conjecture

\[ \forall (C_{CF}) \ (\text{get-constant } n \ (\text{fix-strategy } C_{CF})) = (\text{get-constant } n \ C_{CF})) \]

• Main conjecture in ACL2

(defun get-constant-n-fix-strategy1
  (implies (well-formed-classfilep classfile)
   (equal (get-constant n
      (equal (get-constant n
         (fix-strategy1 rule-list classfile))
         (get-constant n classfile))))

())