Abbreviated Output for Input in ACL2: An Implementation Case Study

Matt Kaufmann
University of Texas at Austin
kaufmann@cs.utexas.edu

This material is based in part upon work supported by DARPA and the National Science Foundation under Grant No. CNS-0429591 and by the National Science Foundation under Grant Nos. EIA-0303609 and ISS-0417413.

OVERVIEW

Talk objectives:

- Introduce ACL2 abbreviated output and the new *iprinting* mechanism for reading it back in;
- Impart a sense of current ACL2 development.

For the paper, the second is the main goal.

In today's talk, I'll focus on the first, while working in comments about ACL2 development.

HISTORY AND MOTIVATION

- 1973 Tech Report: Bob Boyer's pretty-printing algorithm
- 1989: Boyer and J Moore start ACL2 and enhance algorithm with evisceration, e.g. to print `(A B ...)` instead of `(A B C D E F G)`
  - Drawback: Can't read an eviscerated form
- With `HONS` extension of ACL2 by Boyer and Warren Hunt, very large ("galactic") objects can be fatal to print.

Goal: provide a capability that addresses these concerns

Example of ACL2 support for user interaction, a major priority

Iprinting

The term “iprinting” could stand for:

- interactive printing
- “`#@i#`” printing (by the way, we call `#@i#` an “iprint token”)
- “index printing”

Well, at least it’s not “iPrinting”....

DEMO slide 1

```
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil))

ACL2 Query (:SET-IPRINT): Action (T, NIL, RESET,
RESET-ENABLE, SAME, Q or ?):  nil ; old behavior

ACL2 Observation in SET-IPRINT: Ipprinting remains
disabled.

ACL2 Query (:SET-EVISC-TUPLE): Do you wish to set
TERM-EVISC-TUPLE? (Y, N, ALL, REST, Q, ABORT or ?):
all
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#) E ...)  U V W ...)
```

DEMO slide 1

```
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil))

ACL2 Query (:SET-IPRINT): Action (T, NIL, RESET,
RESET-ENABLE, SAME, Q or ?):  nil ; old behavior

ACL2 Observation in SET-IPRINT: Ipprinting remains
disabled.

ACL2 Query (:SET-EVISC-TUPLE): Do you wish to set
TERM-EVISC-TUPLE? (Y, N, ALL, REST, Q, ABORT or ?):
all
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#) E ...)  U V W ...)
```

DEMO slide 1

```
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil))

ACL2 Query (:SET-IPRINT): Action (T, NIL, RESET,
RESET-ENABLE, SAME, Q or ?):  nil ; old behavior

ACL2 Observation in SET-IPRINT: Ipprinting remains
disabled.

ACL2 Query (:SET-EVISC-TUPLE): Do you wish to set
TERM-EVISC-TUPLE? (Y, N, ALL, REST, Q, ABORT or ?):
all
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#) E ...)  U V W ...)
```

DEMO slide 1

```
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil))

ACL2 Query (:SET-IPRINT): Action (T, NIL, RESET,
RESET-ENABLE, SAME, Q or ?):  nil ; old behavior

ACL2 Observation in SET-IPRINT: Ipprinting remains
disabled.

ACL2 Query (:SET-EVISC-TUPLE): Do you wish to set
TERM-EVISC-TUPLE? (Y, N, ALL, REST, Q, ABORT or ?):
all
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#) E ...)  U V W ...)
```

DEMO slide 1

```
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil))

ACL2 Query (:SET-IPRINT): Action (T, NIL, RESET,
RESET-ENABLE, SAME, Q or ?):  nil ; old behavior

ACL2 Observation in SET-IPRINT: Ipprinting remains
disabled.

ACL2 Query (:SET-EVISC-TUPLE): Do you wish to set
TERM-EVISC-TUPLE? (Y, N, ALL, REST, Q, ABORT or ?):
all
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#) E ...)  U V W ...)
```

DEMO slide 1

```
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil))

ACL2 Query (:SET-IPRINT): Action (T, NIL, RESET,
RESET-ENABLE, SAME, Q or ?):  nil ; old behavior

ACL2 Observation in SET-IPRINT: Ipprinting remains
disabled.

ACL2 Query (:SET-EVISC-TUPLE): Do you wish to set
TERM-EVISC-TUPLE? (Y, N, ALL, REST, Q, ABORT or ?):
all
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#) E ...)  U V W ...)
```

