What's New in the Community Books

Since the ACL2-2017 Workshop

Cuong Chau[9], Alessandro Coglio[5], John Cowles[10], Jared Davis[1], Andrew Gacek[7], Ruben Gamboa[10], Shilpi Goel[3], Mark Greenstreet[8], David Greve[7], Matt Kaufmann[9], Keshav Kini[6], Carl Kwan[8], Mihir Mehta[9], J Moore[9], David Russinoff[2], Julien Schmaltz[4], Rob Sumners[3], Sol Swords[3], Yan Peng[8]

[1]Apple, [2]ARM, [3]Centaur, [4]Eindhoven Institute of Technology, [5]Kestrel Institute, [6]Oracle, [7]Rockwell Collins, [8]University of British Columbia, [9]University of Texas at Austin, [10]University of Wyoming

ACL2-2018 Workshop

New Libraries

`build/ifdef.lisp`: Defines `ifdef` and `ifndef` forms which test environment variables; supported by the build system.

`centaur/acre`: New regular expression implementation supporting features somewhat similar to Perl regexes.

`centaur/bitops/sparseint.lisp`: Library representing bignums as balanced trees to efficiently support operations that preserve large ranges of bits.

`centaur/glmc`: Interface to hardware model checkers.

`centaur/truth`: Integer-encoded truth table library.

`coi/quantification/quantified-congruence.lisp`: A library for proving congruences about quantified formulae.

New Libraries

`kestrel/apt`: APT (Automated Program Transformations), a toolkit to transform programs and program specifications with automated support.

- Includes two of Kestrel's ∼40 transformations.
- Also includes some utilities used across transformations.
- More forthcoming.

`kestrel/auto-termination`: `defunt` is a variant of `defun` that can prove termination using previously-proved termination theorems from a large set of community books, as described in the paper *DefunT: A Tool for Automating Termination Proofs by Using the Community Books* at this workshop.

New Libraries

`kestrel/bitcoin`: A (small start towards a) library for the Bitcoin cryptocurrency and ecosystem.

- Executable specification of Base58 encoding and decoding.
- Executable specification of Base58Check encoding.

`kestrel/ethereum`: A library for the Ethereum cryptocurrency and ecosystem.

- Executable specification of RLP (Recursive Length Prefix) encoding; declarative specification of RLP decoding.
- Executable specification of hex-prefix encoding.
- Kestrel is actively working on this.

New Libraries

`kestrel/java`: A library for Java.

- AIJ (ACL2 In Java), a deep embedding of ACL2 in Java.
- ATJ (ACL2 To Java), a Java code generator for ACL2.
- These are described in the paper *A Simple Java Code Generator for ACL2 Based on a Deep Embedding of ACL2 in Java* at this Workshop.

New Libraries

`kestrel/utilities/apply-fn-if-known.lisp`: Apply a function, expressed as a package and a name, if it exists.

`kestrel/utilities/auto-instance.lisp`: `defthm<w` will attempt to prove a theorem directly from previously-proved theorems by generating suitable hints, using `previous-subsumer-hints`.

`kestrel/utilities/digits-any-base`: Conversions between natural numbers and their representations in arbitrary bases.

- Big and little endian.
- Minimal, minimal non-zero, or specified length.
- Several theorems, e.g. about inversions.

New Libraries

`kestrel/utilities/er-soft-plus.lisp`: The logic-mode utilities `er-soft+` and `er-soft-logic` produce soft errors with specified error triples.

`kestrel/utilities/fixbytes`: Fixtypes for unsigned and signed bytes, and true lists thereof.

- Macros to create fixtypes and theorems for a specified size. The size may be a constrained nullary function, e.g. useful to formalize C bytes.
- Several instances available; just include the respective file(s).
- These are candidate extensions of the `fty` library.

`kestrel/utilities/include-book-paths.lisp`: List paths via `include-book` down to a given book; may be useful for reducing book dependencies.

