Note-8-3-for-talk

`Current Interactive Session`

ACL2 Version 8.3 (April, 2020) Notes

NOTE! New users can ignore these release notes, because the documentation has been updated to reflect all changes that are recorded here.

Below we roughly organize the changes to ACL2 since Version 8.2 into the following categories of changes: existing features, new features, heuristic and efficiency improvements, bug fixes, changes at the system level, Emacs support, and experimental versions. Each change is described in just one category, though of course many changes could be placed in more than one category.

Note that only ACL2 system changes are listed below. See also note-8-3-books for a summary of changes made to the ACL2 Community Books since ACL2 8.2, including the build system. Also note that with each release, it is typical that the value of constant `*ACL2-exports*` has been extended, and that some built-in functions that were formerly in `:program` mode are now guard-verified `:logic` mode functions.

Changes to Existing Features

It is now permitted for an evaluator to be ancestral in a metafunction or clause-processor. See evaluator-restrictions, or see the source code comment “Essay on Correctness of Meta Reasoning”, for discussion of the remaining restrictions. Thanks to Sol Swords for requesting this improvement and for many helpful discussions. Moreover, he found a bug in a proof in the above Essay, which has been been fixed; he made a key observation that led to completion of that fix. Also thanks to Rob Sumners for helpful discussions.

The accumulated-persistence utility no longer overcounts accumulated frames due to nested (recursive) rule applications. Although that shortcoming was documented, it was unfortunate and we thank Sol Swords for an email that nudged us into making this improvement and provided helpful insight, and Eric Smith for suggesting this improvement more than 10 years ago (!).

In a (new-style) signature, any symbol whose `symbol-name` is `"*"` now designates a non-stobj argument. Formerly, only the symbol `*` in the `"ACL2"` package could be used in that way. Thanks to Jared Davis for suggesting (in 2007!) that we consider such a change.

More functions can now be given warrants. In particular: the requirement of a natural number measure for recursive definitions has been relaxed to allow lexicographic combinations of natural numbers as defined by the `llist` function in the Community Books at `books/ordinals/`, and it is possible to warrant some functions that use local stobjs as long as they don't call `apply$`. See defwarrant.

The function `symbol-name-lst` is now a guard-verified logic-mode function (formerly it was a program-mode function). Thanks to Alessandro Coglio for suggesting that it might be good to document this function, which led us to this change (and to its being documented).

The pattern language for the macro, `case-match`, now includes the case `(quote~ sym)`, where `sym` is a symbol and this case matches any symbol whose `symbol-name` is `(symbol-name sym)`.

The following symbols, when used in special syntactic roles in the macro `loop$`, may be in any package: `for`, `in`, `on`, `from`, `to`, `by`, `of-type`, `when`, `until`, `sum`, `collect`, `always`, `thereis`, and `append`. Thanks to Mertcan Temel for requesting this enhancement.

`:Expand` hints now act more reliably for equality-variants, by expanding away guard-holders. Thanks to Sol Swords for supplying this enhancement and sending the following example that works now but formerly failed:

```
(defthm member-of-cons
  (equal (member x (cons x y)) (cons x y))
  :hints(("Goal" :in-theory (disable member)
          :expand ((member x (cons x y))))))
```

An analogous improvement has been made for `:by`, `:use`, and `:hands-off` hints.

The macros `defequiv`, `defrefinement`, and `defcong` now conform to the following principle discussed in a new documentation topic, packages-for-generated-symbols: ideally, utilities generate symbols in the `current-package`, at least by default. These three macros now have a `:package` keyword argument whose default value is `:current`; with this value the macros conform to the above principle. See their documentation topics. To get the previous behavior, use the `:legacy` value. (Another way to deal with failures caused by this change may be to fix packages when referring to generated symbols, such as changing `acl2::x-equiv` to `x-equiv`.) In particular, the `:legacy` option was used to update definitions in the community-books for macros that generate `defcong` forms. Also, all three macros now do some error-checking (rather than leaving that entirely to the generated `defthm` form), and the unsupported `:doc` keyword argument has been removed from these macros. Thanks to Pete Manolios for suggesting all of these changes, and for providing not only implementations but also modifications to the community-books.