DEMO slide 1

```
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil))

ACL2 Query (:SET-IPRINT): Action (T, NIL, RESET,
RESET-ENABLE, SAME, Q or ?):  nil ; old behavior

ACL2 Observation in SET-IPRINT: Ipprinting remains
disabled.

ACL2 Query (:SET-EVISC-TUPLE): Do you wish to set
TERM-EVISC-TUPLE? (Y, N, ALL, REST, Q, ABORT or ?):
all
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#) E ...)  U V W ...)
```

DEMO slide 1

```
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil))

ACL2 Query (:SET-IPRINT): Action (T, NIL, RESET,
RESET-ENABLE, SAME, Q or ?):  nil ; old behavior

ACL2 Observation in SET-IPRINT: Ipprinting remains
disabled.

ACL2 Query (:SET-EVISC-TUPLE): Do you wish to set
TERM-EVISC-TUPLE? (Y, N, ALL, REST, Q, ABORT or ?):
all
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#) E ...)  U V W ...)
```

DEMO slide 1

```
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil))

ACL2 Query (:SET-IPRINT): Action (T, NIL, RESET,
RESET-ENABLE, SAME, Q or ?):  nil ; old behavior

ACL2 Observation in SET-IPRINT: Ipprinting remains
disabled.

ACL2 Query (:SET-EVISC-TUPLE): Do you wish to set
TERM-EVISC-TUPLE? (Y, N, ALL, REST, Q, ABORT or ?):
all
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#) E ...)  U V W ...)
```

DEMO slide 1

```
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil))

ACL2 Query (:SET-IPRINT): Action (T, NIL, RESET,
RESET-ENABLE, SAME, Q or ?):  nil ; old behavior

ACL2 Observation in SET-IPRINT: Ipprinting remains
disabled.

ACL2 Query (:SET-EVISC-TUPLE): Do you wish to set
TERM-EVISC-TUPLE? (Y, N, ALL, REST, Q, ABORT or ?):
all
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#) E ...)  U V W ...)
```

DEMO slide 1

```
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil))

ACL2 Query (:SET-IPRINT): Action (T, NIL, RESET,
RESET-ENABLE, SAME, Q or ?):  nil ; old behavior

ACL2 Observation in SET-IPRINT: Ipprinting remains
disabled.

ACL2 Query (:SET-EVISC-TUPLE): Do you wish to set
TERM-EVISC-TUPLE? (Y, N, ALL, REST, Q, ABORT or ?):
all
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#) E ...)  U V W ...)
```

DEMO slide 2

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Ipprinting has been
enabled.
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#@1#) E . #@2#) U V W . #@3#)
ACL2 !>'#@1#
(C D)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B (#@4#) E . #@5#) U V W . #@6#)
ACL2 !>(without-evisc
         '((A B (#@1#) E . #@2#) U V W . #@3#))
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 2

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Ipprinting has been
enabled.
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#@1#) E . #@2#) U V W . #@3#)
ACL2 !>'#@1#
(C D)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B (#@4#) E . #@5#) U V W . #@6#)
ACL2 !>(without-evisc
         '((A B (#@1#) E . #@2#) U V W . #@3#))
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 2

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Ipprinting has been
enabled.
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#@1#) E . #@2#) U V W . #@3#)
ACL2 !>'#@1#
(C D)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B (#@4#) E . #@5#) U V W . #@6#)
ACL2 !>(without-evisc
         '((A B (#@1#) E . #@2#) U V W . #@3#))
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 2

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#@1#) E . #@2#) U V W . #@3#)
ACL2 !>'#@1#
(C D)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B (#@4#) E . #@5#) U V W . #@6#)
ACL2 !>(without-evisc
         '((A B (#@1#) E . #@2#) U V W . #@3#))
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 2

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#@1#) E . #@2#) U V W . #@3#)
ACL2 !>'#@1#
(C D)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B (#@4#) E . #@5#) U V W . #@6#)
ACL2 !>(without-evisc
         '((A B (#@1#) E . #@2#) U V W . #@3#))
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 2

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Ipprinting has been
enabled.
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#@1#) E . #@2#) U V W . #@3#)
ACL2 !>'#@1#
(C D)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B (#@4#) E . #@5#) U V W . #@6#)
ACL2 !>(without-evisc
         '((A B (#@1#) E . #@2#) U V W . #@3#))
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 2

