Note-8-6

ACL2 Sources

ACL2 Version 8.6 (xxx, 20xx) 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 8.5 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-8-6-books for a summary of changes made to the ACL2 Community Books since ACL2 8.5, including the build system.

DARPA Note. The following statement applies to items below that are marked with "See DARPA Note above": Release was approved by DARPA with "DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited."

Changes to Existing Features

<<<<<<<<<<<<<<<<<<<<<<< (1) <<<<<<<<<<<<<<<<<<<<<<<

:Logic mode for fmt, (er soft ...), etc.

Some built-in functions that were formerly in `:program` mode are now guard-verified `:logic` mode functions. Among the most significant such built-in functions are the following.

- `One-way-unify` (see `books/system/brr-near-missp.lisp`)
- `Genvar` (see `books/system/brr-near-missp.lisp`)
- `Fmt`, `error1` (which supports macros `(er soft ...)` and `er-soft`), and related printing utilities — and some code was modified to support their conversion to `:logic` mode [See DARPA Note above.]

Note that because of the `error1` change noted above, `(er soft ...)` can now be used

in `:logic` mode code.

>>>

<<<<<<<<<<<<<<<<<<<<<<<< (2) <<<<<<<<<<<<<<<<<<<<<<<<

The cbd works for file operations other than just book operations.

The connected book directory (that is, the cbd) now elaborates relative pathnames to absolute pathnames not only for book operations, but for all file operations. For example, `(open-input-channel "foo" :character state)` now interprets filename `"foo"` relative to the cbd, where formerly it was generally interpreted relative to the directory in which ACL2 was invoked. (Technical note: ACL2 accomplishes the new behavior by arranging that `set-cbd` modifies not only the cbd but also the Lisp global, `*default-pathname-defaults*`.) Thanks to Alessandro Coglio, Eric McCarthy, and Eric Smith for suggesting consideration of such a change and for helpful discussions.

>>>

The function `hons-enabledp` is no longer defined, and `:hons` has been removed from the Lisp global, `*features*` (so, readtime conditionals `#+hons` and `#-hons` should be avoided, especially since `#+hons` is always false even though the system is hons-enabled). These were both deprecated in the preceding release (ACL2 Version 8.5). Note that the hons-enabled features of ACL2 have been included in all builds by default since Version 7.0 (January, 2015) and in all builds since Version 7.2 (January, 2016).

The proof-builder now takes into account various aspects of rewriting specified by the logical world that it formerly ignored (pertaining to match-free, case-split-limitations, untouchable functions, non-linear arithmetic (see set-non-linearp), the backchain-limit for rewriting, and the rw-cache-state).

The utilities `set-inhibit-er-soft`, `set-inhibit-er-soft!`, `toggle-inhibit-er-soft`, and `toggle-inhibit-er-soft!` have been renamed by dropping the suffix "`-soft`", hence they are now the following, respectively: `set-inhibit-er`, `set-inhibit-er!`, `toggle-inhibit-er`, and `toggle-inhibit-er!`. The relevant table has similarly been renamed from `inhibit-er-soft-table` to `inhibit-er-table`. These changes reflect their relevance for the new utilities, `er-hard` and `er-hard`, in addition to `er-soft`.

The macro `warrant` now `force`s the warrants listed.

:Induction rules now support the use of syntaxp hypotheses.

The utility read-file-into-string has been improved in the following ways.

- The value of the :start argument may now be any natural number less than the length of the input file. (Formerly one needed to use this utility to read the preceding bytes first, which can be much slower.) Thanks to Eric McCarthy, Eric Smith, and Grant Jurgensen for requesting this improvement and for helpful discussions.
- While the default behavior is the same for when the corresponding Lisp stream is closed, a new keyword argument, :close, can be supplied to control that behavior.
- Miscellaneous clean-up has been made in the implementation.
- Restrictions have been tightened a bit to avoid what could be considered a soundness bug. See discussion about that in the section on "Bugs" below.

The trace$ option :evisc-tuple :print, which continues to use raw Lisp printing, has undergone the following improvements when printing entry and exit values. Thanks to Eric McCarthy for a query that led to these improvements.

- Values are now pretty-printed.
- Array values are no longer displayed as stobjs. (This includes stobjs themselves, since they are arrays.)

The use of (:executable-counterpart rewrite-lambda-modep) to control the behavior of lambda object rewriting by the prover (see rewrite-lambda-object) has been elaborated.

Warnings for non-recursive functions in left-hand sides of rewrite rules (similarly for linear, forward-chaining, and type-prescription rules) now consider bodies of lambda objects that could be rewritten (see rewrite-lambda-object).

The output is more informative when :pr is applied to an undefined primitive, such as car or binary-+, or a macro-alias for one of those, such as +. Thanks to Eric Smith for pointing out odd output for examples like :pr binary-+.

The definition of pseudo-termp has been simplified by dropping a superfluous true-listp check. Thanks for the suggestion from Eric Smith. A new lemma, pseudo-termp-consp-forward, has been added to prevent some existing proofs from failing due to the

change.

The failure message regarding useless-runes (see useless-runes-failures) has been restricted to the case that a proof was attempted by the event (though there may be rare exceptions). Thanks to Eric Smith for requesting such a change.

Certain guard proof obligations for DO loop$s (so-called special conjectures (e), (f), and (g)) have been combined into one conjecture with a conjunction of three conditions in the conclusion to speed up proofs. See the comment starting with "Special conjectures (e), (f), and (g)" in the function special-conjectures in the source file history-management.lisp.

The rewriter has been changed to handle certain calls of ev$ faster. See rewriting-calls-of-apply$-ev$-and-loop$-scions.

A few new lemmas have been added to the standard apply$ book to simplify applications of assoc-equal-safe faster.

<<<<<<<<<<<<<<<<<<<<<<< (3) <<<<<<<<<<<<<<<<<<<<<<<

I'm highlighting this item on inhibiting certain errors to make it clear that I'm open to making it possible to suppress more error messages.

Time-limit and theory-invariant errors may now be inhibited much as step-limit errors, by using (set-inhibit-er "Time-limit") or (set-inhibit-er "Theory"), respectively. Thanks to Eric Smith for requesting this enhancement and its use in the implementation of the utility, prove$.

>>>

<<<<<<<<<<<<<<<<<<<<<<< (4) <<<<<<<<<<<<<<<<<<<<<<<

The new WALKABOUT command, CMDS, gives me an excuse to mention WALKABOUT. (A separate item below mentions the new WALKABOUT command, UP.)

A new command, (cmds c1 c2 ... cn), has been added to walkabout.

>>>

The documentation for redundant-events includes the following, which however was not enforced when the new event is in :program mode; that has been fixed.

```
4. If either the old or new event is a mutual-recursion event, then
   redundancy requires that both are mutual-recursion events
   that define the same set of function symbols.
```

A new feature, called “transparent” functions, can allow one to avoid the restriction on a rule of class `:meta` or `:clause-processor` that there are no common ancestors of its evaluator and meta function. See evaluator-restrictions for relevant background, and see transparent-functions for documentation of the new feature. Thanks to Mertcan Temel for requesting a way to work around that restriction, and thanks to Sol Swords for suggesting (and naming) the notion of transparent functions. We are also grateful to Sol for providing a very helpful sketch of a correctness proof.