The translation of a term `(and t u0)` is `(if 't u1 'nil)`, where `u1` is the translation of `u0`. That term, `(if 't u1 'nil)`, is now generally displayed to the user ("untranslated") as `(and t u0)`, but formerly it was displayed as `u0`, which could be confusing. Thanks to Stephen Westfold, who sent an example showing how, when using the proof-builder's `REWRITE` command to replace a subterm `a0` by `t` in a term `(and a b)` could lead to confusion, since the resulting `(and t b)` was printed only as `b`. Note: For Boolean contexts, the analogous change was also made for terms `(and u0 t)`.

It no longer causes an error to call `trans-eval` on an expression that references a locally-bound `stobj`, that is, one bound by `with-local-stobj` or `stobj-let`. The user is responsible for understanding that when calling `trans-eval`, all stobj variables in the supplied expression refer to globally-bound stobjs, that is, stobjs stored in the `user-stobj-alist` field of the ACL2 `state`. See the new documentation topic, trans-eval-and-locally-bound-stobjs, for relevant discussion. (Another topic, user-stobjs-modified-warnings, may also be helpful for understanding the interaction of `trans-eval` with stobjs.) Thanks to Sol Swords for suggesting this change and convincing us of its suitability.

The `defstobj` event now supports stobjs with fields that are hash tables in raw Lisp but are represented logically as association lists. Thanks to Sol Swords for providing not only the design but also the implementation, which was moved from community book `books/add-ons/hash-stobjs.lisp` into the ACL2 sources (with small modifications to both the new code and existing code). That book still provides lemmas that may be helpful for reasoning about hash-table fields, as well as some tests. See also `defstobj`.

Added suitable guards, with custom error messages, to `add-invisible-fns` and to `remove-invisible-fns`. Also removed confusing messages for each in the case of redundancy. Thanks to Pete Manolios for pointing us to a bug in the documentation for the latter (which we have fixed), which led us to the addition of guards.

The notion of *untouchable* macro is no longer directly supported. Specifically: the form `(push-untouchable SYM t)` is now illegal if `SYM` is already the name of a macro; and after this call of `push-untouchable` it is illegal to define `SYM` as a macro. However, a macro can be made effectively untouchable by defining it with the new utility, `defmacro-untouchable`. Note that the alleged support for untouchable macros was already incomplete, as explained in an example in the form `(deflabel note-8-3 ...)` in community-book `books/system/doc/acl2-doc.lisp`.

The event macro, `thm`, is now treated like `defthm` in the following way: if keyword `:hints` is supplied, then the hints are checked syntactically when skipping proofs (see ld-skip-proofsp) except during `include-book` or the second pass of `encapsulate`. For example, evaluation of the form `(thm (equal x x) :hints bad-hints)` now causes an error after evaluating `(set-ld-skip-proofsp t state)`, while before this change, it did not.

The default slow-alist-action (see slow-alist-warning) is now `:break` instead of `:warning` in ACL2. (It remains `:warning` in ACL2(p); see unsupported-waterfall-parallelism-features.)

For a user-defined `:induction` rule to be applied, it is no longer required for the induction scheme associated with a recursive definition to be enabled. For an example of the effect of this change, see the community-book, `books/system/tests/induction-rule-with-disabled-scheme.lisp`. Thanks to Pete Manolios for reporting this issue, including the sending of the events in that book. Also see induction (as that documentation has been updated).

An undocumented kind of fake-rune is no longer reported by `show-accumulated-persistence`. Thanks to Eric Smith for bringing this issue to our attention.

The algorithm for removing guard-holders has been modified to avoid diving into calls of `hide`. However, it is possible to obtain the former behavior; see guard-holders.

Since its earliest years, ACL2 uses evaluation to simplify ground terms (terms with no free variables). ACL2 would sometimes generate a call of `hide` around a term that fails to evaluate because of an attempt to call a constrained function. Now, that call incorporates a comment saying which constrained function is responsible for the failure. See comment. Also see hide for how to fix proof failures caused by this new behavior by using `:expand` hints or even by turning off this new behavior using `defattach`. Thanks to Rob Sumners (in 2003), Francisco J. Martin-Mateos (in 2004), and Anna Slobodova (in 2005), perhaps among others, for discussions leading to this enhancement.