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#@1#) E . #@2#) U V W . #@3#)
ACL2 !>'#@1#
(C D)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B (#@4#) E . #@5#) U V W . #@6#)
ACL2 !>(without-evisc
         '((A B (#@1#) E . #@2#) U V W . #@3#))
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 2

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#@1#) E . #@2#) U V W . #@3#)
ACL2 !>'#@1#
(C D)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B (#@4#) E . #@5#) U V W . #@6#)
ACL2 !>(without-evisc
         '((A B (#@1#) E . #@2#) U V W . #@3#))
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 2

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#@1#) E . #@2#) U V W . #@3#)
ACL2 !>'#@1#
(C D)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B (#@4#) E . #@5#) U V W . #@6#)
ACL2 !>(without-evisc
         '((A B (#@1#) E . #@2#) U V W . #@3#))
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 2

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
ACL2 !>'((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
((A B (#@1#) E . #@2#) U V W . #@3#)
ACL2 !>'#@1#
(C D)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B (#@4#) E . #@5#) U V W . #@6#)
ACL2 !>(without-evisc
         '((A B (#@1#) E . #@2#) U V W . #@3#))
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 3

Next example:

1. Return to default settings.
2. Designate only the *term-evisc-tuple*: `:TERM` but not `:LD`, `:TRACE`, or `:ABBREV`).

(See `:doc set-evisc-tuple`.)

```
ACL2 !>(set-evisc-tuple :default ; Avoid query:
                        :sites :all)
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil)
                        :sites :term)
 (:TERM)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 3

Next example:

1. Return to default settings.
2. Designate only the *term-evisc-tuple*: `:TERM` but not `:LD`, `:TRACE`, or `:ABBREV`).

(See `:doc set-evisc-tuple`.)

```
ACL2 !>(set-evisc-tuple :default ; Avoid query:
                        :sites :all)
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil)
                        :sites :term)
 (:TERM)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 3

Next example:

1. Return to default settings.
2. Designate only the *term-evisc-tuple*: `:TERM` but not `:LD`, `:TRACE`, or `:ABBREV`).

(See `:doc set-evisc-tuple`.)

```
ACL2 !>(set-evisc-tuple :default ; Avoid query:
                        :sites :all)
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil)
                        :sites :term)
 (:TERM)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 3

Next example:

1. Return to default settings.
2. Designate only the *term-evisc-tuple*: `:TERM` but not `:LD`, `:TRACE`, or `:ABBREV`).

(See `:doc set-evisc-tuple`.)

```
ACL2 !>(set-evisc-tuple :default ; Avoid query:
                        :sites :all)
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil)
                        :sites :term)
 (:TERM)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 3

Next example:

1. Return to default settings.
2. Designate only the *term-evisc-tuple*: `:TERM` but not `:LD`, `:TRACE`, or `:ABBREV`).

(See `:doc set-evisc-tuple`.)

```
ACL2 !>(set-evisc-tuple :default ; Avoid query:
                        :sites :all)
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil)
                        :sites :term)
 (:TERM)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 3

Next example:

1. Return to default settings.
2. Designate only the *term-evisc-tuple*: `:TERM` but not `:LD`, `:TRACE`, or `:ABBREV`).

(See `:doc set-evisc-tuple`.)

```
ACL2 !>(set-evisc-tuple :default ; Avoid query:
                        :sites :all)
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil)
                        :sites :term)
 (:TERM)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 3

Next example:

1. Return to default settings.
2. Designate only the *term-evisc-tuple*: `:TERM` but not `:LD`, `:TRACE`, or `:ABBREV`).

(See `:doc set-evisc-tuple`.)

```
ACL2 !>(set-evisc-tuple :default ; Avoid query:
                        :sites :all)
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil)
                        :sites :term)
 (:TERM)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 3

Next example:

1. Return to default settings.
2. Designate only the *term-evisc-tuple*: `:TERM` but not `:LD`, `:TRACE`, or `:ABBREV`).

(See `:doc set-evisc-tuple`.)