When there is an attachment to a common ancestor of the evaluator and meta function of a proposed rule of class `:meta` or `:clause-processor`, the resulting error message now includes ancestor paths leading from the evaluator or meta function to a common ancestor.

A new `walkabout` command, `up`, moves up a level and exits the `walkabout` loop when already at the top level. It is thus equivalent to the existing command, `0`, which is still supported although `up` is highlighted in the documentation; see walkabout). Thanks to Eric Smith for suggesting `up`.

<<<<<<<<<<<<<<<<<<<<<<< (5) <<<<<<<<<<<<<<<<<<<<<<<

The next item gives me an excuse to remind you about iprinting, which now works better with break-rewrite and cw.

Arranged that iprinting that takes place during break-rewrite or `cw` is better reflected outside break-rewrite. For examples, see community-books input file `books/system/tests/iprint-and-brr-input.lsp`, which contains comments on problematic interactions between break-rewrite and iprinting in Version 8.5, and which generates (via the `run-script` utility) the output in file `iprint-and-brr-log.txt` in that directory. Note that `read-object` now updates data structures so that it can read back in what was just printed by `cw`; see a comment in the definition of `read-object` in the ACL2 source code.

>>>

<<<<<<<<<<<<<<<<<<<<<<< (6) <<<<<<<<<<<<<<<<<<<<<<<

I mention the following because I’d forgotten that OPTIMIZE is supported as a DECLARE

form in ACL2.

When a defined function has a declare form with `(optimize ...)`, that is now included in a declare form of the executable-counterpart function (see evaluation), which had not been the case.

>>

It had been the case that for type declarations of `flet` definitions in a surrounding `defun` form, they were dropped in the `defun` form's executable-counterpart (see evaluation). Now they are included.

The behavior of `pso` and related utilities (`pso!`, `psof`, and `psog`) has been modified to avoid introducing warnings that were not originally printed. For example, the output generated by `:pso` below no longer prints the warning that had been suppressed for the `defthm` event below.

```
(set-inhibit-output-lst '(warning proof-tree))
(defthm foo t
  :hints (("Goal" :use car-cons))
  :rule-classes nil)
:pso
```

The above change has additional, small output effects, probably for the better. For example, output from the form `(with-output :off (error summary) (thm (equal x y)))` no longer prints the line shown below (at the end).

```
ACL2 Error [Failure] in ( THM ...):  See :DOC failure.
```

<<<<<<<<<<<<<<<<<<<<<<<<<< (7) <<<<<<<<<<<<<<<<<<<<<<<<<

There have been several improvements involving loop$ — iteration in ACL2 is cool!

When an event fails, then if it involves definition runes for `loop$` scions, the failure message may suggest including the book `projects/apply/top` if it hasn't already been included. That book provides quite a few lemmas about `loop$` scions.

>>

Replaced `length` calls in the defun of `pseudo-termp` with calls of a new macro, `len$`, which is a call of `mbe` that invokes `length` in the `:exec` code and `len` in the `:logic` code. Thanks to Eric Smith for requesting such an enhancement, and for discussing

specifics of it, so as to avoid the need for at least one unfortunate rule that if (pseudo-termp term) then (not (stringp (cdr term))), apparently needed because length behaves specially on strings.

When there is an error from evaluation of a form encountered by ld, in a session where the value of ld-error-triples is the default of t and the value of ld-error-action is of the form (:EXIT N), then ACL2 quits with exit status N in some cases where formerly it did not. The following explanation is rather technical; see ld-error-action for relevant background.

> This behavior was already present in the case that the "error on evaluation" was from an evaluation result (mv erp val state) where erp is non-nil; but it has been extended to the case that erp is nil and val is of the form (:STOP-LD . x), as is returned by default by ld upon an evaluation error. A key effect of this change is for the case that a .acl2 file produces an error from a call of build::cert.pl. The following example illustrates; explanation follows below.
>
> ```
> ;;; foo.acl2
> (ld '((defun g (x) y)) :ld-error-action :return!)
>
> ;;; foo.lisp
> (in-package "ACL2")
> ```
>
> Before this change, the command 'cert.pl foo' resulted in a hard Lisp error (as seen in foo.cert.out). To see why, first note that cert.pl executes a sequence of commands as follows (several omitted as shown with "...").
>
> ```
> ...
> (set-ld-error-action (quote (:exit 1)) state)
> ...
> ; instructions from .acl2 file foo.acl2:
> (ld '((defun g (x) y)) :ld-error-action :return!)
> ...
> #!ACL2 (set-ld-error-action (quote :continue) state)
> ...
> ```
>
> The call of ld above returns (mv nil (:STOP-LD 2) state). Because ld-error-action at the top level no longer has the default value of :CONTINUE, that result is considered an error (see ld-error-action) and top-level

evaluation halts. Before this change, then ACL2 did not quit since the value was of the form (mv nil _ state); instead, ACL2 would quit the top-level call of ld, leaving us in raw Lisp. But in raw Lisp, the #! reader macro (see sharp-bang-reader) is undefined; hence an error would be signalled. After the fix, the return value of (mv nil (:STOP-LD 2) state) is treated as an error, so because ld-error-action is (:EXIT N), ACL2 immediately exits with status N.

The pretty-printer has been improved by a contribution from Stephen Westfold to support appropriate indentation, including more conventional pretty-printing for calls of common macros such as defun and defmacro. See ppr-special-syms; we thank Stephen also for supplying the substance of that documentation. Thanks too to Stephen for suggesting several user-defined macros to be pretty-printed with this mechanism, which we have modified by adding suitable table events (e.g., for define).

Runtime guard violation messages from DO loop$ expressions are now much more readable. They also now include a pointer to the do-loop$ documentation, which has new, relevant explanation (first in brief, later in detail) regarding such messages.

Four obsolete fields of the ACL2 state have been removed: big-clock-entry, t-stack, 32-bit-integer-stack, and list-all-package-names-lst, as have some related built-in, undocumented definitions and theorems, including old-check-sum-obj and supporting functions.

Improved hide calls in prover output from failed execution of warrants, by adding suitable notes about attachments or warrant functions not being executable during proofs. Thanks to Eric McCarthy for a remark that led to this change.

State globals inhibit-output-lst, inhibited-summary-types, and ld-level are now untouchable. The macros set-inhibit-output-lst and set-inhibited-summary-types still allow you to modify the values of these variables. Thanks to Peter Dillinger for correspondence (years ago!) leading to these changes.

<<<<<<<<<<<<<<<<<<<< **(8)** >>>>>>>>>>>>>>>>>>>>

The next item is kind of obscure unless you do ACL2 programming with state. It's an example of making the system more convenient to use. The system continues to mature.

The macro state-global-let* no longer requires explicitly supplying a setter for certain built-in state global variables. See state-global-let*.

