Using ACL2's Verified Clause Processor Mechanism to Sort Commutative and Associative Operations

Erik Reeber
CS Department
1 University Station, M/S C0500
The University of Texas
Austin, TX 78712-0233

E-mail: reeber@cs.utexas.edu

Verified Clause Processors

- New form of meta-reasoning available in ACL2 v3.2 and later.
- Clause processors reduce an ACL2 proof goal into a list of new goals that together imply the original.
- Matt Kaufmann, J Strother Moore, Sandip Ray, and Erik Reeber. Integrating External Deduction Tools with ACL2. To Appear in the *Journal of Applied Logic, Special Issue on Empirically Successful Computerized Reasoning (ESCoR)*.
- Similar to meta-rules, with the following advantages:
  - Can generalize
  - Can implement techniques that are not inside-out

Example problem: Sorting

- Proofs sometimes require sorting arguments to operations that are commutative and associative.
  - Some typical sorting theorems for bit-vector addition:

```
(DEFTHM BV+COMMUTE
  (IMPLIES
   (AND (SYNTAXP (NOT (BV+ORD X Y)))
        (BVP X) (BVP Y))
   (EQUAL (BV+BIN X Y) (BV+BIN Y X)))
  :RULE-CLASSES ((:REWRITE :LOOP-STOPPER NIL)))

(DEFTHM BV+REORDER
  (IMPLIES
   (AND (SYNTAXP (NOT (BV+ORD X Y)))
        (BVP X) (BVP Y) (BVP Z))
   (EQUAL (BV+BIN X (BV+BIN Y Z))
          (BV+BIN Y (BV+BIN X Z))))
  :RULE-CLASSES ((:REWRITE :LOOP-STOPPER NIL)))
```

  - may be too inefficient for large expressions.

Sorting Clause Processor

- BV+ may also be sorted using a verified clause processor.
- Clause processor implemented as BV+SORT function satisfying:

```
(DEFTHM CORRECTNESS-OF-BV+SORT
  (IMPLIES (AND (PSEUDO-TERM-LISTP CLAUSE)
                (ALISTP ENV)
                (EVL-BV+ (CONJOIN-CLAUSES (BV+SORT CLAUSE)) ENV))
           (EVL-BV+ (DISJOIN CLAUSE) ENV))
  :RULE-CLASSES :CLAUSE-PROCESSOR)
```

- where:
  - EVL-BV+ is an ACL2 evaluator for IF, BV+BIN, and BVP.
  - PSEUDO-TERM-LISTP recognizes well-formed ACL2 clauses.
  - DISJOIN creates the disjunction represented by an ACL2 clause.
  - CONJOIN-CLAUSES creates the conjunction of disjunctions represented by a list of ACL2 clauses.

Example Usage

- The sorting clause processor is accessed through the hint mechanism:

```
(DEFTHM FIRST-TEST
  (IMPLIES
   (AND (BVP X0) (BVP X1) (BVP X2) (BVP X3))
   (EQUAL (F (BV+BIN X3 (BV+BIN X2 (BV+BIN X1 X0))))
          (F (BV+BIN X0 (BV+BIN X1 (BV+BIN X2 X3))))))
  :HINTS (("GOAL"
           :CLAUSE-PROCESSOR (:FUNCTION BV+SORT))))
```

- In this case, 5 subgoals are produced:

```
Subgoal 5
(IMPLIES (AND (BVP X0) (BVP X1) (BVP X2) (BVP X3))
         (EQUAL (F (BV+BIN X0 (BV+BIN X1 (BV+BIN X2 X3))))
                (F (BV+BIN X0 (BV+BIN X1 (BV+BIN X2 X3)))))).

Subgoal 4
(IMPLIES (NOT (BVP X0))
         (IMPLIES (AND (BVP X0) (BVP X1) (BVP X2) (BVP X3))
                  (EQUAL (F (BV+BIN X3 (BV+BIN X2 (BV+BIN X1 X0))))
                         (F (BV+BIN X0 (BV+BIN X1 (BV+BIN X2 X3))))))).

...
```

Performance Comparison

- Extended example problem to different numbers of bit-vector arguments.
- Without the sorting clause processor:
  - 7.12, 127.60, and 1862.55 seconds to verify property with 100, 200, and 400 arguments respectively.
- With sorting clause processor:
  - 0.05, 0.30, and 1.51 seconds to verify property with 100, 200, and 400 arguments respectively.
  - 19.08 and 139.47 seconds to verify property with 1000 and 2000 arguments (only 0.04 and 0.08 seconds during SCP).
- Journal paper has results without `BVP` hypotheses:
  - 10.55 and 58.15 seconds to prove the 500 and 1000 argument theorem without the clause processor.
  - 0.02 and 0.05 seconds to prove the 500 and 1000 argument theorem with the clause processor.

Conclusion

- Verified clause processors present a new method of extending the theorem prover.
  - Useful for creating efficient domain-specific proof techniques.
- `BV+` sorting clause processor.
  - Much more efficient than traditional rewrite-based approach.
  - Must be accessed through a clause hint.
- Generalized into a macro for any commutative and associative operation.
  - Will make publicly available, "`books/clause-processors/sorting-cp/`".