Useful New Books for General-purpose Theorem Proving

Sol Swords

January 20, 2010

Outline

We'll introduce a few new(ish) books that may be useful in a broad range of theorem proving applications.

tools/bstar:	Hackable let*.
tools/rulesets:	Flexible theory management.
tools/flag:	Induction schemes for mutual recursions.
tools/mv-nth:	Stop rewriting (mv-nth 0 x) to (car x).
defsort/defsort:	Macro to define fast sorting algorithms.
cert.pl:	Parallel build system with automatic dependency scanning.

All of these are available in the ACL2 books repository.
Some talk, mostly demo.

tools/bstar.lisp

B*: Flexible binder macro.

- Use like let*
- Inline MV-LETs, conditionals – no extra indentation
- User-defined binder constructs – very flexible
- In-place ignore/ignorable, type declarations

Demo follows...

tools/rulesets.lisp

Rulesets: Theory management framework

- Like deftheory, but...
- Rulesets are mutable
- They use make-event so that the results don't depend on extraneous events
- They are just table events under the hood, so they may be local, redundant, ...
- Comes with useful macros for use in `IN-THEORY`

Demo follows...

tools/flag.lisp

Flag: Induction schemes for mutual recursions

- Defines a "flag function" for a previously defined mutual recursion
- Also defines a macro useful for proving theorems about that mutual recursion

Demo follows...

tools/mv-nth.lisp

MV-NTH: Simple rewriter for MV-NTH.

- Pet peeve: With MV-NTH enabled,

```
(MV-NTH 0 (function call)) → (CAR (function call))
(MV-NTH 1 (function call)) → (MV-NTH 1 (function call))
```

  - Theorems with MV-LET (or B*) in the conclusion make terms with (MV-NTH 0 ...)
- Disabling MV-NTH leaves you with terms like

```
(MV-NTH 2 (LIST val0 val1 val2))
```

- mv-nth.lisp adds a meta rule that solves the above problem when MV-NTH is disabled.
- (set-inhibit-warnings "theory")

defsort/defsort.lisp

`DEFSORT`: Define a sorting function...

- Automates introduction of sorting functions for arbitrary comparators
- Highly optimized
- Guards proven automatically
- Correctness theorem proven automatically

Demo follows...

cert.pl

cert.pl: Automated book build system

- Directory-oblivious parallelism
- Automatic dependency scanning
- Supports .acl2 file strangeness, `add-include-book-dir`, ...
- Can create a static makefile for users without Perl

Demo follows...

Miscellany

str/top:	String library with optimized functions and nice logical definitions
tools/defevaluator-fast:	Exactly the same as defevaluator, but much faster for large numbers of functions
tools/defined-const:	Defconst, and additionally proves a theorem saying that the constant equals its definition, which in ACL2H is only executed once
clause-processors/join-thms:	Macro for defining required lemmas about disjoin, conjoin, conjoin-clauses for clause processor rules
clause-processors/generalize:	Generalize away specified subterms into new variables
clause-processors/use-by-hint:	Use already proven theorems in clause processors and discharge the resulting side conditions quickly