>>>

A proposed `defaxiom` event is no longer redundant with an existing `defthm` event. Before this change, the following book could (perhaps unfortunately) be certified, even without `certify-book` option `:skip-proofs-okp t`.

```
(in-package "ACL2")
(local (defthm foo (equal (car (cons x x)) x)))
(defaxiom foo (equal (car (cons x x)) x))
```

The prover sometimes reduces a goal without hypotheses (technically speaking, a one-element clause) to `nil` using type reasoning. This heuristic has not changed, but formerly there was no explanation given. Now ACL2 reports the rules (of class `:type-prescription`) that were used.

The default for `memoize` keyword argument `:verbose` has been changed from `t` to `nil`, which (by default) eliminates noise from the output.

When a proof attempt is halted so that it reverts to prove the original goal by induction, the top-level checkpoints are printed under the summary under the banner, "`Key checkpoints before reverting to proof by induction`". This was normally the case already, but not in the special case that the proof is eventually aborted either because a goal of `NIL` is produced or because proof by induction is not allowed (due to a `:DO-NOT-INDUCT` hint or an induction-depth-limit being exceeded). Thanks to Eric Smith for a chat that helped lead to this improvement.

<<<<<<<<<<<<<<<<<<<<<<<< (9) <<<<<<<<<<<<<<<<<<<<<<<

You can customize the error message from table guard violations.

For most built-in tables, error messages for guard failures have been improved, by use of a new macro that is also available to ACL2 users: `set-table-guard`. That utility may be used to set a table guard in a way that produces a user-friendly error message when the guard fails.

>>>

<<<<<<<<<<<<<<<<<<<<<<<< (10) <<<<<<<<<<<<<<<<<<<<<<<

I mention the following only to show you that we listen to feedback, even though it sometimes takes us a long time to act on it.

The prover may now print a parenthetical remark about "dropping false conclusion". That remark points to a new documentation topic, which provides explanation: see clause.

>>>

State global `trace-co` is now untouchable.

ACL2 now warns when the hypothesis of a type-prescription or forward-chaining rule is a call of `syntaxp` or `bind-free`, since these get no special treatment for such rules.

The utility `get-event-data` now returns data from the proof done on behalf of a `thm` event, rather than from the surrounding `make-event` call used for implementing `thm`. Thanks to Eric Smith for requesting this change.

In the case that a proof is interrupted after `(set-debugger-enable t)` and then aborted, ACL2 formerly printed failures in a special way, in particular showing the pstack. This is no longer the case: all event failures are printed in the same way.

A new key, `EVENT`, is available for `get-event-data`. Thanks to Eric Smith for a discussion leading to this addition.

Most ACL2 reader errors are now ACL2 hard errors rather than raw Lisp errors (as they were formerly). As a result, such errors can sometimes be "caught" by suitable programming. Thanks to Grant Jurgensen and Eric Smith for requesting this change. Note that input is still flushed in these cases, perhaps more thoroughly than before.

<<<<<<<<<<<<<<<<<<<<<<<< (11) <<<<<<<<<<<<<<<<<<<<<<<

We (mostly J) did a ton of work on break-rewrite.

`Break-rewrite`, `brr-commands`, `monitor`, and `unmonitor` have been radically changed but in a largely backwards compatible way. It is now possible to cause interactive breaks when a monitored rule is tried but fails to match the target. See `monitor`, in particular, see the "near-miss" break criteria in `monitor` for a discussion. Changes to `break-rewrite` largely just correct anomalous behavior. One significant behavioral change is that the list of monitored runes is locally bound by `break-rewrite`, so that while it can be changed in inferior breaks, when control returns to superior levels (and to the top-level) the list of monitored runes is unchanged. Related changes are discussed in the following three items. [See DARPA Note above for this work, including the following three items.]

- The brr-command :p! (see brr-commands) has been redefined so that in break-rewrite it is a no-op. (Actually, it did not always work as advertised in break-rewrite! Its behavior outside of break-rewrite is unchanged.) A comment in the source code definition of p! explains why.
- The function get-wormhole-status is deprecated, and its new name is get-persistent-whs.
- The macro show-brr-evisc-tuple has been eliminated, but brr-evisc-tuple is available instead.

>>

The definition of bounded-integer-alistp has been modified by adding a guard (posp n) and removing the (integerp n) test from the body of its defun. [See DARPA Note above.]

As before, when calling break$ from a wormhole — for example when inside break-rewrite — one is left at a raw Lisp prompt. However, when returning from that prompt (e.g., with :q if the host Lisp is CCL), one no longer stays in the wormhole, as noted by a message, “Aborting to top level from a wormhole break (see :DOC wormhole).” [See DARPA Note above.]

The function iprint-oracle-updates, which is called by read-object, is now disabled, because it was made more complicated to accommodate the conversion of function eviscerate-top into :logic mode. [See DARPA Note above.]

The utilities :pe and :pr now provide more useful output when applied to function symbols that are built into ACL2 without a defining event. Thanks to Warren Hunt for discussions leading to this improvement.

The guards and bodies have changed slightly for some built-in function definitions, as follows.

- For functions princ$, prin1$, explode-atom, and explode-atom+, the guard now requires of the first argument only that it be an atom; it no longer must be a “good atom”, i.e., a number, a character, a string, or a symbolp. As a result, the definition bodies of princ$, prin1$, and explode-atom have been tweaked slightly.
- The guards for packn1, packn-pos, find-first-non-cl-symbol, and packn have similarly been weakened to require only atom-listp instead of good-atom-

listp, and the definition of good-atom-listp has been removed.

- The guard for (prin1$ x channel state) now requires (symbolp channel); it was a bug that this conjunct was formerly omitted.
- Several functions that take state had guards requiring that built-in state globals — those bound in initial-global-table — are bound in the global-table of the state (using f-boundp-global or boundp-global). However, the predicate state-p1, which recognizes ACL2 states, implies that these conditions all hold; so, they have been removed from those guards. Such a test has also been removed from the definition of main-timer.

The utility, break-on-error, now causes a much cleaner error when invoked in raw-mode. The installation of breaks by break-on-error works even in raw-mode — it's just that evaluation of a call of break-on-error must not take place in raw-mode. Also, the effect of break-on-error, to cause breaks on errors, works better after entering raw-mode than it did previously, when evaluation of a form (er hard ....) would fail to enter the debugger: that has been fixed. Evaluation of (er soft ....) continues to invoke such breaks whether or not in raw-mode. A new aspect of break-on-error is that breaks occur not only for hard and soft errors, but also when aborting to the top level (which is what happens when (er hard ...) is called in raw-mode). Thanks to Eric McCarthy for reporting issues that led to these improvements.

<<<<<<<<<<<<<<<<<<<<<< **(12)** <<<<<<<<<<<<<<<<<<<<<<

Here's another example showing that ACL2 continues to mature.

When ld is invoked with a non-nil value of keyword :ld-missing-input-ok and the input file is missing, the value returned is now :missing-input instead of :eof.