Both `:puff` and `:puff*` have been made more robust. Related changes include:

- A new table, `puff-included-books`, generally prevents the same book from being puffed more than once.
- Book directories are tracked more carefully, which can prevent errors.
- We no longer allow `:puff*` to puff non-trivial `encapsulate` events. (For a technical discussion of reasons for this change, see comments in function `puffed-command-sequence` in ACL2 source file `ld.lisp`.)
- Changes to the `ACL2-defaults-table` no longer persist outside the scope of a puffed `encapsulate` event.
- A `theory-invariant` event no longer stores the event `(in-theory (current-theory :here))` in the world.

New Features

A new xargs keyword, `:guard-simplify` (default `t`), controls certain simplifications that may be applied to the guard conjecture while generating the initial goal. Setting it to `nil` skips all simplifications that depend on the set of currently enabled rules. See verify-guards. Thanks to Sol Swords for designing this feature and providing its implementation, along with documentation and corresponding adjustments to the community books.

Now `defstub` accepts the same keywords as `encapsulate`, both for the new-style signatures and for the old-style signatures. Thanks to Alessandro Coglio for suggesting and implementing this enhancement.

New function `(maybe-flush-and-compress1 name ar)` calls `(flush-compress name)` and then returns `(compress1 name ar)`, except that all this is skipped if the given array is already compressed.

A new utility, `swap-stobjs`, does what its name suggests: it swaps stobjs. Thus, if `st1` and `st2` are stobjs then after returning from execution of a call `(swap-stobjs st1 st2)` of `swap-stobjs`, the global value of stobj `st1` will be the old value of `st2` and the global value of stobj `st2` will be the old value of `st1`. See swap-stobjs. Thanks to Sol Swords for requesting this feature and for helpful discussions about it.

By invoking `(set-absstobj-debug :ignore)`, which requires an active trust tag (see defttag), one can defeat invariance checks for abstract stobj fields that otherwise are protected by specifying `:protect t` (or by using the `:protect-default` keyword to get that effect). See defabsstobj. Thanks to Sol Swords for suggesting consideration of adding some such capability. In a few preliminary tests we found an average drop of about 3.5% in the time it took to run the tests.

It is now permitted to define functions in which recursive calls occur from inside `loop$` statements. See `loop$-recursion`. Such functions do not automatically suggest induction schemes. Furthermore, care must be taken when formulating inductively provable theorems about such functions. See loop$-recursion-induction and `definductor`.

You can now change the second line in the startup banner for a GitHub version of ACL2 obtained between releases. See startup-banner. Thanks to Andrew Walter for requesting this enhancement.

Heuristic and Efficiency Improvements

ACL2 keeps a complete list of all the runes in the tau database (see introduction-to-the-tau-system). Formerly this list was duplicate-free, but that is no longer the case. As a result we have seen some faster performance; in particular, this change has cut 17% from the time to include the community book, `"centaur/sv/top"`.

Made slight efficiency improvement for table update (`:put`) events.

Improved efficiency of the maintenance of stobj-related arrays (the so-called stobj accessor arrays) by using the new function, `maybe-flush-and-compress1`. That code is related to printing stobj field accesses using field names rather than indices. (For background on printing untranslated terms, see term.) For example, after evaluating the form `(defstobj st fld)`, the form

```
(thm (equal (fld st) xxx)
     :hints (("Goal" :in-theory (disable nth))))
```

produces the (untranslated) goal `(EQUAL (NTH *FLD* ST) XXX)` rather than `(EQUAL (NTH 0 ST) XXX)`. A further change has been to reduce the frequency of ensuring that those arrays are up-to-date. (Technical note: this latter change is to source function `update-wrld-structures`, which has an explanatory comment, including an example of a 2.6% time reduction.)

Improved the speed of theory updates by avoiding repeated length computations. As a result, we have seen about a 5% time reduction on MacOS, and between 3% and 4% on Linux, for executing the form, `(include-book "centaur/sv/top" :dir :system)`.

The definition of the macro `ec-call` has been tweaked to speed up compilation in some cases when the host Lisp is SBCL. In particular, for admitting the definition of `apply$-prim` in community-book `books/projects/apply-model-2/apply-prim.lisp` with host Lisp SBCL, we have seen the time decrease from 1294.33 seconds to 8 seconds.

A heuristic for “lazy” rewriting of calls of `mv-nth` has been made much more efficient in some cases. Thanks to Sol Swords for investigating performance issues leading him to implement such a change, and for making modifications to community-books so that they continue to certify after the change. Those interested in implementation details may start with source function `simplifiable-mv-nth1`.

