What's New in the Community Books

Since the ACL2-2018 Workshop

Cuong Chau[1], Alessandro Coglio[3] (presenter), Shilpi Goel[2], Eric McCarthy[3], Mihir Mehta[5], Yan Peng[4], David Russinoff[1], Eric Smith[3], Sol Swords[2], Mertcan Temel[5], Stephen Westfold[3]

[1]ARM, [2]Centaur, [3]Kestrel Institute,
[4]University of British Columbia,
[5]University of Texas at Austin

ACL2-2020 Workshop

New Libraries

`centaur/fgl`: Bitblasting rewriter, successor to GL.

- A rewriter that does bitblasting, rather than a bitblaster that has a rewriter.
- More extensible and customizable than GL.
- Adds incremental SAT support.
- Some fancy rewriter features discussed in the paper *New Rewriter Features in FGL* at this workshop.

New Libraries

`centaur/meta`: New library of metareasoning utilities.

- Based on **`clause-processors/pseudo-term-fty.lisp`**, treats pseudo-term as an FTY sum type.
- Unification, substitution, variable collection.
- Unconditional rewriter.
- Let-abstraction algorithm.
- Utilities for understanding rewrite, equivalence, and congruence rules.
- Term measure that decreases under beta-reduction.
- Utility that uses meta-extract plus a runtime check to effectively extend the functions understood by an evaluator.

New Libraries

`centaur/svl`: A framework to simulate and reason about Verilog modules with design hierarchy.

- Uses `centaur/sv` and `centaur/vl` books to generate simulation-ready SVL designs. May sometimes be used in lieu of SVTV.
- Selected submodules may not be flattened to retain design hierarchy.
- Does not support combinational loops (e.g., latches).

New Libraries

`kestrel/alists-light`: A "minimalist" library about alists.

- Covers `acons`, `assoc-equal`, `strip-cars`, `strip-cdrs`, `pairlis$`, etc.
- Introduces `lookup-equal`, etc.
- "Minimalist" approach seeks to minimize include-books, especially non-local ones. Example: "Just give me some theorems about function X. Don't make me include books about Y or Z. Don't make me include the books you used to prove the theorems about X."

`kestrel/arithmetic-light`: A "minimalist" library about arithmetic.

- Covers built-in arithmetic operations (`+`, `*`, `floor`, `mod`, `numerator`, etc.).
- Minimalist approach: One book per function. Almost zero non-local `include-books`.

New Libraries

`kestrel/axe`: Kestrel's Axe toolkit.

- Holding area: Almost nothing there yet!
- Will include Axe Rewriter, Axe Prover, Axe Equivalence Checker.
- Will include Axe Lifters (JVM, x86).
- Will include tools for input finding, answering questions about programs.
- Claim to fame: Structure-shared representation of terms as DAGs allows rewriting over very large terms (e.g., verifying crypto algorithms by unrolling loops).
- See `www.kestrel.edu/axe`.

`kestrel/booleans`: A lightweight library about operations on booleans.

- Covers `bool-fix`, `boolif`, `boolor`, `booland`, `boolxor`.
- Functions are guaranteed to return booleans, so you can easily tell the type. No need to examine every leaf of large nests.

New Libraries

`kestrel/bv`: Kestrel's library about bit-vectors (BVs).

- Represents bit-vectors as natural numbers.
- Supports the Kestrel JVM model and Axe.
- Functions are guaranteed to return bit vectors, so you can easily tell the type. No need to examine every leaf of large nests.
- Covers `bvchop`, `slice`, `bvcat`, `bvplus`, `bvxor`, BV rotation, etc.
- Much more material to add (in development over 10 years).

`kestrel/bv-lists`: A library about lists of BVs (see also `kestrel/bv`).

- Unpacking: Splitting a BV into a list of smaller BVs.
- Packing: Combining a list of smaller BVs into a single BV.
- Conversions between lists of bits and lists of bytes (big-endian and little-endian).
- Operations that map BV operations over lists: `bvchop-list`, `bvxor-list`.

