Note-8-5

ACL2 Sources

ACL2 Version 8.5 (xx, 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 8.4 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-8-5-books for a summary of changes made to the ACL2 Community Books since ACL2 8.4, 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

Weakened hypotheses from built-in theorems (thus strengthening them) `symbol-equality`, `true-listp-first-n-ac-type-prescription`, `ordered-symbol-alistp-getprops`, `add-pair-preserves-all-boundp`, and `symbol<-asymmetric`. Eliminated built-in theorem `main-timer-type-prescription` entirely (that rule was already deduced by ACL2 at definition time). Simplified guards slightly for built-in functions `serialize-write-fn` and `serialize-read-fn`. Thanks to Eric Smith for correspondence, based on his linter, leading to these improvements.

The utility `without-evisc` formerly always (or nearly always) returned the error-triple (`mv nil :invisible state`) after printing the result. Now it generally returns (`mv t nil state`) when evaluation of the given form causes an error. See without-evisc. Thanks to Karthik Nukala and Eric Smith for reporting the former (undesirable) behavior.

One would get an error when including an uncertified book when a `:type-prescription` specified in an `xargs` declaration failed a validity check, even when no such failure occurs when that book is certified. (That could happen because type-

prescription information from locally included books is saved in the book's certificate file and is used when checking such :type-prescription declarations.) This situation now generates a warning rather than an error. Thanks to Karthik Nukala and Eric Smith for sending an example that pointed out this problem.

Output from :oops may now be inhibited as OBSERVATION output, by using set-inhibit-output-lst or with-output.

Raw-mode has been fixed so that reasonable results are printed when multiple values are returned by a function defined in raw-mode or raw Lisp (rather than in the logic). The following examples illustrate the problem and the fixed behavior. The utilities add-raw-arity and remove-raw-arity are no longer necessary (and they had no effect anyhow, at least in recent ACL2 versions), so they have been removed.

```
(defstobj st fld)
(set-raw-mode-on!)
(defun f1 (st) st)
(f1 st) ; printed a vector; now prints <st>
(defun f2 (state) state)
(f2 state) ; printed ACL2_INVISIBLE::|The Live State Itself|;
           ; now prints <state>
(defun bar (x st state) (mv x st state))
(bar 3 st state) ; printed 3; now prints (3 <st> <state>)
```

The keyword :concrete for defabsstobj and the keyword :witness-dcls for defun-sk are no longer supported, as they were deprecated in ACL2 Version 8.4 in favor of :foundation and declare forms, respectively. Also: the following symbols, deprecated in ACL2 Version 8.4, are no longer names of events.

```
logical-defun
merge-sort-symbol-<
merge-symbol-<
strict-merge-sort-symbol-<
strict-merge-sort-symbol-<-cdrs
strict-merge-symbol-<
strict-symbol-<-sortedp
symbol-<
```

The string "Proof succeeded" or "Proof skipped" was formerly printed when SUMMARY output is not inhibited (see set-inhibit-output-lst), but now that printing takes place instead when PROVE output is not inhibited. Thanks to Pete Manolios for pointing out a printing issue that is resolved with this change.

Some error messages were improved for the proof-builder, primarily when refusing a command to dive into an OR expression. Thanks to Warren Hunt for bringing this issue to our attention.

There was a restriction on certain events, notably defconst and defmacro events, to ensure that they are not ancestrally dependent on loop$ or lambda$ expressions. That restriction has been removed in the case that the body of the event is a quoted constant.

A defwarrant event may complete more quickly because a generated hint now disables the function.

Functions that take or return stobjs may now have badges and warrants. One cannot (yet) take advantage of these in general because one cannot put a stobj into a list, as required for the second argument of apply$. However, this change supports the use of stobjs in loop$ expressions that use the new keyword, DO.

The function the-check, which is generated by calls of the, is now a guard-holder.