The raw Lisp representation of stobjs has been improved to avoid some indirection in two ways: a scalar field (one that is not an array or hash table) with non-trivial type had been wrapped in a one-element array, but no longer; and if there is only one field, and it is an array or hash table, then that field is the entire stobj. (The first change is however avoided when the host Lisp is GCL.) Thanks to Warren Hunt and Sol Swords for suggesting these changes. Technical note: in the course of making these changes, a bug was exposed in source function `raw-ev-fncall`; that has been fixed.

Reduced the computation and consing for theory management, which might reduce time by a percent or two when including some books. (Technical note. The idea was to expand the cases in which the use of `compress1` is replaced by more efficient code. That code is specific to enabled structures and may be found in ACL2 source function `update-enabled-structure-array`. We also arranged, in `load-theory-into-enabled-structure`, to double the array size when that exceeds the expansion by a minimal suitable multiple of 500.)

Changed how compound-recognizer rules are stored in the logical world (per function symbol, rather than in a single alist), which a few experiments suggest might reduce time by a couple percent or so.

A tweak was made to how properties are ordered when stored in the ACL2 logical world, which experiments show provides small speed-ups. (Technical note: the change is to constant `*current-acl2-world-key-ordering*`, and also, functions `symbol-class` and `logicp` avoid calling `getprop` for the symbol, `cons`.)

ACL2 now avoids summary calculations during `include-book`. We have seen this change cut more than 9% of the time for the event `(include-book "centaur/sv/top" :dir :system)`.

We now avoid translation of `default-hints` for termination proofs during `include-book`. We have seen this change cut more than 4% of the time for the event `(include-book "centaur/sv/top" :dir :system)`.

ACL2 now saves, in certificate files, the translated bodies of `defun` and `defthm` events. (See term for a discussion of translated terms.) This can speed up `include-book`; for example, we have measured approximately a 3% reduction in time for the event, `(include-book "centaur/sv/top" :dir :system)`, but with a space trade-off of about 41% more bytes allocated. (Implementation note: the relevant algorithms and code are discussed in an expanded version of the Essay on Cert-data in the ACL2 source code.) Thanks to Eric Smith for pointing out a bug (which we then fixed) in a preliminary implementation of this feature.

Some stack overflows may be avoided by a change to a built-in system function, `cons-count-bounded-ac`, so that it is now tail recursive as it CDRs the list, rather than as it takes the CAR. Thanks to Shilpi Goel for reporting the problem with an example and to Sol Swords for suggesting this fix. That example exhibited a second stack overflow, due to a built-in memoized system function that is no longer called when computing a call of the built-in system function, `pkg-names`.

The second pass of `encapsulate` now uses fast-alists when calculating new triples in the logical world (in ACL2 system function `new-trips`. We have seen this change result in cutting the time by 4.7% and the bytes allocated by 34% for including the community book, `"centaur/sv/top"`.

Computation of the guard proof obligation has been sped up in some cases involving evaluation of ground terms (terms without free variables). Thanks to Warren Hunt for sending us an example that prompted us to make this change.

Bug Fixes

As noted in the documentation for lemma-instance, ACL2 may avoid proving some constraints required for functional-instantiation that were previously proved. There was such support even in the case that the previous proof was done on behalf of a `defattach` event, but that support has been made more complete (by keeping more functional substitutions in canonical form).

A hard Lisp error has been fixed that could occur (probably only rarely) after adding rules of class `:definition` that introduce recursion.

The function `meta-extract-formula` could return a non-trivial value (i.e., not `'T`) when applied to a program-mode function. We thank Sol Swords for reporting this bug with a proof of `nil` that exploited it. This soundness bug has been fixed. That work led us to fix the following related bugs (which could also be soundness bugs, though we have not checked). First, the function `fncall-term` could similarly return a non-trivial value when applied to a program-mode function. Second, the utility `mfc-rw`, as well as other such `mfc-xx` utilities in support of extended-metafunctions and `meta-extract-contextual-fact`, could be called on terms containing program-mode function symbols.

Eliminated an error occurring when attempting to compute the guard proof obligation for a constrained function, in particular, when using the `:gthm` utility on such a function (also see guard-theorem). Thanks to Alessandro Coglio for pointing out this bug and for noting that `t` could be a reasonable result for the guard theorem in such cases.

