ACL2(ml): Machine Learning for ACL2

J. Heras and E. Komendantskaya.
Queries to jonathanheras@computing.dundee.ac.uk

January 10, 2014

Abstract

This manual describes ACL2(ml), a machine-learning extension to the Emacs interface for ACL2. The description of the underlying methods implemented in ACL2(ml) can be found in [1].

Contents

1 Requirements

Before installing ACL2(ml), you need to download, install and configure the following software.

- ACL2.
- Emacs.

You can follow the instructions presented in `http://www.cs.utexas.edu/users/moore/publications/acl2-programming-exercises1.html`.

2 Installation

Before using ACL2(ml), it is necessary to configure some variables. First of all, open your *.emacs* file. This file is usually located in */home/user/.emacs* in Linux, for aquamacs this is */Library/Preferences/Aquamacs Emacs/Preferences.el*. At the end of the file include the line:

```
(load-file "ACL2(ml)-location/main.el")
```

where `ACL2(ml)-location` must be changed with the path to the folder where you have downloaded ACL2(ml).

Now, go to the folder where you have downloaded ACL2(ml) and open the file *main.el*. Modify the following constants:

- **home-dir**: you must replace the current path assigned to this constant with the path where you have downloaded ACL2(ml).
- **acl2-dir**: you must replace the current path assigned to this constant with the path to your executable image of ACL2.

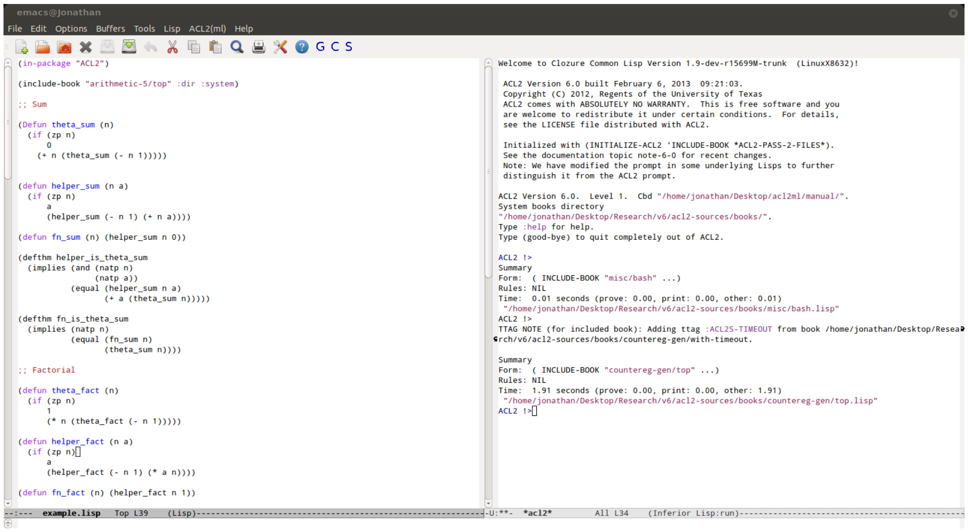
This finishes the installation of ACL2(ml).

3 Using ACL2(ml)

To illustrate the use of ACL2(ml), we will use the file `example.lisp` which can be find in the same folder of this manual. The example `example.lisp` includes the definition of several recursive and tail recursive arithmetic functions and the proof of their equivalences.

3.1 Getting started

Open file `example.lisp` using Emacs (the file `example.lisp` contains the definition of several recursive and tail recursive arithmetic functions and the equivalence among them), start ACL2(ml) using M-x `start-acl2ml`, this splits the Emacs window vertically, keeping the file `example.lisp` on the left window and starting ACL2 in the right screen – some ACL2 libraries are automatically loaded. If you have installed everything properly, your Emacs screen will look like Figure 1. As you can see, the Emacs interface includes a new menu called ACL2(ml) and three new buttons (G, C, S) in the menu bar.


