While-language Challenge: First Progress

2007 ACL2 Workshop

John Cowles, University of Wyoming
Dave Greve, Rockwell Collins
Bill Young, University of Texas

Last updated: November 13, 2007

The Issue

The `defpun` work of Manolios and Moore showed that it is possible to introduce "partial functions" soundly into ACL2.

However, `defpun` comes with rather severe syntactic restrictions on the class of functions admitted.

Can this be usefully extended to include a much larger class of functions?

The While Language

G. Smith uses the following language to explore information flow analysis:

$$
\begin{array}{lll}
\mathit{cmd} & ::= & \mathit{var} := \mathit{exp} \mid \\
 & & \textbf{skip} \mid \\
 & & \textbf{if } \mathit{exp} \textbf{ then } \mathit{cmd} \textbf{ else } \mathit{cmd} \mid \\
 & & \textbf{while } \mathit{exp} \textbf{ do } \mathit{cmd} \mid \\
 & & \mathit{cmd} \ ; \ \mathit{cmd}
\end{array}
$$

It's trivial to model this in ACL2, except for that pesky **while** clause.

Modeling in Lisp

The "obvious" interpreter function isn't admissible within ACL2 because there is no well-founded measure for the **while** clause.

```
(defun run (stmt st)
  (case (op stmt)
        (skip     (run-skip stmt st))
        (assign   (run-assignment stmt st))
        (if       (if (true-p (evaluate (arg1 stmt) st))
                      (run (arg2 stmt) st)
                    (run (arg3 stmt) st)))
        (while    (if (false-p (evaluate (arg1 stmt) st))
                      st
                    (run stmt (run (arg2 stmt) st))))
        (sequence (run (arg2 stmt)
                       (run (arg1 stmt) st)))
        (otherwise st)))
```

Dealing with the Issue

The obvious work-around is to add a *clock* argument that decreases in each recursive call, or at least in the while clause.

But this complicates the semantics and makes reasoning about the program behavior more difficult.

The while language isn't in the appropriate form to be handled by `defpun`. Is there a better way?

The Challenge

Young's challenge: Construct an ACL2 function (necessarily total) that satisfies the "obvious" defining equation of the while language.

Kaufmann's extension: extend `defpun` to allow ACL2 to admit a more general class of partial functions, including the while language.

The Responses

- Dave Greve: the "Rockwell Solution"
- John Cowles: the "Wyoming Solution"
- Sandip Ray: the "Sandip Subsumption"