New Libraries

kestrel/crypto: Executable specifications and abstract interfaces of cryptographic functions.

- Specification of short Weierstrass elliptic curves in a prime field. Proof of closure under the group operation. Instantiation to the secp256k1 curve used by Bitcoin and Ethereum.
- Specification of Deterministic ECDSA (Elliptic Curve Digital Signature Algorithm) following IETF RFC 6979.
- Specification of HMAC (Hashed Key Message Authentication Code) following IETF RFC 2104. Instantiated with SHA-256 and SHA-512.
- Specification of PBKDF2 (Password-Based Key Derivation Function 2) following IETF RFC 8018. Instantiated with HMAC-SHA-512.
- Specification of the Keccak family of permutations, sponge functions, and hash functions, and the FIPS 202 SHA-3 hash functions based on Keccak.
- Specifications of the SHA-2 family of hash functions (SHA-224, SHA-256, SHA-384, SHA-512) defined in FIPS 180-4.
- Specifications of some common message padding operations.

New Libraries

`kestrel/event-macros`: Utilities to develop event macros (i.e. macros at the event level) more quickly and consistently.

- Validation and elaboration of inputs of event macros.
- Handling of applicability conditions (i.e. theorems that must be proved for the event macro to apply).
- Controlling of screen output of event macros.
- XDOC constructors for both user and developer documentation of event macros.

New Libraries

`kestrel/file-io-light`: A lightweight library about file I/O.

- Deals with opening I/O channels, writing to files, etc.
- Provides lightweight books with rules about built-in functions.
- Defines some new functions (e.g., `write-bytes-to-channel`).

`kestrel/library-wrappers`: A directory of books that “wrap” other libraries.

- Each book includes some other library or book and then disables rules that may cause problems and/or introduces improved versions.
- Eventually, the wrapped libraries could be improved (and the wrapper books eliminated).

New Libraries

`kestrel/lists-light`: A “minimalist” library about lists.

- Small books about many built-in functions (`cons`, `append`, `take`, `union-equal`, etc.).
- Also defines some new functions (`perm`, `memberp`, `subrange`, etc.).
- Uses a minimalist style, as with other “light” libraries. Also helps with “auditing” a development (e.g., when reading all defintions in all books included by a spec).
- Perhaps list libraries are best included only locally in other developments (like arithmetic libraries).

New Libraries

`kestrel/prime-fields`: A formalization of operations over prime fields.

- Includes `add`, `sub`, `neg`, `mul`, `pow`, `inv`, and `div`, all modulo a supplied prime.
- `pow` calls existing `mod-expt-fast` for speed.
- Provides many simplification rules (in-progress; normal forms may change).
- Includes `bind-free` rules for equalities: canceling common addends and moving negated addends.

New Libraries

`kestrel/std/basic`: An extension of the Std/basic library.

- Added several functions to manipulate symbols.
- Added `mbt$`, a variant of `mbt` that requires `non-nil` instead of `t`.
- These will be gradually moved to the `std/basic` directory.

`kestrel/std/system`: A new Std/system library with standard system utilities that complement the built-in ones.

- Several utilities have been moved here from `kestrel/utilities/system`, and improved in the process.
- New utilities have been added.
- This is being gradually moved to the `std/system` directory.

New Libraries

`kestrel/std/util`: An extension of the Std/util library.

- Added `defarbrec`, to introduce recursive functions without proving termination right away. Compared to similar existing tools, it is mainly aimed at use with APT.
- Added `deffixer`, to introduce fixers and associated theorems.
- Added `defiso`, to verify and record isomorphic mappings; described in the paper *Isomorphic Data Type Transformations* at this workshop.
- Added `defsurj`, to verify and record surjective mappings.
- Added `defmax-nat`, to declaratively define the maximum of a (possibly infinite) set of natural numbers.
- Added `defmacro+`, which enhances `defmacro` with XDOC integration.
- These will be gradually moved to the `std/util` directory.