emacs@Jonathan
File Edit Options Buffers Tools Lisp ACL2(ml) Help
G C S
--:--- example.lisp   Top L39   (Lisp)
-U:**- *acl2*   All L34   (Inferior Lisp:run)

```lisp
(in-package "ACL2")

(include-book "arithmetic-5/top" :dir :system)

;; Sum

(Defun theta_sum (n)
  (if (zp n)
      0
    (+ n (theta_sum (- n 1)))))


(defun helper_sum (n a)
  (if (zp n)
      a
      (helper_sum (- n 1) (+ n a))))

(defun fn_sum (n) (helper_sum n 0))

(defthm helper_is_theta_sum
  (implies (and (natp n)
                (natp a))
           (equal (helper_sum n a)
                  (+ a (theta_sum n)))))

(defthm fn_is_theta_sum
  (implies (natp n)
           (equal (fn_sum n)
                  (theta_sum n))))

;; Factorial

(defun theta_fact (n)
  (if (zp n)
      1
      (* n (theta_fact (- n 1)))))

(defun helper_fact (n a)
  (if (zp n)
      a
      (helper_fact (- n 1) (* a n))))

(defun fn_fact (n) (helper_fact n 1))
```

```
Welcome to Clozure Common Lisp Version 1.9-dev-r15699M-trunk  (LinuxX8632)!

 ACL2 Version 6.0 built February 6, 2013  09:21:03.
 Copyright (C) 2012, Regents of the University of Texas
 ACL2 comes with ABSOLUTELY NO WARRANTY.  This is free software and you
 are welcome to redistribute it under certain conditions.  For details,
 see the LICENSE file distributed with ACL2.

 Initialized with (INITIALIZE-ACL2 'INCLUDE-BOOK *ACL2-PASS-2-FILES*).
 See the documentation topic note-6-0 for recent changes.
 Note: We have modified the prompt in some underlying Lisps to further
 distinguish it from the ACL2 prompt.

ACL2 Version 6.0.  Level 1.  Cbd "/home/jonathan/Desktop/acl2ml/manual/".
System books directory
"/home/jonathan/Desktop/Research/v6/acl2-sources/books/".
Type :help for help.
Type (good-bye) to quit completely out of ACL2.

ACL2 !>
Summary
Form:  ( INCLUDE-BOOK "misc/bash" ...)
Rules: NIL
Time:  0.01 seconds (prove: 0.00, print: 0.00, other: 0.01)
 "/home/jonathan/Desktop/Research/v6/acl2-sources/books/misc/bash.lisp"
ACL2 !>
TTAG NOTE (for included book): Adding ttag :ACL2S-TIMEOUT from book /home/jonathan/Desktop/Resea
rch/v6/acl2-sources/books/countereg-gen/with-timeout.

Summary
Form:  ( INCLUDE-BOOK "countereg-gen/top" ...)
Rules: NIL
Time:  1.91 seconds (prove: 0.00, print: 0.00, other: 1.91)
 "/home/jonathan/Desktop/Research/v6/acl2-sources/books/countereg-gen/top.lisp"
ACL2 !>
```

Figure 1: Initial screen of ACL2(ml).

3.2 Advancing in a proof script using ACL2(ml)

You can evaluate ACL2 expressions in ACL2(ml) putting the cursor before the expression and then pressing C-c C-t – this will send the expression to ACL2. In addition, you can evaluate all the expressions up to a concrete point using C-c C-u. The list of all ACL2(ml) shortcuts and functions is given in Table 1.

For instance, go to the end of the example.lisp file and evaluate the expressions up to that point using C-C C-u. Now, let us explain the functionality of ACL2(ml).

3.3 Clustering

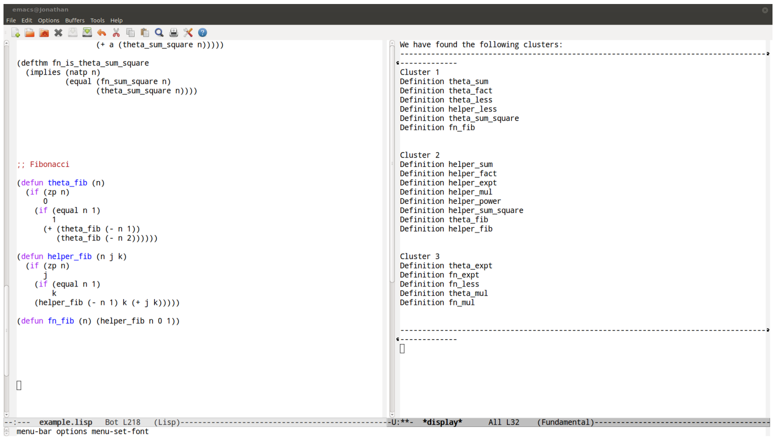
The first functionality allows the user to show the definitions and theorems that are similar using a machine-learning technique called clustering. Clustering groups objects that are correlated. To use this functionality, press the C button of the menu bar or use C-c C-c. Emacs will ask you if you want to cluster definitions or theorems.

