# Note-8-2-for-talk

Current Interactive Session

ACL2 Version 8.2 (May, 2019) 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 8.1 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-8-2-books for a summary of changes made to the ACL2 Community Books since ACL2 8.1, 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

The built-in function `fix-true-list` is now a macro that expands to a new built-in function, `true-list-fix`, whose definition follows the efficient definition of list-fix that was in community-book `books/std/lists/list-fix.lisp`. In that book, list-fix is now a macro that expands to `true-list-fix`. The use of macro-aliases (see add-macro-alias) should generally make this change backward compatible for users of `list-fix`. Thanks to Mihir Mehta for taking the lead on implementing these changes and to Jared Davis for permission to integrate definitions and documentation from his Kookamara books into the ACL2 sources.

Made some improvements pertaining to `apply$`:

- A quoted lambda object that may ultimately be passed as the “function” for a call of `apply$` may now have a `declare` form. See also the discussion of `lambda$` below.
- The macro `warrant` no longer causes an error for the 800+ ACL2 primitives that are built into the definition of `apply$`. Instead, it simply avoids generating (needless) conjuncts for those primitives.
- The rewriter can now evaluate ground terms that involve calls of `apply$` or `badge` on user-defined function symbols. Note that correctness of such an evaluation

depends on the truth of corresponding warrants, which will be forced if not known.

- The event formerly named `def-warrant` is now `defwarrant`. This event may now be redundant; hence `defun$` may also be redundant.
- The `:args` command now prints the badge and warrant for a function, and it avoids printing a package prefix in the case of unknown-constraints.
- The optimization for caching “tame compliant” lambdas, introduced in the preceding release (see note-8-1), has been improved.

It is no longer illegal to supply an abstract stobj as the so-called “concrete stobj” in a `defabsstobj` event. Thanks to Sol Swords for initiating a discussion leading to this enhancement.

Calls of the function `synp` were formerly required to result from macroexpansion of `syntaxp` or `bind-free` calls, or at least nearly so. That restriction has been lifted, although the restrictions on calls of `synp`, `syntaxp`, and `bind-free` remain for hypotheses of rules of class `:rewrite`, `:definition`, and `:linear`, or resulting from evaluation of hypotheses of `:meta` rules. Thanks to Sol Swords for requesting this change, so that `defevaluator` forms can include `synp`.

For calls of `:pso` and related utilities (`:pso!`, `:psof`, and `:psog`), the Time reported in the summary is now the original time, not time related to running `:pso`.

The macro `delete-assoc` has been renamed `remove1-assoc`, to reflect more clearly that at most one pair is removed, and also for consistency with Common Lisp nomenclature (where “delete” operations are generally destructive and “remove” operations are not). Moreover, other functions and macros whose name has prefix `"DELETE-ASSOC"` have been similarly renamed to have prefix `"REMOVE1-ASSOC"`; for example, `delete-assoc-eq` has been renamed `remove1-assoc-eq`. (The old `"DELETE-ASSOC"`-based names still exist as macros — indeed, as macro-aliases for their renamed versions (see add-macro-alias) — but those may be deleted in the future.) Analogous functions and macros have been introduced that remove all pairs with a given key, rather than only one; see remove-assoc. (These and related theorems formerly appeared in the community-book, `books/centaur/misc/remove-assoc.lisp`.) Thanks to Alessandro Coglio for a query and subsequent discussion leading to these changes.

The system-utilities `all-ffn-symbs` and `all-ffn-symbs-lst` are now defined just as macro abbreviations for calls of system utility `all-fnnames1`, thus eliminating some source code duplication. We may deprecate `all-ffn-symbs` and `all-ffn-symbs-lst` in the future.

The implementation of verify-termination has been improved so that it no longer can generate (expand to) the form (value-triple :redundant). Redundancy is now handled for verify-termination by checking redundancy of the generated defun form. For an example that failed before this change, see community-book books/system/tests/verify-termination/top.lisp.