>>

When defbadge is applied to a function symbol that is built into ACL2 with a badge, such as nth, the result is a no-op and an observation is printed to that effect. (Formerly the badge-table was needlessly extended and ACL2 reported that the function symbol was being given a badge.)

<<<<<<<<<<<<<<<<<<<<<< **(13)** <<<<<<<<<<<<<<<<<<<<<<

For verify-guards, the :hints will default to the :guard-hints from the existing definition. Etc. More system maturity!

Suppose that a verify-guards event for a defined function symbol, fn, does not include a :hints keyword, but the existing definition of fn includes a :guard-hints keyword in its xargs declaration. Then, unlike previously, the value of that :guard-hints keyword is now used as the value of :hints for that verify-guards event. The keywords :guard-debug and :guard-simplify of verify-guards similarly now default to values of the corresponding xargs in the old definition, if supplied there. This change brings the behavior of verify-guards in line with that of verify-termination. See verify-guards for a discussion of the case of mutual-recursion, where only the definition of fn is relevant for keyword values, not other definitions in its clique.

>>>

A hint of the form :induct X is now an error if X is an atom other than t or a quoted constant, since those do not suggest any induction scheme. Thanks to Eric Smith for raising the question of what :induct nil should do and helping with examples showing that the current behavior is not really consistent. Thanks to Alessandro Coglio for suggesting that :induct nil cause an error since it serves no purpose and could be confused with :do-not-induct t, which is quite different.

The built-in defaxiom events code-char-char-code-is-identity and char-code-code-char-is-identity no longer have forced hypotheses. Thanks to Alessandro Coglio for pointing out that these rewrite rules were rather unique as built-in rewrite rules that force hypotheses. Note that the original versions of these two rules, stated as defthm events with suffix "-FORCED" added to the names, may be found in community book books/std/basic/code-char-char-code-with-force.lisp; including this books (perhaps locally) may rescue a proof that now fails because of the change.

Fixed an induction message when limiting the number of cases, in particular, replacing “we had to fold ... into a single IF-expression” by “we had to termify ... (see :DOC termify)” and explaining in the new documentation topic, termify.

<<<<<<<<<<<<<<<<<<<<<<<<< (14) <<<<<<<<<<<<<<<<<<<<<<<

Finally there is support for more than one elim rule for the same function symbol (as is the case for rewrite rules).

It is now legal to introduce more than one elim rule for the same function symbol. Thanks to Eric Smith for pointing out that the current implementation was inconsistent in accepting a replacement elim rule when including a book but not at the top level.

>>

<<<<<<<<<<<<<<<<<<<<<<< (15) <<<<<<<<<<<<<<<<<<<<<<

You can read a stobj inside break-rewrite (and other wormholes). Yet more MATURITY!

It is no longer illegal to read a stobj when in a wormhole state, for example, when inside break-rewrite.

>>

New Features

<<<<<<<<<<<<<<<<<<<<<<< (16) <<<<<<<<<<<<<<<<<<<<<<

Linear arithmetic can be strengthened.

The new zero-ary attachable system function, `heavy-linear-p`, allows for enhanced use of linear-arithmetic during rewriting, specifically with the test (first) argument of a call of `IF`. Thanks to Eric Smith for suggesting the development of such a feature, which can be useful in rewriting-based tools.

>>

There is a new '`make`' target to build an ACL2 executable using `save-exec`. See save-exec, in particular the new discussion at the end of that topic. Thanks to Eric Smith for requesting such a utility.

New utilities `get-cpu-time` and `get-real-time` return the cpu time and real (wall clock) time that has elapsed since the start of the ACL2 session. Thanks to Eric McCarthy for suggesting the addition of such utilities.

<<<<<<<<<<<<<<<<<<<<<<< (17) <<<<<<<<<<<<<<<<<<<<<<

Some hard errors can be suppressed (as with soft error and warnings):

```
ACL2 !>(er-hard 'my-ctx "My Error" "Oops, got an error: ~x0" 'yes)


HARD ACL2 ERROR [My Error] in MY-CTX:  Oops, got an error: YES

```

```
ACL2 Error [Evaluation] in TOP-LEVEL:  Evaluation aborted.  To debug
see :DOC print-gv, see :DOC trace, and see :DOC wet.

ACL2 !>(set-inhibit-er "My Error")
 ("My Error")
ACL2 !>(er-hard 'my-ctx "My Error" "Oops, got an error: ~x0" 'yes)


ACL2 Error [Evaluation] in TOP-LEVEL:  Evaluation aborted.  To debug
see :DOC print-gv, see :DOC trace, and see :DOC wet.

ACL2 !>(set-inhibit-er "My Error" "Evaluation")
 ("My Error" "Evaluation")
ACL2 !>(er-hard 'my-ctx "My Error" "Oops, got an error: ~x0" 'yes)
ACL2 !>
```

The new utility er-hard is analogous to er-soft, but for hard errors instead of soft errors (see er).

>>>

<<<<<<<<<<<<<<<<<<<<<<<<< (18) <<<<<<<<<<<<<<<<<<<<<<<

:TC is like :TRANS but also "cleans": useful for loop$ and mbe (in particular).

```
ACL2 !>:trans (mbe :logic (atom x) :exec (endp x))

(RETURN-LAST 'MBE1-RAW
             (ENDP X)
             (ATOM X))

=> *

ACL2 !>:tc (mbe :logic (atom x) :exec (endp x))
 (ATOM X)
ACL2 !>
```

A new command, :tc (translate and clean), has been added. It translates a given form and then "cleans it up", returning a logically equivalent but often simpler term in which logically irrelevant but operationally important tags have been removed. The variants :tca and :tcp use different degrees of "cleaning." These are particularly useful for seeing the logical meanings of loop$ terms as well as terms involving mbe and return-last.

>>>

<<<<<<<<<<<<<<<<<<<<<<<< (19) <<<<<<<<<<<<<<<<<<<<<<<

Book directories can be relocated using the project-dir-alist.

A directory of books may now be relocated so that those books are still treated as certified. This is supported by a new `project-dir-alist`, which associates keywords with “project directories” and is set using environment variable `ACL2_PROJECTS`; see project-dir-alist. By default, the `project-dir-alist` has only one entry, which associates the keyword `:SYSTEM` with the community-books directory, `books/`. The `project-dir-alist` generalizes the notion of system books directory, assigning meaning to the `:dir` argument of `include-book` and `ld` and to sysfile arguments of `add-include-book-dir` and `add-include-book-dir!`. Thanks to Sol Swords for requesting such a capability and for helpful design discussions. (Technical Note: Implementation-level changes are summarized in comments in the form `(defxdoc note-8-6 ...)` in community-book `system/doc/acl2-doc.lisp`.)

>>>

<<<<<<<<<<<<<<<<<<<<<<<< (20) <<<<<<<<<<<<<<<<<<<<<<<

Inside :brr you can now issue the command, :pot-list.

A new break-rewrite command, `:pot-list`, shows the list of the polynomials that are assumed in the current context. See brr-commands and see brr@. Thanks to Alessandro Coglio and Eric Smith for conversations leading to this enhancement.