Printing of checkpoints now takes place any time SUMMARY output is enabled (see set-inhibit-output-lst. Formerly, both SUMMARY and ERROR output needed to be enabled. Related tweaks, probably not user-visible, were made in support of utilities for obtaining and displaying checkpoints programmatically: see checkpoint-list (which mentions related utilities as well), and we thank to Eric Smith for requesting such utilities.

The guard formula utilities (see guard-formula-utilities) continue to perform simplification as before, except that way to specify the level of simplification has changed. Thanks to Eric Smith for a query and subsequent discussion that led to these changes. See guard-simplification for a detailed explanation; below is a summary. (The reason for these changes is that the value T formerly meant different things in the two cases below — all simplification and limited simplification, respectively — and the value NIL also meant different things in those two cases — limited simplification and no simplification, respectively.)

- For xargs keyword :guard-simplify and related utilities guard-obligation and verify-guards-formula:
  The default value for simplification remains T. However, the value is now :LIMITED for specifying reduced simplification; formerly it was NIL, which is now illegal.
- For :guard-theorem lemma-instances and the related utility gthm (also the low-

level utility, guard-theorem):
The default simplification is unchanged but now corresponds to a new default value, :LIMITED; formerly it was T, which is now illegal. The value NIL continues to be appropriate for avoiding simplification.

Improved untranslate so that when untranslating a translated term, an attempt is made to call mv where appropriate. Thanks to Alessandro Coglio and Eric Smith for requesting this enhancement. The improvement also restores type declarations under let, let*, and mv-let (see declare and type-spec); and it also restores ignore declarations in let and let* forms, where previously that was only the case for mv-let forms. In addition, a new utility, maybe-convert-to-mv, may be called explicitly to convert an untranslated term to one that calls mv in leaves reached via transversal of the true and false branches of its top-level IF tree, where the traversal appropriately passes through let, let*, and mv-let forms, as well as calls of prog2$, mbe, mbt, ec-call, time$, and a few other macros related to return-last as well as return-last itself. See examples under a comment about "preserving executability" in the community-book, books/system/tests/untranslate.lisp. (Note: The changes also include a bug fix that avoids generating an mv-let expression when there are fewer than two bound variables.)

The second argument of certify-book is no longer allowed to be the symbol, T, or any symbols whose symbol-name is "T". (This option seems to have been essentially unused but it complicated the source code.)

Suppose ld is called in the scope of local, in particular, as with (local (ld ...)). Then for each command C read by that call of ld that is not already of the form (local ...), C is read as though it had been (local C). (This change has been made in support of local portcullis events, a new feature described further below.)

New Features

A new loop$ keyword, DO, supports an imperative style of programming in loops. In particular, DO loop$ expressions may use setq and mv-setq for assigning to one or several variables (respectively), they may reference and return stobjs, and they may return multiple values. See loop$.

One can now suppress output from cw, cw!, fmt-to-comment-window, and fmt-to-comment-window!, and from utilities that use these such as time$, by inhibiting a new output type, COMMENT. (Thus, that symbol has been added to the value of *valid-

output-names*.) See set-inhibit-output-lst and with-output. Thanks to Eric McCarthy for a conversation via GitHub Issue #1293 that led to this enhancement. Moreover, new macros cw+ and cw!+ and new functions fmt-to-comment-window+ and fmt-to-comment-window!+ never suppress output; the "+" suffix is intended to indicate that feature. Thus, these new utilities behave like the previous utilities without the "+" suffix. Thanks to Eric Smith, Karthik Nukala, and Alessandro Coglio for observing inappropriate suppression of output from the utility er-soft+ and the connection of this problem to the addition of the COMMENT output type; it was resolved by using fmt-to-comment-window+ in place of fmt-to-comment-window in the implementation of a utility underlying er-soft+ (see community-book books/tools/er-soft-logic.lisp).