New Libraries

kestrel/utilities/integer-range-*.lisp: Utilities related to `integer-range-p`.

- Parameterized recognizer `integer-range-listp`.
- Parameterized fixers `integer-range-fix` and `integer-range-list-fix`.
- Several theorems.

kestrel/utilities/magic-macroexpand.lisp: Logic-mode macroexpansion.

kestrel/utilities/messages.lisp: A few utilities for `msgp` values, e.g. to convert the first character to upper/lower case.

kestrel/utilities/orelse.lisp: Try one event, then a second one if the first fails.

New Libraries

`kestrel/utilities/proof-builder-macros.lisp`: A book that defines some proof-builder macros. Current contents include definitions of:

- `when-not-proved` to skip instructions when all goals have been proved;
- `prove-guard` and `prove-termination`, for using previously-proved guard or termination theorems efficiently; and
- a more general macro, `fancy-use`, for using lemma instances efficiently.

New Libraries

`kestrel/utilities/skip-in-book.lisp`: The utility, `skip-in-book`, wraps around a form to prevent its evaluation during book certification or inclusion.

`kestrel/utilities/symbols.lisp`: Some utilities for symbols.

- These could become a new `std/symbols` library.

`kestrel/utilities/system/paired-names.lisp`: Utilities for names consisting of two parts with a customizable separator in between. (Used by APT, but more general.)

`kestrel/utilities/untranslate-preprocessing.lisp`: A macro `add-const-to-untranslate-preprocess` to keep a named constant unexpanded in the screen output.

New Libraries

kestrel/utilities/xdoc: XDOCumentation utilities.

- Constructors of well-tagged XDOC strings, e.g.

  ```
  (xdoc::p "This is a paragraph.")
  (xdoc::ul
    (xdoc::li "First unordered item.")
    (xdoc::li "Second unordered item."))
  ```
- `defxdoc+` extends `defxdoc` with `:order-subtopics t/nil` and `:default-parent t/nil`.
- These are candidate extensions of the xdoc library.

New Libraries

`projects/arm`: Proofs of correctness of some floating-point operations, as implemented in the FPU of an Arm Cortex-A class high-end processor.

`projects/async/tools/convert-edif.lisp`: Convert between EDIF format and a convenient s-expression format.

`projects/avr-isa`: Formal model of the ISA of the AVR 8-bit controller.

- Supports a paper at the ACL2-2013 Workshop; see comments in the file `avr8_isa.lisp`.

`projects/irv`: Formalization of an instant-runoff voting scheme, described in a rump talk at this Workshop.

New Libraries

`projects/pltpa`: An ACL2 Implementation of the Edinburgh Pure Lisp Theorem Prover of 1973.

`projects/rac`: A translator from RAC (Restricted Algorithmic C) to ACL2.

- Replaces `projects/masc`.

`projects/sat/zz-resolution-checker`: An early SAT proof-checker from 2011 based on resolution (see README).

New Libraries

`std/io/open-channels.lisp`: Lemmas about how open channels are affected or unaffected by various state-modifying functions.

`std/stobjs/updater-independence.lisp`: Utility for defining stobj and stobj-like accessor/updater independence theorems.

`std/util/termhints.lisp`: Hint utility described in the paper *Hint Orchestration Using ACL2's Simplifier* at this Workshop.

`tools/run-script.lisp`: This utility supports testing of evaluation of the forms in a given file, to check that the output is as expected. Several community books utilize it.

`workshops/2018`: Supporting materials for some of the papers at this Workshop. The supporting materials for other papers at this Workshop are elsewhere, not under this directory.

Improved Libraries

`centaur/aignet`: And-Inverter Graph (AIG) representation for Boolean functions and finite-state machines.

- New verified AIGNET transforms including FRAIGing, DAG-aware balancing and rewriting.
- AIGNET natively supports XORs, i.e. represents them using one node instead of three.

`centaur/bitops/rotate.lisp`: Bit-vector rotation libraries.

