How Can I Do That with ACL2?
Recent Enhancements to ACL2

Matt Kaufmann and J Strother Moore
Introduction (1)

ACL2 Version 3.5 was released in May, 2009. Release note items (see :DOC release-notes) since then:

(+ 41 ; 3.6 (8/2009)
  3 ; 3.6.1 (9/2009)
100 ; 4.0 (7/2010)
  19 ; 4.1 (9/2010)
  68 ; 4.2 (1/2011)
  93) ; 4.3 (7/2011)
= 324

So far there are more than 30 items for the next release (not discussed today).
Introduction (1)

ACL2 Version 3.5 was released in May, 2009. Release note items (see :DOC release-notes) since then:

(+ 41 ; 3.6  (8/2009)
  3 ; 3.6.1  (9/2009)
 100 ; 4.0  (7/2010)
 19 ; 4.1 (9/2010)
 68 ; 4.2 (1/2011)
 93) ; 4.3 (7/2011)

= 324

So far there are more than 30 items for the next release (not discussed today).
**Introduction (1)**

ACL2 Version 3.5 was released in May, 2009. Release note items (see :DOC release-notes) since then:

(+ 41 ; 3.6  (8/2009)
  3 ; 3.6.1 (9/2009)
100 ; 4.0   (7/2010)
  19 ; 4.1  (9/2010)
  68 ; 4.2  (1/2011)
  93) ; 4.3  (7/2011)
= 324

So far there are more than 30 items for the next release (not discussed today).
Common characteristics of improvements:

- Requested by users
- Pragmatic, typically related to programming, proof control, and system infrastructure
Today’s Goal and Approach

The goal today is to raise awareness of ACL2 capabilities.

Only a few of the 324 improvements are discussed in the paper, which has hyperlinks to the documentation.

This talk will touch only briefly on those improvements; see the paper and ACL2 documentation for details.

This talk and paper are at the user level. See source code and its comments for implementation-level details.
Today’s Goal and Approach

The goal today is to raise awareness of ACL2 capabilities.

Only a few of the 324 improvements are discussed in the paper, which has hyperlinks to the documentation.

This talk will touch only briefly on those improvements; see the paper and ACL2 documentation for details.

This talk and paper are at the user level. See source code and its comments for implementation-level details.
Today’s Goal and Approach

The goal today is to raise awareness of ACL2 capabilities.

Only a few of the 324 improvements are discussed in the paper, which has hyperlinks to the documentation.

This talk will touch only briefly on those improvements; see the paper and ACL2 documentation for details.

This talk and paper are at the user level. See source code and its comments for implementation-level details.
Today’s Goal and Approach

The goal today is to raise awareness of ACL2 capabilities.

Only a few of the 324 improvements are discussed in the paper, which has hyperlinks to the documentation.

This talk will touch only briefly on those improvements; see the paper and ACL2 documentation for details.

This talk and paper are at the user level. See source code and its comments for implementation-level details.
Outline

- Programming Features
- Proof Debug, Control, and Reporting
- System-level Enhancements
- Concluding Remarks
Programming Features

- **Equality variants**
  - Defattach (*rump session talk*)
  - Return-last (*macros with side effects*)
  - Avoiding guard violations
    (with-guard-checking, ec-call)
- **Printing without state**
- **Parallelism** (*D. Rager’s rump session talk*)
- **Other recent programming support**
  - `mv-list`
  - `:redef+`, `:redef-`
  - Improvements to `trace$`, `time$`
  - `pkg-imports`
  - `file-write-date$`
  - `(append)`, `(append x)"
Programming Features

- Equality variants
- Defattach (*rump session talk*)
- Return-last (*macros with side effects*)
- Avoiding guard violations
  (*with-guard-checking, ec-call*)
- Printing without state
- Parallelism (*D. Rager’s rump session talk*)
- Other recent programming support
  - mv-list
  - :redef+, :redef-
  - Improvements to trace$, time$
  - pkg-imports
  - file-write-date$
  - (append), (append x)
Programming Features