New Libraries

kestrel/utilities/conjunctions.lisp: Utilities for manipulating conjunctions.

kestrel/utilities/declares0.lisp: Basic utilities for manipulating declares (more will be added).

kestrel/utilities/def-constant-opener.lisp: A utility that generates an opener theorem for a function when all arguments are constant (used by Axe).

kestrel/utilities/defopeners.lisp: A utility for making opener rules for recursive functions.

kestrel/utilities/deftest.lisp: A utility for isolating tests and running them with extensive guard checking.

kestrel/utilities/defthm-events.lisp: Utilities for processing defthm forms.

New Libraries

`kestrel/utilities/defun-events.lisp`: Utilities for processing defun forms.

`kestrel/utilities/disables.lisp`: A book that disables some built-in functions that may be convenient to have disabled from the start.

`kestrel/utilities/doublets2.lisp`: Utilities that deal with doublets (true lists of length 2).

`kestrel/utilities/equal-of-booleans.lisp`: Rules to break an equality of two booleans into the equivalent conjunction of two implications.

`kestrel/utilities/erp.lisp`: Utilities for returning errors (which are often assigned to a variable called `erp`).

`kestrel/utilities/forms.lisp`: Basic utilities about forms that look like function calls.

New Libraries

`kestrel/utilities/keyword-value-lists2.lisp`: Utilities about keyword-value-lists.

`kestrel/utilities/make-and.lisp`: A utility to make an untranslated conjunction.

`kestrel/utilities/make-and-nice.lisp`: A utility to make a, possibly simplified, untranslated conjunction.

`kestrel/utilities/make-or.lisp`: A utility to make an untranslated disjunction.

`kestrel/utilities/make-or-nice.lisp`: A utility to make a, possibly simplified, untranslated disjunction.

`kestrel/utilities/my-get-event.lisp`: A utility to get the (untranslated) event that introduced a function.

New Libraries

kestrel/utilities/obags: Ordered bags, i.e. finite bags represented as non-strictly ordered lists. Cf. omaps (below) and osets (in Std/osets).

kestrel/utilities/omaps: Ordered maps, i.e. finite maps represented as strictly ordered alists. Cf. obags (above) and osets (in Std/osets).

kestrel/utilities/pack.lisp: Utilities for making symbols from strings, natural numbers, characters, and other symbols.

kestrel/utilities/remove-duplicates-equal-dollar.lisp: A utility to remove duplicates, keeping the first of each set.

New Libraries

`kestrel/utilities/smaller-termp.lisp`: A utility to compare the sizes of terms.

`kestrel/utilities/sublis-expr-plus.lisp`: Added `sublis-expr+`, which replaces terms by variables also inside lambda expressions.

`kestrel/utilities/substitution.lisp`: Utilities that perform substitution.

`kestrel/utilities/terms.lisp`: Various utilities for manipulating terms.

`kestrel/utilities/world.lisp`: Utilities for querying the ACL2 logical world.

New Libraries

projects/rp-rewriter: A customized rewriter and a verified clause processor. Implemented for multiplier verification but it is a generic rewriter. Details are discussed in the paper *RP-Rewriter: An Optimized Rewriter for Large Terms in ACL2* at this workshop.

projects/rp-rewriter/lib/{mult,mult2}: An automated and efficient tool to verify integer multipliers.

- Supports Booth Encoding, simple partial products, and Wallace-tree like multipliers.
- Uses SVL and RP-Rewriter.
- Two libraries, same functionality.
  - `mult`: more user-friendly debugging.
  - `mult2`: 30% faster.
- Very efficient: 64x64 multipliers in less than 2 seconds, some 1024x1024 multipliers in less than 10 minutes.

New Libraries

`std/testing`: A new Std/testing library, with standard utilities for building tests.

- `misc/assert.lisp` and `misc/eval.lisp` have been moved here and refactored into smaller files.
- This library contains utilities to build tests, not actual tests (aside from perhaps some to test the testing utilities).

`std/typed-alists`: A new Std/typed-alists library, with alists of various types.

- Analogous to Std/typed-lists.
- Add more typed alists, as needed.

Improved Libraries

`build`: Books build system.