You can now arrange that an interrupt will kill a proof immediately by evaluating (assign abort-soft nil), and you can restore the default behavior — where an interrupt instructs the proof to quit cleanly at an appropriate opportunity — by evaluating (assign abort-soft t). Note that this can interfere with :redo-flat; see abort-soft. Thanks to Eric Smith for reporting an inability to abort a series of proof attempts using the prove$ utility, which led to this enhancement.

A stobj may now have a field of type STOBJ-TABLE, which associates arbitrary stobj names with corresponding stobjs. As of this writing, that feature should be considered experimental; see stobj-table. Thanks to Rob Sumners for suggesting the idea and to him and Sol Swords for useful design discussions.

The utility wet has a new keyword option, :fullp, that allows it to work even when there is a raw Lisp error. Thanks to Eric Smith for requesting that wet be able to work in such cases.

Rewriting of lambda objects (see rewrite-lambda-object) may now be disabled, by disabling the executable-counterpart rune for (trivial function), rewrite-lambda-modep — for example, with a hint :in-theory (disable (:e rewrite-lambda-modep)). Also improved a couple of warnings and a bit of documentation pertaining to such rewriting.

The undocumented last-ld-result feature has been replaced by a new documented feature, a ld-history that records command input/output history. Thanks to Eric Smith for requesting this feature.

The ld utility accepts a new keyword argument, :useless-runes, which functions much the same as does the :useless-runes keyword argument of certify-book. See

useless-runes. Thanks to Eric Smith for requesting this feature (which may have been requested previously as well).

A new event, set-inhibit-er-soft, allows the user to turn off error output of various types. The related utility toggle-inhibit-er-soft can turn on or off a single type of error output. Non-local versions of these utilities are set-inhibit-er-soft! and toggle-inhibit-er-soft!. Many error messages cannot yet be controlled this way, but this may be remedied somewhat with community feedback. Thanks to Eric Smith for a request to inhibit step-limit error output, which led to this enhancement.

The functions l<, lexp, and d<, originally defined in community-book books/ordinals/lexicographic-book.lisp, are now built into ACL2. However, it is still useful to include an ordinals book, for example community-book books/ordinals/ordinals, if you want to reason about such functions.

A stobj field of HASH-TABLE type may now specify an element type; for example, (hash-table eql nil integer) specifies a test of eql, no size restriction, and the element type, integer, indicating that only integers are stored in the hash table. The :initially keyword of such a field is no longer ignored, but instead provides the value returned when looking up a key that is not bound in the hash table. Moreover, the element type may be a stobj, indicating that the values in the hash table are all instances of that child stobj; see nested-stobjs. In that case, the :initially keyword would be meaningless because one always gets a fresh instance of the child stobj when looking up an unbound key; hence :initially is illegal for any hash-table field with a stobj element type. Thanks to Rob Sumners for requesting this enhancement and for helpful discussions about its design.

It is now possible to certify a book in a logical world that contains local events. Thus, a book's certificate file may include local events among its portcullis commands, which will be skipped when including the book. A key application of this new capability is to save an ACL2 executable (using save-exec) after locally including set B of books at the top level, and then — in fresh ACL2 sessions started by running that executable — certify other books that may include some books of B. Those inclusions would then be redundant during certification of a book, but not later when the book is included. If B is a large set of large books, then this process can be much more efficient than doing certifications from a basic ACL2 executable, which would involve running include-book on all required books from B. Of course, one could previously have saved such an executable and certified books on top of it, but then all of B would be included whenever including those books, which could be slow and also could result in many more rules than desired. Thanks to Sol Swords and others at Intel Corp. for requesting

this enhancement and engaging in initial discussions.

Heuristic and Efficiency Improvements

Improved the efficiency of some computations involving calls of defattach with option :aokp t, in particular, of include-book events for books including such calls. Thanks to Mertcan Temel for reporting this efficiency issue and sending an include-book event, whose execution time was reduced from 24 seconds to 10 seconds by this change.

