Another Introduction to the ACL2 Waterfall

David L. Rager
ragerdl@cs.utexas.edu
July 30, 2010

It is useful to view the ACL2 waterfall from two angles. The first angle introduces the flow of the waterfall’s proof process (see Figure 1). This includes the parts that perform simplification, destructor elimination, fertilization, generalization, and the elimination of irrelevance. Induction is also considered part of the waterfall, but it occurs at a higher functional level.

The second view of the waterfall comes from examining the functions themselves. With the exception of induction, each of the proof steps used in the waterfall are implemented with functions called by waterfall-step1. The call graph for the waterfall can be found in Figure 2. The indentations represent calls to the indented function name from the function name relative to its indent. As an example, waterfall1-lst is called from waterfall1, and since waterfall1-lst has no other function names at the same indentation level, we know waterfall1 calls no other function that we deemed to be significant enough to warrant listing. Waterfall1-lst is shown as calling itself, because it recurs on all but the first element of the given clause list. The use of the infix XOR indicates that the function calls are part of a Lisp cond that results in calling exactly one of the functions. The optional call is on the same conceptual level as the function call following it, so waterfall0 is not indented. Since waterfall0 can directly call five distinct waterfall functions, it is the longest and most complex function listed.

Waterfall1-lst accepts a list of clauses to prove. It begins the proof of each clause by calling waterfall1 on the first element in that list of clauses. Waterfall1-lst’s recursive call is on the rest of that list of clauses. In ACL2 vernacular, each of these clauses is known as a subgoal.

References

Figure 1: The ACL2 Waterfall, Highlighting the Simplifier[1]

Figure 2: Function Call Graph for the ACL2 Waterfall