- **`build/*.certdep`**: Track dependencies of books on certain ACL2 system features—see XDOC topic `acl2-system-feature-dependencies`.
- Add `ifdef-(un)define(!)` for setting/unsetting environment variables (non-)locally, with effects on `if(n)def` forms understood by the build system.
- **`build/cert.pl`** supporting libraries refactored into proper (?) Perl modules.

Improved Libraries

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

- Added `isodata` and `propagate-iso`, the isomorphic data type transformations described in the paper *Isomorphic Data Type Transformations* at this workshop.
- Added `parteval`, a partial evaluation transformation.
- Added `casesplit`, a case splitting transformation.
- Extended `tailrec`, the tail recursion transformation.
- Extended `restrict`, the domain restriction transformation.
- Added an APT defaults table to customize certain behaviors of the APT transformations.
- Added some APT-specific XDOC constructors for both user and developer documentation.

Improved Libraries

kestrel/bitcoin: A library for the Bitcoin cryptocurrency and platform.

- Formalized BIP (Bitcoin Improvement Proposal) 32, 39, 43, and 44. (These are for cryptocurrency wallets.)
- Added verified executable attachments for some declaratively specified functions.

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

- Completed the RLP development, which started before ACL2-2018. This development is described in the paper *Ethereum's Recursive Length Prefix in ACL2* at this workshop.
- Formalized Modified Merkle Patricia tree and the Ethereum database.
- Formalized the construction of signed transactions.
- Formalized the calculation of an account address from a public key.

Improved Libraries

`kestrel/fty`: Extensions of the FTY library.

- Added `deflist-of-len`, for lists of specified size.
- Added `defset`, for osets of specified types.
- Added `defomap`, for omaps of specified types.
- Improved `defbyte` and `defbytelist` to generate more theorems.
- Added `defbyte-ihs-theorems`, to generate theorems about `defbyte` fixtypes and functions in the IHS library.
- Added `deffixequiv-sk`, to automate the proofs of `deffixequiv` for `defun-sk` functions.
- Added `defflatsum`, for flat (i.e. not tagged) sums of disjoint types.

Improved Libraries

`kestrel/hdwallet`: A proof-of-concept hierarchical deterministic wallet for cryptocurrencies.

- Uses components from the cryptographic, Bitcoin, and Ethereum libraries.
- Holds ether, but can be extended to other currencies.
- Packaged into a Docker image available on the Docker Hub. Docker build information is included.
- See the file `kestrel/hdwallet/README.md` for build and usage instructions.

Improved Libraries

kestrel/java: A library for Java.

- Significantly extended ATJ, the Java code generator for ACL2. In particular, a shallow embedding approach is now supported. Described in a rump talk at this workshop.
- Extended the Java language formalization with models of various aspects of the language's syntax and semantics.
- Added a grammar of Java written in ABNF (Augmented Backus-Naur Form), and processed it with the verified ABNF grammar parser in `kestrel/abnf`. (This parses the ABNF grammar of Java, not Java.)

kestrel/soft: Macros to mimic second-order functions and theorems.

- Simplified macros to no longer require explicit function parameters.

Improved Libraries

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

- Added functions to group and ungroup digits between larger and smaller bases.
- Added macros to generate specialized, more concise functions for specified bases.

`kestrel/utilities/strings`: String utilities (to be eventually integrated with Std/strings).

- Added functions to convert between strings or character lists and even-length lists of hexadecimal digits.

`kestrel/utilities/typed-lists`: Typed list utilities (to be eventually integrated with Std/typed-lists).

- Added `bit-listp`, with associated theorems.

Improved Libraries

`projects/filesystems`: The filesystem books now cover many more system calls, include many more cosimulation tests, and have verification of the transformations between HiFAT, an abstract model of FAT32 with directory trees, and LoFAT, the concrete model of FAT32 which replicates the on-disk FAT32 format. These verified transformations support refinement proofs between LoFAT syscalls and their HiFAT equivalents, and these in turn support proofs (of which examples can be seen in the directory) to be carried out for real executable programs which use file operations. Work is ongoing to make code proofs simpler and more automatable with the use of separation logic. An earlier stage of this work was covered in an ITP 2019 paper (Mehta, Cook.)
As by-products of this work, various improvements have been contributed to the standard libraries, the Kestrel books and the ACL2 sources.