Evaluation of some large forms caused stack overflows (from ACL2 source function bad-lisp-consp). This is probably much less likely now. Thanks to Eric Smith for reporting this issue to the acl2-help list with a helpful example, which formerly caused a stack overflow for ACL2 built on SBCL but no longer does so.

The function princ$ now prints characters more rapidly.

In some cases, include-book calls may run a bit faster thanks to a couple of small changes. The primary change is for heuristic loop-stopper generation when the conclusion of a rewrite rule contains a lambda expression. Formerly all such expressions were expanded away. Now, they are expanded away in the left-hand side, and both the unexpanded and expanded versions of the left-hand side are compared to the unexpanded right-hand side (see the final Remark in the loop-stopper documentation). We have seen a 7% reduction of time for the event, (include-book "projects/x86isa/top" :dir :system).

Book certification has (in the past, and still) an optimization that avoids needless rolling back of the world during the include-book phase; see certify-book, in particular the discussion of “roll back”. Any local event within an encapsulate form was reason to roll back the world to before that encapsulate event. That is no longer the case unless the encapsulate form introduces packages in the first pass through that encapsulate form that are hidden or missing after the second pass through it, presumably because of a local include-book event in that encapsulate form.

Bug Fixes

A soundness bug in defstobj has been fixed that allowed a field to contain the type-spec (SATISFIES pred) when pred has a guard other than t, which generated an unsound guard theorem. Thanks to Eric Smith for sending the following example.

```
(defstobj s (field :type (satisfies evenp) :initially 4))

(defthm bad
  nil
  :rule-classes nil
  :hints (("Goal" :use (:instance (:guard-theorem fieldp) (x t)))))
```

A soundness bug was due to incompleteness of the prohibition against local portcullis commands. For an example, see a comment in the form `(defxdoc note-8-5 ...)`, community-books file `books/system/doc/acl2-doc.lisp`. This has been fixed by allowing local events among the portcullis commands (and treating them properly; see relevant discussion, including how `save-exec` may be helpful, in the "New Features" section above).

The use of `set-guard-checking` before calling `certify-book` was making it possible for a subsequent `include-book` to fail for that book. For example, in a fresh ACL2 session one could evaluate the command `(set-guard-checking :none)` followed by `(table foo 0 (car 3))`, and then certify a book; but including that book would then fail with a guard violation, because the `set-guard-checking` call was not saved in the certificate file, hence was not evaluated when attempting to include the book. This has been fixed by treating that situation much as local portcullis commands are now treated (see discussion about that in the "New Features" section above). The situation is explained now in the final Remark in the documentation for local-incompatibility.

Fixed an error that could occur when the break-rewrite utility is displaying failure information for an attempt to apply a linear rule containing free-variables. Thanks to Karthik Nukala and Eric Smith for sending a bug report with a replayable example.

Strengthened syntax checking for accessor expressions in `stobj-let` bindings. See a comment about this in `(defxdoc note-8-5 ...)`.

The writing of useless-runes files was sensitive to the global evisc-table, which could cause failures when reading those files. This has been fixed.

An error could occur when encountering a redundant `defwarrant` event while including a book. This has been fixed, by arranging that a `defwarrant` event always expands to the same `encapsulate` form.

An assertion error was fixed, occurring with a call of `certify-book` when the value of environment variable `"ACL2_USELESS_RUNES"` was (erroneously) `"0"`.

When the hints specified for a goal include :do-not-induct NAME for some symbol NAME other than t, :otf, :otf-flg-override, or nil, then that goal is to be skipped, giving it a "bye" as with a :by hint. This would fail however when the induction-depth-limit is reached: that is, the proof would fail immediately rather than continuing so that the skipped goal is printed upon failure at the end. The following example now has the desired behavior; thanks to Alessandro Coglio for raising this issue by sending a proof-builder example, where the :induct command failed for (as it turns out) the same reason.

```
(set-induction-depth-limit 1)
(thm (equal (append (append x y) z) (append x y z))
     :hints (("Goal"
              :induct t :do-not-induct foo :do-not *do-not-processes*)))
```