>>>

It is now possible to cause `make-event` to do proofs during its expansion phase even in a context where proofs are generally skipped (see ld-skip-proofsp). See make-event, in particular the discussion there labeled as “(4)”. Thanks to Eric Smith for requesting such a feature.

<<<<<<<<<<<<<<<<<<<<<<<< (21) <<<<<<<<<<<<<<<<<<<<<<<

Another loop$ improvement: automation of induction based on DO loop$ expressions.

The induction mechanism in the prover can now deduce induction suggestions from some `DO loop$`s. See stating-and-proving-lemmas-about-loop$s for a brief discussion.

>>>

A new :linear rule, acl2-count-car-cdr-linear, is now built into ACL2, as follows. Thanks to Eric Smith for suggesting this improvement (slightly renamed here) to what we originally added.

Theorem: acl2-count-car-cdr-linear

```
(defthm acl2-count-car-cdr-linear
  (implies (consp x)
           (equal (acl2-count x)
                  (+ 1 (acl2-count (car x))
                     (acl2-count (cdr x)))))
  :rule-classes :linear)
```

<<<<<<<<<<<<<<<<<<<<<<<<< (22) <<<<<<<<<<<<<<<<<<<<<<<<<

Macrolet supports local macro definitions, like flet supports local function definitions.

The Common Lisp utility, macrolet, is now supported in ACL2. Thanks to Alessandro Coglio for discussion leading us to make this addition. See macrolet.

>>>

Lambda objects in positions of ilk :FN are now subjected to a size limitation. See explain-giant-lambda-object.

The utilities with-output and with-output! have been enhanced in the following two ways; see with-output for details.

- A new keyword, :inhibit-er-hard, can be supplied a non-nil value to turn off hard errors when error output is inhibited. Thanks to Eric Smith for a conversation leading to this enhancement.
- The keyword :off for the utility with-output (also with-output!) can take on a new value, :all!, which is treated exactly the same as using arguments :off :all :gag-mode nil :inhibit-er-hard t.

The new utility with-cbd creates a scope for an indicated value of the connected book directory (see cbd). Calls of with-cbd are allowed in books as well as in encapsulate and progn events; see embedded-event-form. Thanks to Sol Swords for requesting that with-cbd be legal in embedded events. Thanks to Sol Swords and Mertcan Temel for reporting a bug in an initial implementation.

The new utility, with-current-package, evaluates a given form with respect to an indicated current-package. Thanks to Sol Swords for requesting this utility.

<<<<<<<<<<<<<<<<<<<<< **(23)** <<<<<<<<<<<<<<<<<<<<<

Fast unsound book certification!

See fast-cert for a "fast-cert" mode for faster, but possibly unsound, book certification, in particular when using a saved executable that contains local events. Thanks to Sol Swords for requesting such a capability and for helpful design discussions.

>>

<<<<<<<<<<<<<<<<<<<<< **(24)** <<<<<<<<<<<<<<<<<<<<<

You can turn warnings into (hard) errors (see the :DOC for an example).

Added utility set-warnings-as-errors, which can change warnings to hard errors. Thanks to Mark Greenstreet for the idea and for discussions that were helpful in refining it.

>>

A new ld special, ld-always-skip-top-level-locals, has the effect of skipping local top-level forms. Thanks to Sol Swords for requesting such a capability, to support faster loading of .port files by the build system (see build::cert.pl).

The symbol, number, is now a legal type-spec.

<<<<<<<<<<<<<<<<<<<<< **(25)** <<<<<<<<<<<<<<<<<<<<<

Something like (foo st st) is legal for stobj st, when foo is defined with read-only stobj arguments st1 and st2 that are congruent to st.

It is now permitted for a stobj s to occur more than once as an actual parameter in a function call, provided each such occurrence is in a position where a stobj congruent to s is expected (possibly s itself). Thanks to Sol Swords for providing a relevant example, which appears in a comment in the definition of function stobjs-in-out in the ACL2 sources.

>>

<<<<<<<<<<<<<<<<<<<<< **(26)** <<<<<<<<<<<<<<<<<<<<<

With-brr-data shows the source of a mystery term in a proof; there's a talk on this later today. Is 15 years a long time?

A new utility, with-brr-data, together with related query utilities, allows one to find the source of a term in prover output. Thanks to Sol Swords for requesting something like this in 2008 (!) and to Eric Smith for requesting a related utility (see community-books `kestrel/utilities/brr-data-failures.lisp` and `kestrel/utilities/brr-data-all.lisp`). See with-brr-data.

>>

`Make-event` takes a new keyword, `save-event-data`. When that keyword is non-`nil`, event-data from the expansion saved is preserved; see the new section at the end of make-event-details. This new feature was motivated by the desire to preserve a proof's event-data in calls of `thm`, as described in the "Changes" section above.

New utilities allow one to explore what has changed when an event fails in a book that formerly certified. See saving-event-data. Thanks to Eric Smith for requesting such a capability and for helpful bug reports for early versions of these utilities.

<<<<<<<<<<<<<<<<<<<<<<<< (27) <<<<<<<<<<<<<<<<<<<<<<<

Issue the command (set-verify-guards-eagerness 3) to overcome certain verify-termination failures.

A new legal value for `set-verify-guards-eagerness`, `3`, causes guard verification even when `:verify-guards nil` has been declared. This can be helpful when a `verify-termination` event in a book is intended to verify guards but a locally included book declares `:verify-guards nil` in the corresponding `verify-termination` event. (Technical note: That issue occurs because `verify-termimnation` invokes `make-event`, and the expansion is saved when locally including the sub-book.)

>>

When environment variable `ACL2_WRITE_BOOKDATA` has a non-empty value, then files of the form `*__bookdata.out` will be written; see bookdata for explanation and an exception. Thanks to Eric Smith for requesting this enhancement.

<<<<<<<<<<<<<<<<<<<<<<<< (28) <<<<<<<<<<<<<<<<<<<<<<<

Finally, thm can take :instructions.

Thm now takes an optional `:instructions` keyword argument, like `defthm`. Thanks to Warren Hunt for requesting this enhancement.

>>

<<<<<<<<<<<<<<<<<<<<<<<< (29) <<<<<<<<<<<<<<<<<<<<<<

:Trans is like ;trans but shows the sequence of expansions, including make-event expansions. **Demo** will use the following input from :DOC std::defaggregate.*

```
(include-book "std/util/defaggregate" :dir :system)

; Not much information:
:trans (std::defaggregate employee     ;; structure name
              (name salary position)   ;; fields
              :tag :employee           ;; options
              )

; Much more information:
:trans* t (std::defaggregate employee     ;; structure name
                 (name salary position)   ;; fields
                 :tag :employee           ;; options
                 )
```

New utilities `trans*` and `trans*-` can show repeated macroexpansions of a form leading to its final translation, where `trans*` also shows `make-event` expansions. Thanks to Warren Hunt for requesting a utility that does repeated macroexpansion.

>>

<<<<<<<<<<<<<<<<<<<<<<<< (30) <<<<<<<<<<<<<<<<<<<<<<

A new proof-builder command shows linear database: POT-LST.