```
ACL2 !>(set-evisc-tuple :default ; Avoid query:
                        :sites :all)
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil)
                        :sites :term)
 (:TERM)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 3

Next example:

1. Return to default settings.
2. Designate only the *term-evisc-tuple*: `:TERM` but not `:LD`, `:TRACE`, or `:ABBREV`).

(See `:doc set-evisc-tuple`.)

```
ACL2 !>(set-evisc-tuple :default ; Avoid query:
                        :sites :all)
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil)
                        :sites :term)
 (:TERM)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 3

Next example:

1. Return to default settings.
2. Designate only the *term-evisc-tuple*: `:TERM` but not `:LD`, `:TRACE`, or `:ABBREV`).

(See `:doc set-evisc-tuple`.)

```
ACL2 !>(set-evisc-tuple :default ; Avoid query:
                        :sites :all)
 (:TERM :LD :TRACE :ABBREV)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>(set-evisc-tuple (evisc-tuple 3 4 nil nil)
                        :sites :term)
 (:TERM)
ACL2 !>'((A B (#@1#) E . #@2#) U V W . #@3#)
((A B ((C D)) E F G)
 U V W X Y 1 2 3 4 5)
ACL2 !>
```

DEMO slide 4

```
ACL2 !>(fms "~x0~%"
            (list (cons #\0 '((A B (#@1#) E . #@2#)
                              U V W . #@3#)))
            *standard-co*
            state
            (term-evisc-tuple nil state))

((A B (#@7#) E . #@8#) U V W . #@9#)
<state>
ACL2 !>(fms "~x0~%"
            (list (cons #\0 '((A B (#@1#) E . #@2#)
                              U V W . #@3#)))
            *standard-co*
            state
            nil)

((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
<state>
ACL2 !>
```

DEMO slide 4

```
ACL2 !>(fms "~x0~%"
            (list (cons #\0 '((A B (#@1#) E . #@2#)
                              U V W . #@3#)))
            *standard-co*
            state
            (term-evisc-tuple nil state))

((A B (#@7#) E . #@8#) U V W . #@9#)
<state>
ACL2 !>(fms "~x0~%"
            (list (cons #\0 '((A B (#@1#) E . #@2#)
                              U V W . #@3#)))
            *standard-co*
            state
            nil)

((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
<state>
ACL2 !>
```

DEMO slide 4

```
ACL2 !>(fms "~x0~%"
            (list (cons #\0 '((A B (#@1#) E . #@2#)
                              U V W . #@3#)))
            *standard-co*
            state
            (term-evisc-tuple nil state))

((A B (#@7#) E . #@8#) U V W . #@9#)
<state>
ACL2 !>(fms "~x0~%"
            (list (cons #\0 '((A B (#@1#) E . #@2#)
                              U V W . #@3#)))
            *standard-co*
            state
            nil)

((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
<state>
ACL2 !>
```

DEMO slide 4

```
ACL2 !>(fms "~x0~%"
            (list (cons #\0 '((A B (#@1#) E . #@2#)
                              U V W . #@3#)))
            *standard-co*
            state
            (term-evisc-tuple nil state))

((A B (#@7#) E . #@8#) U V W . #@9#)
<state>
ACL2 !>(fms "~x0~%"
            (list (cons #\0 '((A B (#@1#) E . #@2#)
                              U V W . #@3#)))
            *standard-co*
            state
            nil)

((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
<state>
ACL2 !>
```

DEMO slide 4

```
ACL2 !>(fms "~x0~%"
            (list (cons #\0 '((A B (#@1#) E . #@2#)
                              U V W . #@3#)))
            *standard-co*
            state
            (term-evisc-tuple nil state))

((A B (#@7#) E . #@8#) U V W . #@9#)
<state>
ACL2 !>(fms "~x0~%"
            (list (cons #\0 '((A B (#@1#) E . #@2#)
                              U V W . #@3#)))
            *standard-co*
            state
            nil)

((A B ((C D)) E F G) U V W X Y 1 2 3 4 5)
<state>
ACL2 !>
```

ROBUSTNESS (example)

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Iprinting has been enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>'(a b c d e)
(A B C . #@2#)
ACL2 !>'(#@5# #@2#)
****************************************************
************ ABORTING from raw Lisp ************
Error:  Out-of-bounds index in #@5#.  See :DOC set-iprint.

****************************************************
```

Note: We don't get noise from reading stuff after the error.