Improvements have been made to the summary printed on conclusion of an event. It had been possible to have duplicates in the "Hint-events" field of the summary; that has been fixed. Also, the handling of :instructions within hints has changed in the following two ways, to be consistent with the use of :instructions at the top level (rather than within :hints). The "Rules" and "Hint-events" fields of the summaries incorporate information from calls of the proof-builder. Also, the "Hint-events" field no longer contains (:CLAUSE-PROCESSOR PROOF-BUILDER-CL-PROC).

A new variable, TRACE-LEVEL, may be used in calls of trace$; see trace$. This replaces the use of the state global variable of the same name, which has been eliminated, thus avoiding an error involving TRACE-LEVEL that is mentioned below.

Defun-sk is now sensitive to the default-verify-guards-eagerness, and guard verification is always delayed to near the end of the generated event to avoid failures due to the small theory present at defun time. Thanks to Alessandro Coglio for emails on leading to these improvements. Some additional small tweaks have been made to defun-sk, in particular to check that there are not two or more distinct values associated with xargs keywords :verify-guards, :non-executable, or (even if not distinct) :guard-hints.

The function integer-range-p now uses a type declaration in place of the :guard, which may slightly improve efficiency. Thanks to Eric Smith for suggesting this possibility.

The macro, thm, may now be used in event contexts, in particular in calls of encapsulate and progn and in books. (See embedded-event-form.) Thanks to Ruben Gamboa for an email that led to this change.

The :pf command now does a more complete job of showing induction schemes for induction rules. (Some corresponding code cleanup has also been done.)

Warnings about using enabled rules have been improved. Now, when using an executable-counterpart rule (which, admittedly, is unusual; it is equivalent to using the corresponding definition rule), the warning will correctly recommend disabling the definition rule instead of the executable-counterpart rule.

The logical definition of read-file-into-string2 (in support of read-file-intostring) has been simplified, and no longer involves untouchable functions symbols. Thanks to Mihir Mehta for a query that led to this enhancement.

The built-in function take now has a recursive definition, exactly along the lines of the old theorem `take-redefinition` from the community book `books/std/lists/take.lisp` (written by Jared Davis); this theorem has now been removed. The definition of `take` uses mbe, where the `:exec` component calls `first-n-ac` as before for execution efficiency. We thank Mihir Mehta for providing this enhancement, including updates to the books.

## New Features

A new macro, loop$, is an ACL2 version of the Common Lisp `loop` macro.

A new construct, `lambda$`, may be used in place of `lambda` to be passed as the "function" for a call of apply$. The syntactic requirements for such uses of `lambda$` are much less strict than for quoted `lambda` objects; in particular, the body need not be in translated form (see term).

A new macro, partial-encapsulate, allows one to introduce constrained functions without specifying all of the constraints. This functionality was already available using a trust tag, by way of a rather convoluted application of define-trusted-clause-processor; however, partial-encapsulate may be used without a trust tag. See partial-encapsulate, which in particular points to an example of typical usage, in `books/demos/partial-encapsulate.lisp`. Thanks to Sol Swords for requesting such a capability.

There is now, by default, a limit of 9 on the nesting depth of inductions; see induction-depth-limit and to modify this default, see set-induction-depth-limit.

A new evisc-tuple, the brr-evisc-tuple, controls printing inside the break-rewrite loop. See brr-evisc-tuple and set-evisc-tuple.

Many of the brr-commands now abbreviate ("eviscerate") by default using the new brr-evisc-tuple (see above), and for each for those a corresponding command with suffix "+" prints in full. For example, such commands include `:path` and `:path+`; see brr-commands for the full list of commands. Thanks to Stephen Westfold and others at the 2018 Developer's Workshop for discussing this issue.

A new signature is legal for clause-processors, to support the return of rules and event names to be printed in the summary. See make-summary-data.