A new command has been added to the proof-builder-commands, to display the linear arithmetic database. See ACL2-pc::pot-lst. Thanks to Dave Greve for bringing forward the idea of such a command.

>>

<<<<<<<<<<<<<<<<<<<<<<<< (31) <<<<<<<<<<<<<<<<<<<<<<

Get even more “raw” proof output with (set-raw-proof-format :clause). I’ll demo this if

time with (thm (equal (append (append x y) z) (append x y z))). (But who cares, since normally gag-mode suppresses printing of prover goals.)

The utility `set-raw-proof-format` now takes a new legal value, `:clause`, which is like `t` except that in addition to printing runes as lists in output from the simplifier, it prints all goals as clauses. Thanks to Eric Smith for encouraging such an enhancement.

>>

<<<<<<<<<<<<<<<<<<<<<<<< (32) <<<<<<<<<<<<<<<<<<<<<<<

Type-set reasoning can be strengthened.

It is possible to cause ACL2 to increase its effort in type-reasoning. See set-dwp. Thanks to Eric Smith for correspondence leading to this feature.

>>

Heuristic and Efficiency Improvements

<<<<<<<<<<<<<<<<<<<<<<<< (33) <<<<<<<<<<<<<<<<<<<<<<<

The next item is a small tweak to the prover. It highlights two opposite things: ACL2's reasoning engine is actually quite stable, but it continues to evolve in small ways.

Added a "desperation heuristic" to compute a stronger context, for the extra try at simplification made after a goal is not changed by simplification. Thanks to Warren Hunt and Vivek Ramanathan for supplying an example of a theorem whose proof had a surprising failure but now succeeds. (Technical Remark describing this change: When a clause has most recently settled down at the time that the simplify process is invoked (a so-called "desperation heuristics" attempt), then the literals are reordered before building the type-alist, so that the literals that involve at most one variable precede the other literals.)

>>

Generation of guard clauses (and, probably rarely, other goals) has been sped up in certain extreme cases. For details, see system-attachments, specifically the discussion of CONJOIN-CLAUSE-SETS-BOUND in the "Summary of attachable system functions". Thanks to Alessandro Coglio for sending an example that led to our discovery of the quadratic behavior eliminated by this change.

Duplicate entries in type-alists (proof contexts) are now avoided in many cases. (Implementation note: some calls extending the type-alist with an existing term/type-set pair are now avoided in source function `assume-true-false-rec`.) Thanks to Eric Smith for pointing out that there can be type-alists with many consecutive identical entries.

Sped up macroexpansion for several common macros, with roughly a 2% to 3% speedup observed for including several large books during development of this change.

Tweaked linear-arithmetic to avoid consideration of an equality between two terms that are both known (via their type-sets) to be non-numeric.

The utility `set-cbd` is more efficient when setting to the current `cbd`, including the common case of calls to `set-cbd` by `ld` and (hence) `wormhole`.

New built-in rewrite rules `all-boundp-initial-global-table-alt` and `all-boundp-initial-global-table` may help with reasoning about `state-p1`. Use `:pe` to see their events. The latter is disabled by default and may useful to enable when developing proofs that rely on built-in state globals being bound.

ACL2 has a procedure for evaluating ground terms (terms without free variables) that is used in the generation of guard obligations as well as in linear-arithmetic and forward-chaining. This procedure was not used on subterms of bodies of lambda expressions, but now it is. Thanks to Eric Smith for requesting this enhancement (in particular for generation of guard obligations).

Bug Fixes

Fixed a soundness bug based on rules of class `:meta` or `:clause-processor` that would be exported from an `encapsulate` event with a non-`nil` signature list. A proof of `nil` in Version_8.5 may be found in the community-books file, `books/system/tests/transparent-functions-input.lsp`; search for this paragraph there.

It was probably a soundness bug to allow a `defaxiom` event to designate a rule of class `:meta` or `:clause-processor` in its `:rule-classes`. That is no longer allowed; `skip-proofs` may be used instead if one believes that the proposed formula is a theorem.

The function `read-file-into-string` has been modified to avoid what might be considered a soundness bug. The change involves causing an error for two reads of the

same file without first incrementing the file-clock of the state. See read-file-into-string for details, in particular for how to avoid that error by evaluating `(increment-file-clock state)` after calling `read-file-into-string`. Formerly the error was avoided if the write-date of the file didn't change between the two reads, but the following example shows how this permitted two calls with identical arguments to produce different results, logically causing `read-file-into-string` to violate the axiom `x = x`.

First run the following shell commands.

```
echo 'test1' > tmp1.txt ; echo 'test2' > tmp2.txt
cp -p tmp1.txt tmp.txt
```

Then start ACL2 and run a command as follows.

```
ACL2 !>(read-file-into-string "tmp.txt")
"test1
"
ACL2 !>
```

Now suspend ACL2 with `control-Z` and run the following shell command.

```
cp -p tmp2.txt tmp.txt
```

Now resume ACL2 with `fg`, and optionally submit some trivial form (say, `3`) just to get the prompt back. Note that the file-clock of the `state` hasn't changed. (Probably the `state` hasn't changed; at any rate, the parts of the state relevant to `read-file-into-string` haven't changed.) So the following call has arguments identical to those in the corresponding call above, yet yields a different result.

```
ACL2 !>(read-file-into-string "tmp.txt")
"test2
"
ACL2 !>
```

After the change to `read-file-into-string`, its call just above causes an error.

Fixed a bug in system function `bounded-integer-listp`, which may have allowed illegal proof-builder commands to be attempted. Thanks to Grant Jurgensen for pointing

out this bug.

Consider calls of defthm and thm that create subgoals before reverting to prove the original goal by induction. The Rules summary printed at the end should exclude rules used only before the start of that induction proof. That was formerly the case for defthm but not thm, but now it is the case for both. You can see the change for the following call, whose Rules summmary formerly included (:ELIM CAR-CDR-ELIM) but no longer does so.

```
(thm
 (equal (append (append x y) z)
        (append x y z))
 :hints (("Goal"
          :expand ((:free (b) (append x b))
                   (:free (a b) (append (cons (car x) a) b))))))
```

Fixed bugs in the definition of source macro position-ac. Thanks to Eric Smith for pointing them out.

Fixed translation of DO loop$ expressions, so that the next-to-last argument of the resulting do$ call quotes the untranslated measure instead of the translated measure.

Several improvements were made to the FOR loop$ utility (also see for-loop$, to reflect more accurately the Common Lisp loop utility. This matters because in guard-verified code, loop$ becomes loop. Here are the most user-visible such changes.