In the case that you select to cluster definitions (d option), you can cluster only the definitions of the current library, use also the libraries that you have exported (see Section 4.3), libraries that you have selected in the ACL2(ml) menu (see Section 4.4), libraries that you have loaded using the `include` command of ACL2, or directly cluster using the whole ACL2 library (this last option is a bit slower than the rest). In this case, we select to cluster only the definitions of the current library using `c`. After a few seconds, a new buffer called `*display*` appears in Emacs, see Figure 2.

ACL2(ml) has found three clusters (we will see how to obtain a different

Function	Shortcut	Functionality
acl2ml-start	-	Initialises ACL2(ml).
acl2ml-evaluate-next-event	C-C C-t	Evaluate next event.
acl2ml-evaluate-upto-here	C-C C-u	Evaluate expressions from the beginning of the buffer up to the current point.
aclml-clusters	C-c C-c	Show clusters of similar definitions or theorems
acl2ml-show-similarities	C-c C-s	Show similar definitions or theorems.
acl2ml-obtain-guards	C-c C-g	Compute the guards for a theorem.
acl2ml-granularity	-	Change the granularity.
acl2ml-algorithm	-	Change the clustering algorithm.
acl2ml-whysimilar	-	Change the option to show the correlation of features.
acl2ml-export-library	C-c C-e	Export the library for further use.

Table 1: Functions and shortcuts available in ACL2(ml).

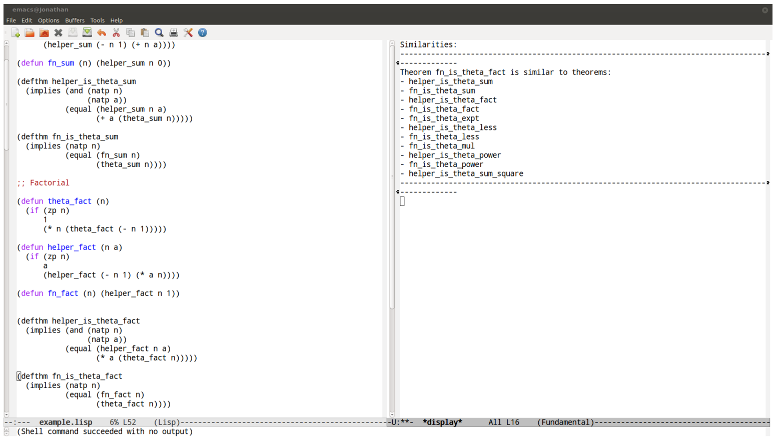


```
                  (+ a (theta_sum_square n)))))

(defthm fn_is_theta_sum_square
  (implies (natp n)
           (equal (fn_sum_square n)
                  (theta_sum_square n))))




;; Fibonacci

(defun theta_fib (n)
  (if (zp n)
      0
    (if (equal n 1)
        1
      (+ (theta_fib (- n 1))
         (theta_fib (- n 2))))))

(defun helper_fib (n j k)
  (if (zp n)
      j
    (if (equal n 1)
        k
    (helper_fib (- n 1) k (+ j k)))))

(defun fn_fib (n) (helper_fib n 0 1))
```

```
We have found the following clusters:
-----------------------------------------------------------------------------
-------------
Cluster 1
Definition theta_sum
Definition theta_fact
Definition theta_less
Definition helper_less
Definition theta_sum_square
Definition fn_fib


Cluster 2
Definition helper_sum
Definition helper_fact
Definition helper_expt
Definition helper_mul
Definition helper_power
Definition helper_sum_square
Definition theta_fib
Definition helper_fib


Cluster 3
Definition theta_expt
Definition fn_expt
Definition fn_less
Definition theta_mul
Definition fn_mul


-----------------------------------------------------------------------------
-------------
```

--:--- example.lisp   Bot L218   (Lisp)
-U:**- *display*   All L32   (Fundamental)
menu-bar options menu-set-font