- Equality variants
- Defattach (*rump session talk*)
- Return-last (*macros with side effects*)
- Avoiding guard violations (*with-guard-checking, ec-call*)
- Printing without state
- Parallelism (*D. Rager’s rump session talk*)
- Other recent programming support
  - `mv-list`
  - `:redef+, :redef-`
  - Improvements to `trace$, time$`
  - `pkg-imports`
  - `file-write-date$`
  - `(append), (append x)`
Programming Features

- Equality variants
- Defattach (*rump session talk*)
- Return-last (*macros with side effects*)
- Avoiding guard violations (*with-guard-checking, ec-call*)
- Printing without state
- Parallelism (*D. Rager’s rump session talk*)
- Other recent programming support
  - mv-list
  - :redef+, :redef-
  - Improvements to trace$, time$
  - pkg-imports
  - file-write-date$
  - (append), (append x)
Programming Features

- **Equality variants**
- **Defattach** (*rump session talk*)
- **Return-last** (*macros with side effects*)
- **Avoiding guard violations**
  - (*with-guard-checking, ec-call*)
- **Printing without state**
- **Parallelism** (*D. Rager’s rump session talk*)
- **Other recent programming support**
  - `mv-list`
  - `:redef+, :redef-`
  - **Improvements to** `trace$, time$`
  - `pkg-imports`
  - `file-write-date$`
  - `(append), (append x)`
Programming Features

- Equality variants
- Defattach (*rump session talk*)
- Return-last (*macros with side effects*)
- Avoiding guard violations
  (*with-guard-checking, ec-call*)
- Printing without state
- Parallelism (*D. Rager’s rump session talk*)
- Other recent programming support
  - mv-list
  - :redef+, :redef-
  - Improvements to trace$, time$
  - pkg-imports
  - file-write-date$
  - (append), (append x)
Programming Features

- Equality variants
- Defattach \textit{(rump session talk)}
- Return-last \textit{(macros with side effects)}
- Avoiding guard violations \textit{(with-guard-checking, ec-call)}
- \textbf{Printing without state}
- Parallelism \textit{(D. Rager’s rump session talk)}
- Other recent programming support
  - \texttt{mv-list}
  - \texttt{:redef+, :redef-}
  - \texttt{Improvements to trace$, time$}
  - \texttt{pkg-imports}
  - \texttt{file-write-date$}
  - \texttt{(append), (append x)}
Programming Features

- Equality variants
- Defattach \textit{(rump session talk)}
- Return-last \textit{(macros with side effects)}
- Avoiding guard violations \textit{(with-guard-checking, ec-call)}
- **Printing without state**
- Parallelism \textit{(D. Rager’s rump session talk)}
- Other recent programming support
  - mv-list
  - \texttt{:redef+}, \texttt{:redef-}
  - \textbf{Improvements to} \texttt{trace$}, \texttt{time$}
  - pkg-imports
  - file-write-date$
  - (append), (append x)
Equality variants

**DEMO**
Observation-cw, warning$-cw: unlike cw, can be turned off

Aside: Other new stateless functions include translate-cmp

Formatted creation of strings (demo)

Printing to a “string channel” (see :DOC io)

Dangerous but cool: with-local-state
Observation–cw, warning$–cw: unlike cw, can be turned off

Aside: Other new stateless functions include translate-cmp

Formatted creation of strings (demo)

Printing to a “string channel” (see :DOC io)

Dangerous but cool: with-local-state
Printing Without State

- Observation-cw, warning$-cw: unlike cw, can be turned off
  - Aside: Other new stateless functions include translate-cmp
- Formatted creation of strings (demo)
- Printing to a “string channel” (see :DOC io)
- Dangerous but cool: with-local-state
Observation-cw, warning$-cw: unlike cw, can be turned off
  Aside: Other new stateless functions include translate-cmp

Formatted creation of strings (demo)
Printing to a “string channel” (see :DOC io)

Dangerous but cool: with-local-state
Observation-\texttt{cw}, warning$\texttt{-cw}$: unlike \texttt{cw}, can be turned off