Apply$ now handles functions that return multiple values. This has widespread ramifications. The structure of badges has changed. There is no longer an "authorization-flag" and there is now an "out-arity" slot in the badge. See badge. Every badged non-primitive function symbol now has a warrant. If `fn` is a warranted function symbol and returns more than one result then `(apply$ 'fn ...)` returns a list of the results, just as `fn` does in the logic. See `apply$`. The warrant of a multi-valued function is just like that for a single-valued function except that `mv-list` is used to coerce the output of the multi-valued function to a list. See `warrant`. All `LAMBDA` objects and `lambda$` expressions must be single-valued, but can use multi-valued functions to compute that value.

## Heuristic and Efficiency Improvements

`Make-event` expansions have often been reduced in size. (Technical note: `record-expansion` has been inserted only on expansions done directly under `encapsulate` events, and some changes have also been made in how and when local events are elided from expansions.) The reduction in total sizes of `.cert` files for a complete ("everything") regression was 8.8%, though in some cases the reduction was substantially larger: for example, the size of `books/centaur/fty/tests/deftranssum.cert` was reduced from 22,474,113 bytes to 16,910,530 bytes, a reduction of nearly 25%.

A tweak to the rewriter can significantly speed up the use of hypothesis-free meta rules on large terms. Thanks to Mertcan Temel for sending an example that motivated this change, whose time was cut from 67 seconds to 19 seconds.

Some small optimizations have been made for the generation of executable-counterpart (so-called "*1*") code (see evaluation).

It has long been the case that certain prover routines, including handling of output from meta functions, transformed results into so-called "quote-normal form", where for example the term `(cons '3 '4)` is replaced by `(quote (3 . 4))`. Now, that transformation avoids recurring inside calls of `hide`. We thank Mertcan Temel, who had a class of examples that motivated this change. One such example took 856.27 seconds of 'prove' time before this change, but only 270.14 seconds after this change, thus eliminating 68.5% of the time.

Proofs involving very large terms could be slowed down by checking those terms for calls of `if`, in support of reporting splitters of type if-intro. That check is now limited by avoiding subterms that are calls of `hide`. Thanks to Mertcan Temel for supplying examples, one of which exhibited a proof time of 302.06 seconds that was reduced to 123.57 seconds with this change, and thanks to Sol Swords and Alessandro Coglio for helpful comments on possible enhancements.

## Bug Fixes

Fixed a bug, probably a soundness bug (though we haven't tried to prove `nil` by exploiting it). The bug is in the computation of the "immediate-canonical-ancestors" of a function symbol, which is used in the implementations of memoization and `defattach`, as well as in interactions between attachments and both `defaxiom` events and `:meta` rules. Thanks to Sol Swords for pointing out this bug and presenting a helpful example.

Fixed three proof-builder bugs:

- Fixed the proof-builder command, `dv` (see ACL2-pc::dv), for diving into calls of `list` and `list*`.
- Fixed a bug in the proof-builder command, `geneqv`. Thanks to Shilpi Goel for reporting this bug with an example.
- The proof-builder numeric "diving" commands `1`, `2`, `3`, etc. — and more generally, the `dv` command — were broken when the current subterm is of the form `(if 't .. ..)`. This has been fixed. Thanks to Keonho Lee for reporting this bug.

Eliminated a hard error labeled as "Implementation error" that could occur when submitting a `:congruence` rule during the second pass of `encapsulate` or the local incompatibility check in Step 3 of `certify-book`. The error occurred when the equivalence relation of the rule had been defined locally, hence was missing during that second pass or local incompatibility check. Now, a useful ordinary ("soft") error occurs, with a useful message. Thanks to Nathan Guermond for reporting this bug with a helpful example.

It had been possible to enter an infinite loop after certain errors involving wormholes and state global variables; now, a clean error occurs instead. The specific error motivating this change involved the combination of both tracing certain system functions (for example, `pop-accp-fn`) and calling `accumulated-persistence`. That specific error has been eliminated by the change to `trace$` involving variable `TRACE-LEVEL` that is mentioned in an item above.

Fixed a bogus error produced by `defchoose` forms containing unused variables with `ignorable` declarations. Also eliminated an extra warning in the case of more than one bound variable with at least one of them unused, which could occur after `(set-ignore-ok :warn)` has been evaluated. Thanks to Sol Swords for finding these bugs and for supplying code that we installed to fix them.