ROBUSTNESS (example)

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Iprinting has been enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>'(a b c d e)
(A B C . #@2#)
ACL2 !>'(#@5# #@2#)
**********************************************
************ ABORTING from raw Lisp ***********
Error:  Out-of-bounds index in #@5#.  See :DOC set-iprint.

**********************************************
```

Note: We don't get noise from reading stuff after the error.

ROBUSTNESS (example)

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Iprinting has been enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>'(a b c d e)
(A B C . #@2#)
ACL2 !>'(#@5# #@2#)
****************************************************
************ ABORTING from raw Lisp ************
Error:  Out-of-bounds index in #@5#.  See :DOC set-iprint.

****************************************************
```

Note: We don't get noise from reading stuff after the error.

ROBUSTNESS (example)

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Ipriting has been enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>'(a b c d e)
(A B C . #@2#)
ACL2 !>'(#@5# #@2#)
****************************************************
************ ABORTING from raw Lisp ************
Error:  Out-of-bounds index in #@5#.  See :DOC set-iprint.

****************************************************
```

Note: We don't get noise from reading stuff after the error.

ROBUSTNESS (example)

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Ipringting has been enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>'(a b c d e)
(A B C . #@2#)
ACL2 !>'(#@5# #@2#)
****************************************************
************ ABORTING from raw Lisp ************
Error:  Out-of-bounds index in #@5#.  See :DOC set-iprint.

****************************************************
```

Note: We don't get noise from reading stuff after the error.

ROBUSTNESS (example)

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Ipriting has been enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>'(a b c d e)
(A B C . #@2#)
ACL2 !>'(#@5# #@2#)
****************************************************
************ ABORTING from raw Lisp ************
Error:  Out-of-bounds index in #@5#.  See :DOC set-iprint.

****************************************************
```

Note: We don't get noise from reading stuff after the error.

ROBUSTNESS (example)

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Ipriting has been enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>'(a b c d e)
(A B C . #@2#)
ACL2 !>'(#@5# #@2#)
****************************************************
************ ABORTING from raw Lisp ************
Error:  Out-of-bounds index in #@5#.  See :DOC set-iprint.

****************************************************
```

Note: We don't get noise from reading stuff after the error.

ROBUSTNESS (example)

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Ipringting has been enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>'(a b c d e)
(A B C . #@2#)
ACL2 !>'(#@5# #@2#)
****************************************************
************ ABORTING from raw Lisp ************
Error:  Out-of-bounds index in #@5#.  See :DOC set-iprint.

****************************************************
```

Note: We don't get noise from reading stuff after the error.

TRANSITIONING TO IPRINTING

```
ACL2 !>(defun foo '(a b c d e f g h i))

ACL2 Error in ( DEFUN FOO ...):  A definition must
be given three or more arguments, but
(FOO '(A B C D E F G ...))  has length only 2.
(See :DOC set-iprint to be able to see elided
values in this message.)
```

Let's take the advice given above.

TRANSITIONING TO IPRINTING

```
ACL2 !>(defun foo '(a b c d e f g h i))

ACL2 Error in ( DEFUN FOO ...):  A definition must
be given three or more arguments, but
(FOO '(A B C D E F G ...))  has length only 2.
(See :DOC set-iprint to be able to see elided
values in this message.)
```

Let's take the advice given above.

TRANSITIONING TO IPRINTING

```
ACL2 !>(defun foo '(a b c d e f g h i))

ACL2 Error in ( DEFUN FOO ...):  A definition must
be given three or more arguments, but
(FOO '(A B C D E F G ...))  has length only 2.
(See :DOC set-iprint to be able to see elided
values in this message.)
```

Let's take the advice given above.

TRANSITIONING TO IPRINTING

```
ACL2 !>(defun foo '(a b c d e f g h i))

ACL2 Error in ( DEFUN FOO ...):  A definition must
be given three or more arguments, but
(FOO '(A B C D E F G ...))  has length only 2.
(See :DOC set-iprint to be able to see elided
values in this message.)
```

Let's take the advice given above.

TRANSITIONING TO IPRINTING (cont.)

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Iprinting has been
enabled.
ACL2 !>(defun foo '(a b c d e f g h i))


ACL2 Error in ( DEFUN FOO ...):  A definition must
be given three or more arguments, but
(FOO '(A B C D E F G . #@2#)) has length only 2.
```

Now we can explore `#@2#` if we like, as before.

