GL: Symbolic Simulation in the ACL2 Logic

Sol Swords

February 11, 2009
What is GL?

- GL is “G in the Logic” – a verified rewrite of Bob and Warren’s G system.
- GL is a utility for proving theorems by symbolic execution.
- Similar to proof by exhaustive testing, but one symbolic execution can replace $n$ concrete tests.
  - Often for very large $n$!
GL allows symbolic execution of ACL2 functions.

- Code transform produces a new function \( G-\text{FOO} \) from a given function \( \text{FOO} \)
- Run \( G-\text{FOO} \) on \textit{symbolic inputs} to produce \textit{symbolic outputs}
- The symbolic outputs produced by \( G-\text{FOO} \) represent the value of \( \text{FOO} \) on all concrete inputs represented by the supplied symbolic inputs.
Symbolic Values

- Symbolic values are objects that represent functions.
  - GL uses BDDs to represent Boolean-valued functions of Booleans, and wraps these in structured objects to produce arbitrary-valued functions of Booleans.
  - An evaluator gives the value of this function on a set of Boolean inputs.

- Example (with \(v_0, v_1\) BDD variables):
  
  \[
  (:\text{G-ITE} (:\text{G-BOOLEAN} . v_0) \text{FOO} . (:\text{G-BOOLEAN} . v_1))
  \]

  represents this function:

  \[
  f(v_0, v_1) = \begin{cases} 
  \text{FOO} & \text{if } v_0 = \text{T} \\
  \text{T} & \text{if } v_0 = \text{NIL} \text{ and } v_1 = \text{T}, \\
  \text{NIL} & \text{if } v_0 = v_1 = \text{NIL}.
  \end{cases}
  \]

- Exercise: What is \text{G-BOOLEANP} of the above object?
DEMO: Symbolic objects and evaluation
Symbolic Functions

- Symbolic versions of ACL2 primitives are defined and proven correct manually.
- Symbolic versions of user functions can be created with the `MAKE-G-WORLD` event.
  - Produces symbolic analogues and correctness lemmas for a set of functions.
- Each symbolic function takes arguments corresponding to the original function’s formals, plus two extra:
  - `hyp` - a BDD describing a working assumption for the simulation. Typically use T at the top level.
  - `clk` - a natural number which is decreased on recursive calls. When it reaches 0, symbolic functions will produce `G-APPLY` objects instead of simulating further. Use something sufficiently large.
DEMO: Symbolic functions
Proofs using GL

General strategy:

- Design symbolic objects that cover all inputs that satisfy the hyps
- Produce the symbolic analogue of the conclusion
- Show that running it on the symbolic inputs always yields T.
- Use the correctness lemma of the symbolic analogue to complete the proof.
- Automated in DEF-G-THM, DEF-G-PARAM-THM.
DEMO: Proofs
Implementation

- Hand-defined primitives
- **MAKE-G-WORLD:**
  - Define new evaluator
  - “Factor” functions
  - Generate symbolic analogues
  - Prove return types and guards
  - Prove correctness theorems
- Automation necessities
  - Restricted theories
  - Specialized clause processors orchestrated by computed hints