Aside: Other new stateless functions include \texttt{translate-cmp}

Formatted creation of strings (\texttt{demo})

Printing to a “string channel” (see :DOC io)

Dangerous but cool: \texttt{with-local-state}
Outline

- Programming Features
- **Proof Debug, Control, and Reporting**
- System-level Enhancements
- Concluding Remarks
Proof Debug, Control, and Reporting

- **Hints**
  - Proof-checker enhancements
    (:instructions in hints; user-defined macro commands in :instructions)
  - Parallelism in proofs (Rager’s talk)
  - Limiting proof effort
    (with-prover-step-limit)
- **Proof debugging**
- New heuristics (rewrite-cache talk; match all free-variables in type-prescription rules)
Proof Debug, Control, and Reporting

- **Hints**
  - Proof-checker enhancements
    (:instructions in hints; user-defined macro commands in :instructions)
  - Parallelism in proofs (Rager’s talk)
  - Limiting proof effort
    (with-prover-step-limit)
  - Proof debugging
  - New heuristics (rewrite-cache talk; match all free-variables in type-prescription rules)
Proof Debug, Control, and Reporting

- Hints
- Proof-checker enhancements
  (:instructions in hints; user-defined macro commands in :instructions)
- Parallelism in proofs (Rager’s talk)
  - Limiting proof effort
    (with-prover-step-limit)
- Proof debugging
- New heuristics (rewrite-cache talk; match all free-variables in type-prescription rules)
Proof Debug, Control, and Reporting

- Hints
- Proof-checker enhancements (:instructions in hints; user-defined macro commands in :instructions)
- Parallelism in proofs (Rager’s talk)
- Limiting proof effort (with-prover-step-limit)
- Proof debugging
- New heuristics (rewrite-cache talk; match all free-variables in type-prescription rules)
Proof Debug, Control, and Reporting

- **Hints**
- Proof-checker enhancements
  (:instructions in hints; user-defined macro commands in :instructions)
- Parallelism in proofs (*Rager’s talk*)
- Limiting proof effort
  (with-prover-step-limit)
- **Proof debugging**
  - New heuristics (*rewrite-cache talk; match all free-variables in type-prescription rules*)
Proof Debug, Control, and Reporting

- **Hints**
- Proof-checker enhancements
  (:instructions in hints; user-defined macro commands in :instructions)
- Parallelism in proofs (*Rager’s talk*)
- Limiting proof effort
  (with-prover-step-limit)
- **Proof debugging**
- New heuristics (*rewrite-cache talk*; match all free-variables in type-prescription rules)
Hints

- Override-hints and backtrack hints (see talk, “Integrating Testing and Interactive Theorem Proving”)

- See :DOC hints-and-the-waterfall for careful explanation of the hints mechanism.

- Distributed book
  books/hints/basic-tests.lisp has examples.

- Other improvements: :no-thanks, :backchain-limit-rw
Hints

- Override-hints and backtrack hints (see talk, “Integrating Testing and Interactive Theorem Proving”)
- See :DOC hints-and-the-waterfall for careful explanation of the hints mechanism.
- Distributed book books/hints/basic-tests.lisp has examples.
- Other improvements: :no-thanks, :backchain-limit-rw
Hints

- Override-hints and backtrack hints (see talk, “Integrating Testing and Interactive Theorem Proving”)
- See :DOC hints-and-the-waterfall for careful explanation of the hints mechanism.
- Distributed book
  books/hints/basic-tests.lisp has examples.
  Other improvements: :no-thanks, :backchain-limit-rw
Hints

- Override-hints and backtrack hints (see talk, “Integrating Testing and Interactive Theorem Proving”)
- See :DOC hints-and-the-waterfall for careful explanation of the hints mechanism.
- Distributed book
  books/hints/basic-tests.lisp has examples.
- Other improvements: :no-thanks, :backchain-limit-rw
Proof debugging

- Accumulated-persistence: useful vs. useless, by hypothesis and conclusion, :runes
- Redo-flat: now works for certify-book
- Forward-chaining-reports
- Set-inhibited-summary-types
- Gag-mode enhancements: suppression of induction schemes; can attach to print-clause-id-okp
Proof debugging