Fixed a bug in processing macro arguments with more than one occurrence of the symbol, :allow-other-keys. Thanks to Eric Smith for pointing out this bug and providing a fix. Here is an example that was failing but is now handled without error (notice that :allow-other-keys is in a value position, not a keyword position, so the duplication is legal).

```
ACL2 !>(defmacro foo (x &key y) `(list ,x ,y))

Summary
Form:  ( DEFMACRO FOO ...)
Rules: NIL
Time:  0.00 seconds (prove: 0.00, print: 0.00, other: 0.00)
 FOO
ACL2 !>(foo 3 :y 4 :z 5 :allow-other-keys t :w :allow-other-keys)


ACL2 Error in macro expansion:  ACL2 prohibits multiple :allow-other-
keys because implementations differ significantly concerning which
value to take.

ACL2 !>:q

Exiting the ACL2 read-eval-print loop.  To re-enter, execute (LP).
? (foo 3 :y 4 :z 5 :allow-other-keys t :w :allow-other-keys)
(3 4)
?
```

Pretty-printed output could be misaligned when an output string specified by the evisc-table contains a newline. Such output is now printed a bit differently, with reasonable

alignment.

Printing of results has been improved in raw-mode in cases of multiple-value return, so that stobj names are used even when at least one returned value is not an ACL2 object.

The trace level is no longer reset when entering break-rewrite or any other wormhole. Thanks to Khairul Azhar Kasmiran for raising this issue and pointing to relevant source code (GitHub Issue #1395).

Changes at the System Level

The hons-enabled features of ACL2 (hons, memoization, and fast-alists) have been included in ACL2 builds by default since Version 7.0 (January, 2015) and in all ACL2 builds since Version 7.2 (January, 2016). Now, essentially all support for building ACL2 without these features has been removed. The function `(hons-enabledp state)` and the feature `:hons` both remain, but are always true and are deprecated, scheduled for removal very soon after the next release. The feature `:acl2-mv-as-values` was always true when feature `:hons` was present, hence was always true; so it has been removed (and `#-acl2-mv-as-values` code has been eliminated).

For ACL2 builds when the host Lisp is SBCL, the Lisp optimization level is now 1 for `SPACE`, which apparently can result in more inlining than the former level of 0, and which has been seen to speed up an application while reducing memory bytes allocated. The default optimization level for `SPACE` can be set to 1 for any Lisp at build time by running `make` with argument `ACL2_SPACE=1`, and the level can be set similarly to any legal value, for example by using `ACL2_SPACE=3` for level 3. We can easily change the default for other Lisps as well, and might do so when there is evidence that this would be useful.

The function `bind-macro-args`, and several of its subfunctions, are now in `:logic` mode with verified guards. Thanks to Eric Smith for verifying guards as per verify-guards-for-system-functions.

ACL2 will now generally signal an error if a filename contains consecutive directory separators, i.e., `"//"` (in non-Windows systems).

EMACS Support

It is now possible to have more than one ACL2-doc buffer. A new buffer is created by

using `Shift-<Return>` to follow a link. Commands that are naturally specific to a given buffer (such as searching and going back) are buffer-local. Thanks to Mayank Manjrekar for the idea and for supplying an implementation (including documentation), which has been incorporated into the ACL2-doc source file, `emacs/acl2-doc.el`.

A bug has been fixed in ACL2-doc that would cause an error when attempting to bring up the acl2-only manual.

The key binding Control-TAB has been removed for the ACL2-doc browser, to avoid conflict with other uses of that key. Thanks to Alessandro Coglio for the idea.

Experimental Versions

An error could formerly occur when using the precomputed useless-runes files in ACL2(r). The useless-runes feature has now been turned off for ACL2(r). Thanks to Eric McCarthy for this change.

Subtopics

Note-8-5-books
Release notes for the ACL2 Community Books for ACL2 8.5