A Generalized Solution for the While Challenge

Sandip Ray

Department of Computer Sciences

University of Texas at Austin

sandip@cs.utexas.edu

http://www.cs.utexas.edu/users/sandip/

Rump Session Presentation

ACL2 Workshop 2007

Bill Young's "While Challenge"

Introduce the following equation in ACL2

```
(equal (run stmt st)
       (case (op stmt)
         (skip (run-skip stmt st))
         (assign (run-assignment stmt st))
         (if (if (zerop (evaluate (arg1 stmt) st)))
                 (run (arg3 stmt) st)
               (run (arg2 stmt) st)))
         (while (if (zerop (evaluate (arg1 stmt) st))
                    st
                  (run stmt (run (arg2 stmt) st)))
         (sequence (run (arg2 stmt) (run (arg1 stmt) st)))
         (otherwise st)))
```

Yesterday's talk: Challenge answered by John Cowles and Dave Greve

- Additionally requires that `run` is strict in `(btm)`

Kaufmann's "Generalized While" Challenge

- Implement a macro for defining operational semantics of languages with unbounded while loops

- Show that a more general reflexive equation can be introduced with ACL2

```
(equal (run x st)
       (cond ((equal st (btm)) (btm))
             ((test1 x st) (finish x st))
             ((test2 x st) (run (dst1 x st) (stp x st)))
             (t (let ((st2 (run (dst1 x st) (stp x st))))
                  (run (dst2 x st st2) st2)))))
```

where

```
(implies (not (equal st (btm)))
         (not (equal (finish x st) (btm))))
```

This talk reports progress in answering Kaufmann's challenges.

Our Results

- Implement a macro for defining operational semantics of languages with unbounded while loops
  - Developed a macro `definterpreter` to introduce such semantics

- Show that a more general reflexive equation can be introduced with ACL2
  - Introduced the suggested equation about `run` given encapsulated functions `test1`, `test2`, `finish`, `btm`, etc.

Basic Approach

First define a “clocked version” of `run`.

```
(defun run-clk (x st clk)
  (cond ((zp clk) (btm))
        ((equal st (btm)) (btm))
        ((test1 x st) (finish x st))
        ((test2 x st) (run-clk (dst1 x st) (stp x st) (1- clk)))
        (t (let ((st2 (run-clk (dst1 x st) (stp x st) (1- clk))))
             (if (equal st2 (btm)) (btm)
               (run-clk (dst2 x st st2) st2 (1- clk)))))))
```

Then eliminate `clk` using quantification.

```
(defun-sk exists-enough (x st)
  (exists clk (and (natp clk)
                   (not (equal (run-clk x st clk) (btm))))))

(defun run (x st)
   (if (exists-enough x st)
       (run-clk x st (exists-enough-witness x st))
     (btm)))
```

Essentially a formalization of Cowles’ proof in an abstract setting.

Macro for Language Interpreter

- Young's equation can be introduced by appropriate functional instantiation of `test1`, `test2`, `dst1`, `stp`, etc.
  - Cowles [private communication] showed the functional instance necessary.
- My macro `definterpreter` automates the functional instantiation and can introduce languages with unbounded while loops.
  - Provides some executability support via `mbe` construct.

A Sneak Peek at Macro

```
(definterpreter run stmt mem
    :op-field (op stmt)
    :bottom nil
    :executable t
    :verify-guards nil
    :vanilla-interpreter (((:name skip)
                            (:interpreter mem))
                           ((:name assign)
                            (:interpreter (run-assignment stmt mem))))
    :sequence ((:name sequence)
               (:arg1 (arg1 stmt))
               (:arg2 (arg2 stmt)))
    :conditional ((:name if)
                  (:test  (zerop (eval-expr (arg1 stmt) mem)))
                  (:tbr (arg3 stmt))
                  (:fbr (arg2 stmt)))
    :while ((:name while)
            (:test (zerop (eval-expr (arg1 stmt) mem)))
            (:body (arg2 stmt)))))
```

Coming Up

- Cowles showed that a number of reflexive equations can be introduced by functional instantiation of the generic theorem.
  - Stay tuned for the next talk

- We are developing a macro defreflexive to automate introduction of such equations.
  - A very preliminary implementation is available.

Details and Request

For details please see

books/workshops/2007/cowles-et-al/support/ray/

I will appreciate any comments, and in particular interface suggestions for the macros.