Fixed an inefficiency in book certification due to calling `fast-alist-free-on-exit` on the wrong objects. Thanks to Sol Swords for pointing out this problem.

The `make TAGS` command could fail to do a proper check that the `etags` program is installed, resulting in a failure when attempting to build an ACL2 executable. This has been fixed. Thanks to Johannes Altmanninger for reporting this problem in GitHub Issue #955.

When a function that calls `apply$` (or `apply$-userfn`, `badge`, or `badge-userfn`) has been memoized with a non-`nil` value for argument `:aokp`, its memo table may need to be flushed when removing a badge. (Such removal typically occurs by undoing a call of `defun$` or `defwarrant`, perhaps because they are local to an `encapsulate` event or to a book.) However, such flushing was not being done. That has been fixed. See new community-book `books/system/tests/apply-with-memoization.lisp` for examples. Among the changes made to the source files that are related to this fix: the “Essay on Memoization with Attachments” has been enhanced to discuss the implementation of such flushing; and functions `doppelganger-apply$-userfn` and `doppelganger-badge$-userfn` (formerly called `concrete-apply$-userfn` and `concrete-badge$-userfn`), which are still not advertised, are now introduced with `partial-encapsulate` (hence have unknown constraints) and are now untouchable.

## Changes at the System Level

The documentation topic, system-utilities, is now about only utilities that pertain to the ACL2 system implementation, rather than arbitrary built-in utilities. Thus, each of the following now has its own topic, rather than being described in system-utilities. (Thanks to Alessandro Coglio for suggesting this reorganization.)

- `alist-keys-subsetp`
- `alist-to-doublets`
- `cons-count-bounded`
- `evens`
- `keyword-listp`
- `merge-sort-lexorder`
- `odds`
- `packn`
- `packn-pos`
- `pairlis-x2`
- `pairlis-x1`

Documentation pertaining to `apply$` and related topics has been extended significantly.

(GCL only) Eliminated compiler output (by setting GCL raw Lisp variables `*compile-verbose*` and `*load-verbose*` to `nil`).

A new documentation topic, efficiency, suggests some ways to speed up proofs and evaluation. The ACL2 community is encouraged to extend (and more generally, improve) this topic!

(LispWorks only) Bytes allocated are now reported in LispWorks (formerly, only in CCL and SBCL) by `time$` and `memsum`.

A new documentation topic, make-event-example-3, explains the new implementation of `thm`, thus providing insight into several common implementation techniques used with `make-event`.

A new documentation topic, rule-classes-introduction, provides a basic guide to which sorts of rules to create from your theorems. Thanks to Mihir Mehta for encouraging the development of this topic.

Fixed “`make STATS`” (which is invoked by “`make DOC`”) to generate a result file `doc/acl2-code-size.txt` that gives accurate statistics on code size on Linux. (The relevant `grep` commands in file `doc/create-acl2-code-size` needed the `-a` option on Linux.)

Updated the CCL installation instructions formerly at :DOC ccl-updates. The topic is now ccl-installation. This now replaces the installation instructions file `installation/ccl.html`; thus, the ACL2 community is now welcome to improve these instructions. Thanks to Eric Smith for discussion that led to this change and to Keshav Kini and Alessandro Coglio for helpful feedback.

The makefile target `certify-books` has been deprecated in both `GNUmakefile` and `books/GNUmakefile`. Thanks to 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.

A message is now printed at when loading file `~/acl2-init.lsp` at startup.

## EMACS Support

Fixed the ACL2-doc browser so that it can handle topic names with the single-quote (`'`) and comma (`,`) characters, by escaping them.

Fixed Emacs support for the the proof-builder dive command (see ACL2-pc::dive), `control-t control-d`, to eliminate trailing zeros, since those are (and have been) disallowed by that command.

A new ACL2-doc command is the question-mark character (?), which goes to a page with one-line command summaries. Thanks to Warren Hunt for a request leading to this enhancement.

## Experimental Versions