Fixed ACL2 raw Lisp error caused by add-default-hints!. Thanks to Pete Manolios for debugging this problem.

Fixed a bug in the proof-builder command `apply-linear` (and its abbreviation, `al`), which was making it impossible to save an event after an interactive session that includes such a command. Thanks to Mihir Mehta for bringing this bug to our attention and sending an example.

Fixed a bug in defun-sk, which was generating a verify-guards or verify-guards? event with :guard-hints instead of :hints. Thanks to Alessandro Coglio for reporting this bug and providing the fix.

The use of :stack :pop in the macro with-output failed to restore gag-mode properly. For example, the use of (with-output :stack :push :gag-mode nil (with-output :stack :pop <form>)) failed to run <form> with the existing value for gag-mode (default: :goals). Thanks to Mihir Mehta for a query that led us to make this fix. Technical note: state global inhibit-output-lst-stack is now a list of pairs (inhibit-output-lst . gag-mode); see with-output.

The proof-builder's DV command was broken for expressions of the form (if t term1 term2); for example, (verify (if t x y)) followed by 1, 2, or 3 caused a raw Lisp error. Thanks to Stephen Westfold for bringing this bug to our attention and pointing out the fix.

We fixed a bug in the implementation of encapsulate that could cause a hard ACL2 error ("Unexpected expansion-alist ... for second pass of encapsulate"). Thanks to Pete Manolios for sending an example that exhibited this bug. (A slightly simplified version of his example may be found in a comment in the definition of function encapsulate-pass-2, ACL2 source file other-events.lisp.)

Defstobj now provides a suitable error message, instead of an implementation error, when new names are duplicated after renaming. Here are examples that now have improved error messages.

```
(defstobj st x :renaming ((create-st x)))
(defstobj st fld :renaming ((fld create-st)))
(defstobj st fld1 fld2 :renaming ((fld1 fld) (fld2 fld)))
```

We fixed an obscure error message when attempting to memoize either IF or RETURN-LAST, and we improved the memoize documentation to mention these and other restrictions on what can be memoized.

Fixed erroneous type declarations in array copying functions. (Technical notes. (1) It's not clear that these bugs have ever had any effect. (2) Those source code functions have been renamed by dropping their "stobj-" prefixes from stobj-copy-array-xxx, now that one of them has application to other than stobjs, in new code mentioned above for enabled structures.)

Fixed the process of translating the type-spec, standard-char, into a term. For example, the following definition failed but now succeeds:
(defun foo (x) (declare (type standard-char x)) (cons 3 x)).

Fixed a bug that could cause the wrong stobj to be displayed by `print-gv`, when congruent stobjs with a single bit-array field are involved. See a comment in ACL2 source function `apply-user-stobj-alist-or-kwote` for an example of this bug.

When including a book that is uncertified because of a stale certificate file, ACL2 was inappropriately using information about type-prescription rules (specifically, those computed by the system at definition time) that was stored in that certificate. This has been fixed, thanks to a bug report from Eric Smith about a related feature mentioned above, on saving translated bodies in certificate files. Moreover, the relevant system function, `include-book-fn1`, has been modified to do a better job of ignoring certificate files of uncertified books.

The utility `set-saved-output` has been badly broken for years, but without complaints (other than from one of us shortly before the release), suggesting that it hasn't been directly called by users. So we have eliminated it.

Changes at the System Level

The makefile target `certify-books` has been removed from `GNUmakefile` and `books/GNUmakefile`. It was deprecated in the preceding release, where we thanked the acl2-books email list (in particular we got feedback from Alessandro Coglio, Shilpi Goel, David Rager, Eric Smith, and Sol Swords, all helpful) for working through this issue.

The keyword `:ACL2` is now a member of the Lisp global, `*features*`, which allows other programs to use read-time conditionals `#+acl2` / `#-acl2` to indicate the presence or absence of ACL2. Thanks to Andrew Walter for suggesting that this might be useful, for example for Quicklisp code.

EMACS Support

The command '`Ctl-t p`' now works in Emacs 25.

The ACL2-doc browser for ACL2+books documentation can be extended with a new command, `U`, to open a URL (or in some cases, a file) in a browser. See file `emacs/acl2-doc-open-url.el` for more information.

Experimental Versions