Improved Libraries

`projects/rac`: Restricted Algorithmic C.

- A new book, `books/projects/rac/lisp/internal-fns-gen.lisp`, which implements two tools that generate functions that compute values of local (bound) variables of an input function (to be used as described in Russinoff's workshop paper):
  - `CONST-FNS-GEN` is applicable to functions with non-recursive definitions.
  - `LOOP-FNS-GEN` can be applied to certain functions with restricted recursive definitions.
- Modifications of `books/projects/rac/src`: The RAC parser has been updated to check the placement restrictions on return statements in RAC programs.

Improved Libraries

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

- Allow Smtlink to handle functions using `cw`.
- Made it possible to use `:smtlink` hint inside an Smtlink proof.
- Added abstract type support for Smtlink.
- Fixed the problem of rewrite-loops by changing computed-hints structure in Smtlink.
- Fixed the **soundness bug** that `unary-/` can be interpreted as integer division in Smtlink.
- Added a link to an example using Smtlink for verifying an ASP* Pipeline. (This example will be moved to the ACL2 repository once we have the next stable version of Smtlink.)
- Made improvements to the documentation.

Improved Libraries

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

- Improved some aspects of the model of segmented memory.
- Added support for additional forms of the `MOV`, `SHLD`, and `SHRD` instructions.
- Improved the `definst` macro to generate more boilerplate code.
- Updates to program instrumentation utilities to allow logging (parts of) the x86 state to a specified file in both CCL and SBCL.

Improved Libraries

`rtl`: Register transfer logic library.

- The book `books/rtl/rel11/lib/srt.lisp` now includes a formalization of a radix-8 SRT square root algorithm.
- The book `books/rtl/rel11/lib/add.lisp` now includes a correctness theorem for a generalized leading zero anticipator that does not assume an ordering of its operands.

Improved Libraries

std/alists: Standard association list library.

- Added a function remove-assocs, which generalizes remove-assoc from single to multiple keys.
- Added functions alist-map-keys and alist-map-vals, which ignore shadowed pairs.
- Added some theorems.

std/basic: Standard basic library.

- Added recognizers bytep and nibblep for “standard” bytes and nibbles.
- Added pos-fix, a fixer for posp.

Improved Libraries

`std/lists`: Standard list library.

- The function `list-fix` has been made built-in, with the name `true-list-fix`.
- The `take-redefinition` theorem has been removed, because the built-in definition of `take` has been changed to be like that redefinition.
- Added a file with theorems about `union-equal`.

`std/strings`: Standard string library.

- Added a variant `strtok!` of `strtok`. It does not treat multiple contiguous delimiters like one.
- Added `printtree` library for efficiently composing large strings from pieces.

Improved Libraries

`std/typed-lists`: Standard typed lists library.

- Added lists of strings and symbols. (Originally in `system/pseudo-good-worldp.lisp`.)

`std/util`: Standard utility library.

- The `defthm-natp`, `defthm-unsigned-byte-p`, and `defthm-signed-byte-p` utilities have been moved here from the X86ISA model.
- The `use-termhint` utility has been moved here from `clause-processors/use-hint.lisp`.
- Added `defret-mutual-generate`, discussed in the paper *Generating Mutually Inductive Theorems from Concise Descriptions* at this workshop.
- Like `defun-nx`, `define` macro also disables the executable-counterpart of a non-executable function now.

Improved Libraries

system/pseudo-good-worldp.lisp: Factored out some reusable functions.

- Predicates for event forms, command forms, event landmarks, command landmarks, and tests-and-calls structures are now in separate files.
- More could be factored out.
- These could be perhaps integrated with Std/system.

xdoc/constructors.lisp: Made some improvements and additions.

xdoc/defxdoc-plus.lisp: Added `defxdoc+`, an enhancement of `defxdoc` that supports more concise expression of ordering of subtopics and default parents.