TRANSITIONING TO IPRINTING (cont.)

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Iprinting has been
enabled.
ACL2 !>(defun foo '(a b c d e f g h i))


ACL2 Error in ( DEFUN FOO ...):  A definition must
be given three or more arguments, but
(FOO '(A B C D E F G . #@2#)) has length only 2.
```

Now we can explore `#@2#` if we like, as before.

TRANSITIONING TO IPRINTING (cont.)

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Iprinting has been
enabled.
ACL2 !>(defun foo '(a b c d e f g h i))


ACL2 Error in ( DEFUN FOO ...):  A definition must
be given three or more arguments, but
(FOO '(A B C D E F G . #@2#)) has length only 2.
```

Now we can explore #@2# if we like, as before.

TRANSITIONING TO IPRINTING (cont.)

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Iprinting has been
enabled.
ACL2 !>(defun foo '(a b c d e f g h i))


ACL2 Error in ( DEFUN FOO ...):  A definition must
be given three or more arguments, but
(FOO '(A B C D E F G . #@2#)) has length only 2.
```

Now we can explore #@2# if we like, as before.

TRANSITIONING TO IPRINTING (cont.)

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
ACL2 !>(defun foo '(a b c d e f g h i))


ACL2 Error in ( DEFUN FOO ...):  A definition must
be given three or more arguments, but
(FOO '(A B C D E F G . #@2#)) has length only 2.
```

Now we can explore #@2# if we like, as before.

TRANSITIONING TO IPRINTING (cont.)

```
ACL2 !>(set-iprint t)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
ACL2 !>(defun foo '(a b c d e f g h i))


ACL2 Error in ( DEFUN FOO ...):  A definition must
be given three or more arguments, but
(FOO '(A B C D E F G . #@2#)) has length only 2.
```

Now we can explore #@2# if we like, as before.

ROLLOVER

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>:mini-proveall
...  output elided ...
Subgoal *1/1'
(IMPLIES (AND #@1302# #@1303# .  #@1304#)
         (ORDERED-SYMBOL-ALISTP #@1305#)).
...  output elided ...
:EOF
ACL2 !>'(a b c d e f)
(A B C . #@1#)
ACL2 !>'(#@1# #@1305#)
((D E F) (REMOVE-FIRST-PAIR KEY L))
ACL2 !>
```

ROLLOVER

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Iprinting has been
enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>:mini-proveall
...  output elided ...
Subgoal *1/1'
(IMPLIES (AND #@1302# #@1303# .  #@1304#)
         (ORDERED-SYMBOL-ALISTP #@1305#)).
...  output elided ...
:EOF
ACL2 !>'(a b c d e f)
(A B C . #@1#)
ACL2 !>'(#@1# #@1305#)
((D E F) (REMOVE-FIRST-PAIR KEY L))
ACL2 !>
```

ROLLOVER

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Iprinting has been
enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>:mini-proveall
...  output elided ...
Subgoal *1/1'
(IMPLIES (AND #@1302# #@1303# .  #@1304#)
         (ORDERED-SYMBOL-ALISTP #@1305#)).
...  output elided ...
:EOF
ACL2 !>'(a b c d e f)
(A B C . #@1#)
ACL2 !>'(#@1# #@1305#)
((D E F) (REMOVE-FIRST-PAIR KEY L))
ACL2 !>
```

ROLLOVER

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>:mini-proveall
...  output elided ...
Subgoal *1/1'
(IMPLIES (AND #@1302# #@1303# .  #@1304#)
         (ORDERED-SYMBOL-ALISTP #@1305#)).
...  output elided ...
:EOF
ACL2 !>'(a b c d e f)
(A B C . #@1#)
ACL2 !>'(#@1# #@1305#)
((D E F) (REMOVE-FIRST-PAIR KEY L))
ACL2 !>
```