- Run-time guard-checking for loop$ operators SUM and APPEND did not include a check that the value produced at each iteration is a number or true list, respectively. That has been fixed so that, for example, the expression (loop$ for v in '(1 a 2) sum v) now causes a guard violation (because a is not a number), where previously it did not.
- For a form (loop$ for tail on lst ...), the target term, lst, no longer needs to satisfy true-listp. For example, the form (loop$ for tail on '(a b . c) collect tail) no longer causes a guard violation.
- Run-time guard-checking for an expression (loop$ for tail on lst ...) now includes a check for the target, lst, that its final tail (i.e., , (last-cdr lst) satisfies the declared type of the corresponding iteration variable. For example, evaluation of the loop$ expression below now produces a guard violation as shown, but it formerly did not produce a guard violation.

```
ACL2 !>(loop$ for tail of-type cons on '(a b c) collect tail)


ACL2 Error [Evaluation] in TOP-LEVEL:  The guard condition
(CONSP LOOP$-LAST-CDR), which was generated from a type declaration,
has failed.
See :DOC set-guard-checking for information about suppressing this
check with (set-guard-checking :none), as recommended for new users.
To debug see :DOC print-gv, see :DOC trace, and see :DOC wet.

ACL2 !>
```

Corresponding run-time checking was added for the types of the lower and upper bounds `lo` and `hi`, the increment `inc`, and the last value tested, in expressions `(loop$ for i from lo to hi by inc ...)`.

Fixed a low-level bug in source function `translate-declaration-to-guard1-gen` that was incorrectly creating untranslated terms in some cases from type declarations of the form `(type (signed-byte _) _)` or `(type (unsigned-byte _) _)`. Here is an example of an event that is rejected without the bug fix.

```
(defun foo (n)
  (apply$ (lambda$ (x)
                   (declare (type (signed-byte 8) x)
                            (xargs :guard (signed-byte-p 8 x) :split-types t))
                   (+ 3 x))
          (nfix n)))
```

Suppose a book is certified in a world where a portcullis command generates a `local` call of `make-event`. Then that event is now ignored when subsequently including that book. Previously it may not have been ignored, because the `local` wrapper could be ignored when writing the book's certificate.

Some handling of exceptional cases in hints has been cleaned up, as follows. Thanks to Eric Smith for a discussion that led to these changes.

- Warnings for repeating a goal name in the hints now appear even when the repetition is only up to case. For example, such a warning is generated now for `:hints (("Goal" :use foo) ("GOAL" :use bar))` where formerly it was not. The discussion of this situation in documentation topic hints has been improved.
- It was incorrectly documentated in topic hints, in the discussion of `:do-not` hints, that it is illegal to associate a goal name with the empty list of hints, as in

("Goal"). This behavior was actually allowed; an empty such hint was simply ignored. This continues to be allowed (for backward compatibility) but the documentation has been updated; also, these empty hints are now ignored for purposes of the warnings mentioned above (formerly they were considered when looking for repetition of goal names).

A bug in the brr commands :eval$, :go$, and :ok$ was fixed so they now behave as described in the documentation for brr-commands.

When a certified book is included, the logical world will no longer be marked as having seen a skip-proofs call, even when the value of ld special ld-skip-proofsp is non-nil at that time. Thus, that situation no longer disqualifies such a world from supplying the portcullis commands to a book to be certified without keyword argument :skip-proofs-okp t of certify-book. Thanks to Sol Swords for pointing out this bug.

Fixed a bug that was causing calls of wormhole to signal an error.

Fixed a bug that could cause a do-loop$ expressions to be inappropriately rejected due to an allegedly ignored variable. An example is below.

```
(include-book "projects/apply/top" :dir :system)
; BUG: The following was formerly necessary, but no longer is.
(set-ignore-ok t)
(defun f (a b)
  (loop$ with x = a with y = b
         do
; The use of (set-ignore-ok t) was needed, but shouldn't have been,
; whether or not the next line is included.
         :measure (+ (len x) (len y))
         (cond ((consp y)
                (let ((z y))
                  (progn (setq y (cdr x))
                         (setq x (cdr z)))))
               (t (return y)))))
```

Fixed the failed redundancy check when setting a table guard that returns two values. For example, the form (table foo nil nil :guard (mv t nil)) was not formerly seen as redundant when evaluating it a second time.

Fixed a bug that was causing cw-gstack to report “Rewriting (to simplify) the first argument” when rewriting the second argument of a call of implies, and fixed an analogous bug for return-last.

Fixed a bug in intersection$ that prevented it from being called with keyword argument :test 'equal. Thanks to Anna Slobodova for bringing this bug to our attention.

Certain error messages from translate and untranslate are now inhibited when they should be (where formerly they weren't). Thanks to Eric Smith for bringing this problem to our attention.

Fixed a bug that in rare cases, for direct prover calls (e.g., with prove$, could cause an error reporting "HARD ACL2 ERROR in pop-warning-frame". Thanks to Eric Smith for bringing this bug to our attention. Note that with this change, then when an event's evaluation causes a hard error (see er and hard-error): summary information may be printed that was formerly omitted; and a superfluous extra failure message may be omitted that was formerly printed.

The function delete-file$ executed in a way that diverged from its logical definition: successful deletion caused return values of (mv t state) but this was provably impossible according to the logical definition. This has been fixed.

Fixed the useless-runes feature to work properly when reading a useless-runes file while certifying a book in a package other than the "ACL2" package. The fix is to ensure that the useless-runes file is read while in the "ACL2" package, which is the same package used when the file was written.

Fixed a bug in handling of the ld keyword :ld-missing-input-ok, by eliminating an error in the case that the specified input file's directory does not exist. The fix avoids executing some of the ld code that was formerly executed. Thanks to Alessandro Coglio and Eric Smith for bringing this bug to our attention.

An error formerly occurred if one first evaluated (defwarrant FN) for some FN and then attempted to include a certified book containing (defbadge FN). That has been fixed. Thanks to Mertcan Temel for reporting this bug. Such a defbadge event is now a no-op, which is reported by an observation except during include-book and except during the second pass of an encapsulate event. Moreover, the defwarrant event now returns the value :WARRANTED instead of T — more precisely, it returns a value-triple whose value component is :WARRANTED — and similarly for defbadge and :BADGED.

We fixed a bug that caused the tau-system sometimes to cause a raw Lisp error when the executable-counterparts of certain primitive recognizers were disabled. Thanks to Eric Smith for reporting this bug and providing an example of it.

We fixed a bug that might cause suitable induction schemes to be eliminated because of inappropriate consideration of :definition rules.

Some proof-builder commands, including :claim and :casesplit, are more robust (specifically, when claiming or splitting on certain terms that are trivially true). Thanks to Drew Walter for supplying an illustrative example.

A bug in trace$ could trigger raw Lisp errors when the :entry, :exit, or :cond option specified an ill-guarded expression. For example, after evaluating (defun f (x) x) and then (trace$ (f :entry (car x))), evaluation of (f 3) caused a raw Lisp error due to evaluation of car on the value, 3. This has been fixed. (Technical Remark: the fix was to replace the expression with the same "oneify" process that is used for generating definitions of executable-counterpart functions; evaluation.)

Eliminated the built-in :type-prescription rule true-listp-take, thanks to Eric Smith, who pointed out that the same rule is already installed as the built-in :type-prescription rule for take.

Changes at the System Level

The 'make' target, save-exec, now builds custom-saved_acl2 unconditionally. Thanks to Grant Jurgensen for pointing out (in GitHub Issue #1422) that there can be untracked implicit dependencies that make this necessary.