Figure 2: Clusters for the example.lisp library. The Proof General window has been split into two windows positioned side by side: the left one keeps the current script, and the right one shows the families of similar lemmas.



```
      (helper_sum (- n 1) (+ n a))))

(defun fn_sum (n) (helper_sum n 0))

(defthm helper_is_theta_sum
  (implies (and (natp n)
                (natp a))
           (equal (helper_sum n a)
                  (+ a (theta_sum n)))))

(defthm fn_is_theta_sum
  (implies (natp n)
           (equal (fn_sum n)
                  (theta_sum n))))

;; Factorial

(defun theta_fact (n)
  (if (zp n)
      1
      (* n (theta_fact (- n 1)))))

(defun helper_fact (n a)
  (if (zp n)
      a
      (helper_fact (- n 1) (* a n))))

(defun fn_fact (n) (helper_fact n 1))


(defthm helper_is_theta_fact
  (implies (and (natp n)
                (natp a))
           (equal (helper_fact n a)
                  (* a (theta_fact n)))))

(defthm fn_is_theta_fact
  (implies (natp n)
           (equal (fn_fact n)
                  (theta_fact n))))
--:---  example.lisp    6% L52    (Lisp)----------------------------------------
(Shell command succeeded with no output)
```

```
Similarities:
---------------------------------------------------------------------------------
-------------
Theorem fn_is_theta_fact is similar to theorems:
- helper_is_theta_sum
- fn_is_theta_sum
- helper_is_theta_fact
- fn_is_theta_fact
- fn_is_theta_expt
- helper_is_theta_less
- fn_is_theta_less
- fn_is_theta_mul
- helper_is_theta_power
- fn_is_theta_power
- helper_is_theta_sum_square
---------------------------------------------------------------------------------
-------------

-U:**-  *display*    All L16    (Fundamental)----------------------------------------
```

Figure 3: Lemmas that are similar to `fn_is_theta_fact`.

number of clusters in Section 4.2). If we focus on cluster 2, we can see that all the auxiliary functions that are used to define tail-recursive functions are grouped in the same cluster.

Analogously for theorems, press the C button and select to cluster theorems (t option). Again, after a few seconds, the `*display*` window shows the groups of theorems.

3.4 Similarities

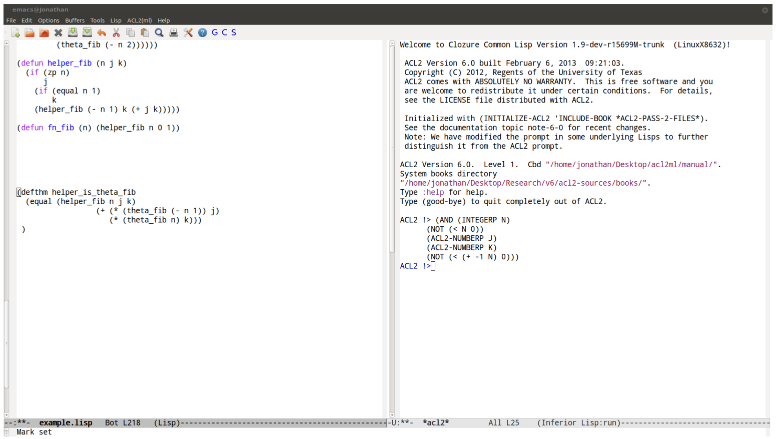
The second functionality included in ACL2(ml) allows to show clusters of similar definitions or theorems related to a concrete definition or theorem. For instance, search the lemma `fn_is_theta_fact`, put the cursor at the beginning of this lemma and press the S button or C-C C-s. After a few seconds, the Emacs buffer `*display*` shows the lemmas that are similar, cf. Figure 4.

3.5 Generating preconditions of theorems using guards

The last functionality generates preconditions of theorems based on the guards associated with the functions included in a theorem. Go to the end of the file, and include the following incomplete theorem:

```
(defthm helper_is_theta_fib
  (equal (helper_fib n j k)
                  (+ (* (theta_fib (- n 1)) j)
                     (* (theta_fib n) k)))
 )
```

This lemma states the equivalence between the recursive function to compute Fibonacci numbers and the auxiliary function of the tail-recursive version. In



```
        (theta_fib (- n 2))))))

(defun helper_fib (n j k)
  (if (zp n)
      j
    (if (equal n 1)
        k
    (helper_fib (- n 1) k (+ j k)))))

(defun fn_fib (n) (helper_fib n 0 1))




(defthm helper_is_theta_fib
  (equal (helper_fib n j k)
                 (+ (* (theta_fib (- n 1)) j)
                    (* (theta_fib n) k)))
 )
```

```
Welcome to Clozure Common Lisp Version 1.9-dev-r15699M-trunk  (LinuxX8632)!

 ACL2 Version 6.0 built February 6, 2013  09:21:03.
 Copyright (C) 2012, Regents of the University of Texas
 ACL2 comes with ABSOLUTELY NO WARRANTY.  This is free software and you
 are welcome to redistribute it under certain conditions.  For details,
 see the LICENSE file distributed with ACL2.

 Initialized with (INITIALIZE-ACL2 'INCLUDE-BOOK *ACL2-PASS-2-FILES*).
 See the documentation topic note-6-0 for recent changes.
 Note: We have modified the prompt in some underlying Lisps to further
 distinguish it from the ACL2 prompt.

ACL2 Version 6.0.  Level 1.  Cbd "/home/jonathan/Desktop/acl2ml/manual/".
System books directory
"/home/jonathan/Desktop/Research/v6/acl2-sources/books/".
Type :help for help.
Type (good-bye) to quit completely out of ACL2.

ACL2 !> (AND (INTEGERP N)
      (NOT (< N 0))
      (ACL2-NUMBERP J)
      (ACL2-NUMBERP K)
      (NOT (< (+ -1 N) 0)))
ACL2 !>
```