- Generalized existing theorems and added a new theorem for compositions of `rotate-left` operations, as well as a theorem for compositions of `rotate-right` operations.
- To do: Add theorems for compositions of `rotate-left` and `rotate-right` with each other.

Improved Libraries

centaur/fty/bitstruct: Define a bit vector type with accessor/updater functions for its fields.

- The :exec part of the mbe in accessor and updater functions now has efficient, heavily type-declared code that avoids bignum operations whenever possible.
- Accessor and updater functions can now be inlined.

centaur/gl: Symbolic simulation framework for solving finite theorems.

- Add hooks in GL to allow calling AIGNET transforms before SAT.
- Improve GL counterexample generation for term-level reasoning.
- Added accumulated-persistence-like rule profiling.

Improved Libraries

`centaur/sv`: Hardware verification library with vector-based expression representation.

- Many SV/SVEX algorithms are now based on `sparseints` so that they scale when dealing with variables thousands/millions of bits in size.

`centaur/vl`: Library for SystemVerilog and regular Verilog.

- Add new SystemVerilog lint check based on accurately determining used/set ranges of vectors.

Improved Libraries

`coi/generalize/generalize.lisp`: A library that generalizes terms that appear as arguments to the function `(gensym::generalize term)`.

- Now supports one-step generalization of multiple terms.

`coi/nary/nary.lisp`: A library supporting parametrized equivalence relations and related congruences.

- Improved support for non-traditional congruences involving implications rather than equalities.

`coi/util/deffix.lisp`: Given an equivalence relation, the macro `def::fix` witnesses an appropriate fixing function.

- Added support for witnessing fixing functions that preserve (fix) a type.

Improved Libraries

`kestrel/soft`: SOFT (Second-Order Functions and Theorems), macros to mimic second-order functions and theorems.

- Added full support for `defun-sk2`.
- Improved user interface.

`kestrel/utilities/...`: Started refactoring some of these utilities to reduce book dependencies.

`kestrel/utilities/copy-def.lisp`: Made improvements: better handling of `mutual-recursion` and of the `:equiv` argument, and generated `:expand` hint for better handling of recursion.

`kestrel/utilities/directed-untranslate.lisp`: Made several improvements to `directed-untranslate`, in particular for `let`, `let*`, `mv`, `mv-let`, and `b*`, including enhanced executability of the result.

Improved Libraries

`kestrel/utilities/error-checking.lisp`: Utilities to check error conditions and return customizable error messages.

- Improved the `def-error-checker` macro, e.g. to support logic-mode error-checking functions.
- Added several error-checking functions.

`kestrel/utilities/osets.lisp`: Utilities for osets.

- Added a fixtype for osets.
- These are candidate extensions of the `std/osets` library.

`kestrel/utilities/strings`: String manipulation libraries.

- Added several new rewrite rules.

Improved Libraries

`kestrel/utilities/system/defun-sk-queries.lisp`: Utilities to query `defun-sk` functions.

- Added support for the recently added `:constrain` option.
- These could become part of a new `std/system` library.

`kestrel/utilities/system/terms.lisp`: Utilities to manipulate terms.

- Added and improved several utilities.
- Moved some utilities to a separate file `term-function-recognizers.lisp`.
- These could become part of a new `std/system` library.

Improved Libraries

`kestrel/utilities/system/world-queries.lisp`: Utilities to query worlds.

- Added and improved several utilities.
- There are two variants for most of these utilities: a "fast" one and a "logic-friendly" one (see documentation for details).
- These could become part of a new `std/system` library.

`kestrel/utilities/user-interface.lisp`: Utilities for customizing screen output of user-defined events.

- Added several utilities.

Improved Libraries

`misc/assert.lisp` & **`misc/eval.lisp`**: Testing utilities.

- Added some utilities moved from `kestrel/utilities/testing.lisp`.
- Added some XDOCumentation.
- Renamed some utilities for greater uniformity (deprecated the old names).
- Reduced book dependencies.