Implementations underlying the functions sys-call, sys-call+, and sys-call* have been cleaned up. In particular, we now expect them to indicate precisely the case of a non-error process return as follows: 0 for sys-call-status, and nil for the first return value of sys-call+ and sys-call*. Also, a bug has been fixed for sys-call+ in the case of GCL as the host Lisp. Thanks to Eric Smith for queries leading to these changes.

<<<<<<<<<<<<<<<<<<<<<<< (34) <<<<<<<<<<<<<<<<<<<<<<<

:DOC system-attachments shows all built-in system attachments, with some brief documentation.

Significantly extended documentation topic system-attachments, which now lists all built-in system attachments, many with brief documentation. Thanks to Eric Smith for suggesting this enhancement.

>>>

<<<<<<<<<<<<<<<<<<<<<<<<< (35) <<<<<<<<<<<<<<<<<<<<<<<<

There has been a lot of recent, major documentation work; I'll go through the following quickly.

Significant new documentation topics, together with subtopics and books supporting those topics, include the following. [See DARPA Note above.]

- Start-here provides a guide for those getting started with ACL2.
- Recursion-and-induction has been extensively modified from past, standalone versions by J Moore, and is now integrated with the rest of the documentation. Thanks to Vivek Ramanathan for helpful feedback.
- Gentle-introduction-to-acl2-programming has also been updated from past standalone versions and integrated into the rest of the documentation.
- Loop$-primer provides an extensive primer on the the ACL2 `loop$` feature.
- Type-reasoning gives a basic introduction to what has sometimes been called "type-set reasoning" but is now generally referred to as "type reasoning". This new topic is now referenced in many other built-in documentation topics. Thanks to Warren Hunt for communication leading to this new topic.
- Soundness discusses what we can conclude when we use ACL2 to prove a formula or to compute the value of an expression.

>>>

Allow `ld` output in raw-mode to go to other than the channel, `*standard-co*`. Thanks to Vivek Ramanathan and Warren Hunt for an example illustrating the issue.

<<<<<<<<<<<<<<<<<<<<<<<<< (36) <<<<<<<<<<<<<<<<<<<<<<<<

The next one is so helpful for me and J that it seemed worth a mention for others who experiment with redefining system code.

(For system hackers only) The feature `:acl2-loop-only` is now true inside the ACL2 read-eval-print loop. Therefore, the function `lp!` is no longer supported or necessary, since `(lp)` enters the loop with feature `:acl2-loop-only` true, just as `(lp!)` did previously.

>>

(CCL only) Stobj array code now has a workaround for a CCL bug found by Yahya Sohail, in the case of reading a stobj array of integers where the element type includes at least one negative number and one non-fixnum. That bug has been around since at least as far back as 2017, and was fixed on June 12, 2023. For those using a CCL version with the bug, this fix may slow down such stobj array reads a bit in the case described above; one measurement showed about 37% more time for such a read. Thanks to Yahya for the bug report, to Warren Hunt for encouraging a workaround, and to the CCL developers (in particular Gary Palter) for fixing the CCL bug.

The undocumented utility, `thm-fn` — which is used in several community-books — now has an additional formal (at the end), `event-form`. That argument can generally be passed as `nil` for appropriate behavior.

Code for `set-cbd` has been tweaked to add assurance that the `cbd` always ends in a forward slash (‘`/`’), as specified. Thanks to Stephen Westfold for a comment leading to this modification.

Updated file `GNUmakefile` in the top-level directory so that when an ACL2 executable is built, files are updated in subdirectory `books/build/` to support the use of `build::cert.pl`. Thanks to Eric Smith for the idea, and thanks to Eric and also Sol Swords for help with the implementation.

The notion of ACL2 state is formalized in function `state-p1`, which has implicitly changed because it depends on the constant `*initial-global-table*`, whose value has changed. That constant’s value, which is still an alist, now includes additional pairs, which are from the constant `*initial-ld-special-bindings*`; thus, `*initial-global-table*` now specifies a value for each so-called “`ld` special”.

The previous ACL2 release (Version_8.5) arranged that when a raw Lisp error is encountered, any available input is cleared from the input channel. However, this can lead to discarding of valid input or an attempt to read values from within a comment; see community-books files `clear-input-1.lsp` and `clear-input-2.lsp`, respectively, in directory `books/system/tests/`. So now, input is cleared on error only when reading from the terminal (technically, from `*standard-oi*`).

EMACS Support

A set of tools for assisting in the conversion of certain HTML to xdoc may be found,

without much documentation, in `emacs/html-to-xdoc.el`. [See DARPA Note above.]

When new ACL2-doc buffers are created by using the `G` or `Shift-<Return>` commands, their name reflects the topic name, e.g., `acl2-doc<REWRITE>` if the topic visited in the new buffer is `REWRITE`. Note that the new buffer name stays the same even if other topics are visited there; its name reflects its topic at the time it was created. Thanks to Warren Hunt for requesting such an enhancement. Note that the former behavior can be restored by evaluating the form `(setq *acl2-doc-short-new-buffer-names* t)` in Emacs.

The initialization file for recent Emacs versions, `books/emacs/emacs-acl2.el`, now correctly loads related files — notable `acl2-doc.el` — from that same directory, rather than from the `emacs/` directory that is directly under the top level of the ACL2 distribution.

<<<<<<<<<<<<<<<<<<<<<<<< (37) <<<<<<<<<<<<<<<<<<<<<<

Documentation at the terminal and in the acl2-doc browser can show fonts and (in acl2-doc) images. Examples: see topics xdoc::markup and x86isa using acl2-doc.

Documentation printed to the terminal or in the ACL2-Doc Emacs browser can respect certain xdoc::markup that was formerly ignored, but is no longer (by default), as follows.

- Fonts are largely respected. For example, text marked as underline is now underlined, and text marked as having typewriter font now has a grey background. This closes GitHub Issue 1487. Thanks to Grant Jurgensen for helpful feedback on the original plan, which had been to use delimiting underscores rather than Select Graphic Rendition (SGR) control sequences. See xdoc::terminal for details and for ways to customize behavior, including avoidance of SGR.
- Images now appear in ACL2-Doc (but not with `:doc` at the terminal), instead of `{IMAGE}`, on systems that can display graphics inside Emacs. Thanks to Warren Hunt for requesting this enhancement.

>>

The documentation for ACL2-Doc says of the search commands `s` and `S`, “go to that topic with the cursor put immediately after the found text”. But the cursor was at the end of the found text, not immediately after it. That has been fixed.

For the ACL2-doc browser, the download (`D`) command now accesses, by default, an

`https` address instead of an `http` address. Thanks to Warren Hunt for suggesting this change.

Experimental Versions

The note “Note: No checkpoints to print.” that might be printed on proof failure is now the same in ACL2(p) as in ACL2, unless waterfall-parallelism is enabled (in which case “no checkpoints” is followed by “ from gag-mode” as before).

Subtopics

Note-8-6-books
Release notes for the ACL2 Community Books for ACL2 8.6