- Accumulated-persistence: useful vs. useless, by hypothesis and conclusion, :runes
- Redo-flat: now works for certify-book
- Forward-chaining-reports
- Set-inhibited-summary-types
- Gag-mode enhancements: suppression of induction schemes; can attach to print-clause-id-okp
Proof debugging

- **Accumulated-persistence**: useful vs. useless, by hypothesis and conclusion, :runes
- **Redo-flat**: now works for certify-book
- **Forward-chaining-reports**
  - Set-inhibited-summary-types
  - **Gag-mode enhancements**: suppression of induction schemes; can attach to print-clause-id-okp
Proof debugging

- Accumulated-persistence: useful vs. useless, by hypothesis and conclusion, runes
- Redo-flat: now works for certify-book
- Forward-chaining-reports
- Set-inhibited-summary-types
  - Gag-mode enhancements: suppression of induction schemes; can attach to print-clause-id-okp
Proof debugging

- Accumulated-persistence: useful vs. useless, by hypothesis and conclusion, runes
- Redo-flat: now works for certify-book
- Forward-chaining-reports
- Set-inhibited-summary-types
- Gag-mode enhancements: suppression of induction schemes; can attach to print-clause-id-okp
Outline

- Programming Features
- Proof Debug, Control, and Reporting
- **System-level Enhancements**
- Concluding Remarks
System-level Enhancements

- Two-run certification to avoid trust tags (see :DOC set-write-acl2x)
- Certifying a subset of the distributed books
- Size and breakdown of ACL2 source code
- An \texttt{mbe} restriction lifted (\textit{OK to have mbe in non-executable definition in encapsulate})
- Aborting just one ld level (:p! instead of :a!)
System-level Enhancements

- Two-run certification to avoid trust tags (see :DOC set-write-acl2x)
- Certifying a subset of the distributed books
- Size and breakdown of ACL2 source code
- An \texttt{mbe} restriction lifted \textit{(OK to have \texttt{mbe} in non-executable definition in encapsulate)}
- Aborting just one \texttt{ld} level \textit{(\texttt{:p!} instead of \texttt{:a!})}
System-level Enhancements

- Two-run certification to avoid trust tags (see :DOC set-write-acl2x)
- Certifying a subset of the distributed books
- **Size and breakdown of ACL2 source code**
  - An `mbe` restriction lifted (*OK to have `mbe` in non-executable definition in encapsulate*)
  - Aborting just one `ld` level (`:p!` instead of `:a!`
System-level Enhancements

- Two-run certification to avoid trust tags (see :DOC set-write-acl2x)
- Certifying a subset of the distributed books
- **Size and breakdown of ACL2 source code**
  - An `mbe` restriction lifted *(OK to have `mbe` in non-executable definition in encapsulate)*
- Aborting just one ld level (:p! instead of :a!)
System-level Enhancements

- Two-run certification to avoid trust tags (see :DOC set-write-acl2x)
- Certifying a subset of the distributed books
- **Size and breakdown of ACL2 source code**
- An `mbe` restriction lifted (*OK to have mbe in non-executable definition in encapsulate*)
- Aborting just one ld level (:p! instead of :a!)
doc/acl2-code-size.txt, Version 4.3:

CODE LINES:
  99666 lines, 4366258 characters
COMMENT LINES:
  52839 lines, 3120461 characters
BLANK LINES (excluding documentation):
  23319 lines, 24823 characters
DOCUMENTATION LINES:
  80275 lines, 3586285 characters
TOTAL:
  256099 lines, 11097827 characters
Concluding Remarks

User community feedback is critical for evolution of ACL2.

Thanks to Sandip Ray for feedback on a draft of this paper, and to NSF (Grant Nos. CCF-0945316 and CNS-0910913).
Concluding Remarks

User community feedback is critical for evolution of ACL2.

Thanks to Sandip Ray for feedback on a draft of this paper, and to NSF (Grant Nos. CCF-0945316 and CNS-0910913).