`misc/expander.lisp`: The expander has been improved in several ways.

Improved Libraries

`misc/install-not-normalized.lisp`: Improved `install-not-normalized` to handle cases in which recursively-defined functions have non-recursive normalized definitions.

`misc/profiling.lisp`: Profiling fixes for recent distributions of CCL.

`projects/apply` & **`projects/apply-model`**: Updated books pertaining to `apply$`.

Improved Libraries

`projects/async`: ASYNC, the framework for modeling and verifying the functional correctness of asynchronous (self-timed) circuit models.

- Developed a new compositional methodology for scalable formal verification of functional properties of self-timed circuit designs.
- Verified the functional correctness of data-loop-free self-timed circuits (see `fifo/`).
- Verified the functional correctness of a self-timed serial adder/subtractor model (see `serial-adder/`).
- Verified the functional correctness of iterative self-timed circuit models that compute the greatest-common-divisor (GCD) (see `gcd/`).
- Verified the functional correctness of self-timed circuits performing arbitrated merge operations (see `arbitration/`).

Improved Libraries

projects/filesystems: Formal models of filesystems.

- M1 and M2, new filesystem models for FAT32, described in the paper *Formalising Filesystems in the ACL2 Theorem Prover: an Application to FAT32* at this Workshop.

projects/sat/lrat: SAT proof-checker extensions (improved theorem, extension to cube-and-conquer; see README).

projects/smtlink: Smtlink, a framework for integrating external SMT solvers into ACL2.

- Smtlink has experienced great architecture refactoring and was moved from `workshop/2015/peng-greenstreet` to `projects/smtlink`, as described in the paper *Smtlink 2.0* at this Workshop.
- Developed new XDOC documentation.
- Added more toy examples and a ring oscillator proof example.

Improved Libraries

`projects/x86isa`: X86ISA, the formal model of the x86 ISA.

- Added support for 32-bit mode; see the paper *Adding 32-bit Mode to the ACL2 Model of the x86 ISA* at this Workshop.
- Improved and extended some documentation.
- The model's modes are now called "views" to avoid overloading the word "mode", which refers to an x86 processor's own modes of operation.
- Opcode dispatch functions and coverage data are generated from annotated opcode maps, which are taken from the Intel manuals.
- Added support for decoding VEX- and EVEX-encoded instructions (AVX/AVX2/AVX512).
- Decode-time exceptions are detected during opcode dispatch now, as opposed to inside individual instruction semantic functions.
- Added support for enabling/disabling machine features that depend on CPUID feature flags.
- Codewalker can now be used to reason about x86 programs.

Improved Libraries

rtl: The register-transfer logic library.

- Added an improved version of SRT division and square root.
- The old version was moved to projects/srt.

std/io/combine.lisp: Byte-combining libraries.

- Added invertibility theorems for combine16u and combine32u.
- To do: make these invertibility theorems compatible with part-select.
- To do: prove similar theorems for combine64u as well as for the signed-integer functions, combine16s et al.

Improved Libraries

`tools/flag.lisp`: The new keyword argument `:last-body` of `make-flag` specifies use of the most recent definition rule.

`tools/include-raw.lisp`: Fixed an issue with option `:do-not-compile t` by extending "fns-with-raw-code" state globals.

`tools/removable-runes.lisp`: Improved `removable-runes` and added related utility, `minimal-runes`, which returns a list of runes to enable that is sufficient for admitting a given event.

Additional Contributions

`workshops/references`: BibTeX references for all the ACL2 Workshop papers, and a LaTeX document that shows them.

`xdoc/fancy/lib/katex`: KaTeX, a JavaScript library for TeX math rendering on the web, has been updated to version 0.8.3.

Developers Guide: The topic `developers-guide` is, together with its subtopics, actually a manual for ACL2 development. It is intended for experienced ACL2 users who may wish to become ACL2 developers.