ROLLOVER

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>:mini-proveall
...  output elided ...
Subgoal *1/1'
(IMPLIES (AND #@1302# #@1303# .  #@1304#)
         (ORDERED-SYMBOL-ALISTP #@1305#)).
...  output elided ...
:EOF
ACL2 !>'(a b c d e f)
(A B C . #@1#)
ACL2 !>'(#@1# #@1305#)
((D E F) (REMOVE-FIRST-PAIR KEY L))
ACL2 !>
```

ROLLOVER

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>:mini-proveall
...  output elided ...
Subgoal *1/1'
(IMPLIES (AND #@1302# #@1303# .  #@1304#)
         (ORDERED-SYMBOL-ALISTP #@1305#)).
...  output elided ...
:EOF
ACL2 !>'(a b c d e f)
(A B C . #@1#)
ACL2 !>'(#@1# #@1305#)
((D E F) (REMOVE-FIRST-PAIR KEY L))
ACL2 !>
```

ROLLOVER

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>:mini-proveall
...  output elided ...
Subgoal *1/1'
(IMPLIES (AND #@1302# #@1303# .  #@1304#)
         (ORDERED-SYMBOL-ALISTP #@1305#)).
...  output elided ...
:EOF
ACL2 !>'(a b c d e f)
(A B C . #@1#)
ACL2 !>'(#@1# #@1305#)
((D E F) (REMOVE-FIRST-PAIR KEY L))
ACL2 !>
```

ROLLOVER

```
ACL2 !>(set-evisc-tuple (evisc-tuple 2 3 nil nil)
                        :iprint t
                        :sites :all)

ACL2 Observation in SET-IPRINT: Ipringting has been
enabled.
 (:TERM :LD :TRACE . #@1#)
ACL2 !>:mini-proveall
...  output elided ...
Subgoal *1/1'
(IMPLIES (AND #@1302# #@1303# .  #@1304#)
         (ORDERED-SYMBOL-ALISTP #@1305#)).
...  output elided ...
:EOF
ACL2 !>'(a b c d e f)
(A B C . #@1#)
ACL2 !>'(#@1# #@1305#)
((D E F) (REMOVE-FIRST-PAIR KEY L))
ACL2 !>
```

DESIGN AND IMPLEMENTATION CHALLENGES

See the paper for details.

- Reclaiming storage (rollover)
- Defining when an iprint token is legal input and providing an appropriate error message when it is not
- Use of ACL2 arrays
  - Reacting suitably to interrupts and “wormhole” (brr) printing
  - Maintaining invariants
- Disabling iprinting during `certify-book`: soundness!
- Modifying the lisp reader
- Challenges in implementing without-evisc
- Protecting ACL2 `state`: untouchables and `make-event`

SEE ALSO:

- `set-iprint`
- `evisc-tuple`
- `set-trace-evisc-tuple`
- `set-evisc-tuple`
- `without-evisc`

... and of course, see :DOC note-3-5.

CONCLUSION

- Galactic objects motivated iprinting: more evisceration, but with ability to recover hidden values
- Scalable (rollover, arrays); robust (error detection)
- Careful documentation (*time-consuming!*), with hyperlinks
- Minimal burden on user: improved interfaces for setting evisc tuples, gentle transition to iprinting

Read the paper if you're interested in more about the design and implementation considerations.

Acknowledgements:
Ideas/collaboration: J Moore
Funding: ForrestHunt, Inc., NSF, DARPA
Encouragement: Bob Boyer, Warren Hunt
Discussions, useful feedback: Sandip Ray
Useful feedback: The referees

CONCLUSION

- Galactic objects motivated iprinting: more evisceration, but with ability to recover hidden values
- Scalable (rollover, arrays); robust (error detection)
- Careful documentation (*time-consuming!*), with hyperlinks
- Minimal burden on user: improved interfaces for setting evisc tuples, gentle transition to iprinting

Read the paper if you're interested in more about the design and implementation considerations.

Acknowledgements:
Ideas/collaboration: J Moore
Funding: ForrestHunt, Inc., NSF, DARPA
Encouragement: Bob Boyer, Warren Hunt
Discussions, useful feedback: Sandip Ray
Useful feedback: The referees

CONCLUSION, Quoting from Paper

ACL2 is much more than a reasoning engine, to an extent well beyond most or all other mechanized theorem provers. It is a highly interactive system, providing a programming environment with a read-eval-print loop that supports large objects and configurable I/O, and with "administrative" functions including `make-event` and the management of book certificates. The addition of new features to ACL2 thus requires attention to issues as described in the bulleted list above: harmony with existing ACL2 features (especially, but not solely, with respect to soundness); efficiency and scalability; user interaction; documentation, including error and warning messages; and prioritization.