--:**- example.lisp   Bot L218   (Lisp)--
-U:**- *acl2*   All L25   (Inferior Lisp:run)--------------------------------
Mark set

Figure 4: Guards for `helper_is_theta_fib`.

this lemma, the preconditions are missed, we can ask to ACL2(ml) what are the guards for the functions of this theorem, this guards can be used as precondition for the theorem. To this aim, put the cursor at the beginning of the lemma and press the G button or C-c C-g. The result is shown in the `*acl2*` buffer, cf. Figure 4.

If we use these conditions in the theorem:

```
(defthm helper_is_theta_fib
  (implies (AND (INTEGERP N)
      (NOT (< N 0))
      (ACL2-NUMBERP J)
      (ACL2-NUMBERP K)
      (NOT (< (+ -1 N) 0)))
  (equal (helper_fib n j k)
                 (+ (* (theta_fib (- n 1)) j)
                    (* (theta_fib n) k)))
 ))
```

ACL2 can complete the proof of the theorem.

4 Configuration of ACL2(ml)

ACL2(ml) provides several options that can be configured from the ACL2(ml) and also from the Emacs shell.

4.1 Changing the clustering algorithm

The user can select different algorithms to obtain similar lemmas. The algorithms which are available are: K-means, EM and FarthestFirst, see Figure 5.

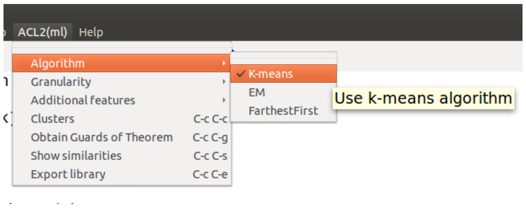


Figure 5: The Algorithms submenu.

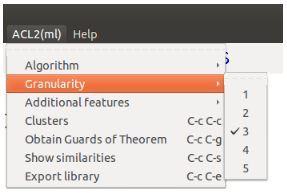


Figure 6: ACL2(ml) granularity menu.

(In our experiments, K-means usually provides the most accurate results). The user can also change the algorithm using M-x acl2ml-algorithm, and selecting there the concrete algorithm.

4.2 Granularity menu

This option allows the user to select the granularity of the families of similar lemmas, by selecting a value between 1 and 5, where 1 stands for a low granularity (producing big and general families of similar lemmas) and 5 stands for a high granularity (producing small and precise families of similar lemmas). The user can configure this option from the ACL2(ml) menu (see Figure 4.2) or using M-x acl2ml-granularity.

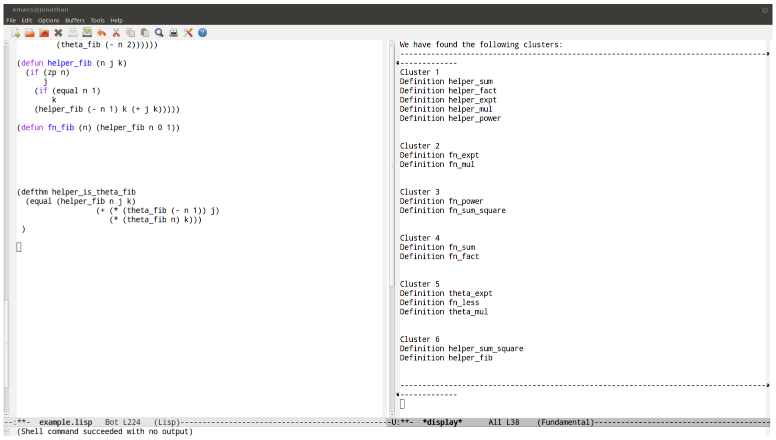
Figure 7 shows the similar definitions of our current file. As we can see, the definitions of each cluster have a more clear correlation among them that in the case of the cluster of Figure 2. Analogously for the theorems that are similar to `fn_is_theta_fact` cf. Figure 8 and 4.

4.3 Export Library

Using the Export library option of the ACl2(ml) menu (or C-c C-e), the user can export the library for further use.

4.4 Available libraries for clustering

This option allows the user to find families of similar lemmas across several libraries previously exported.



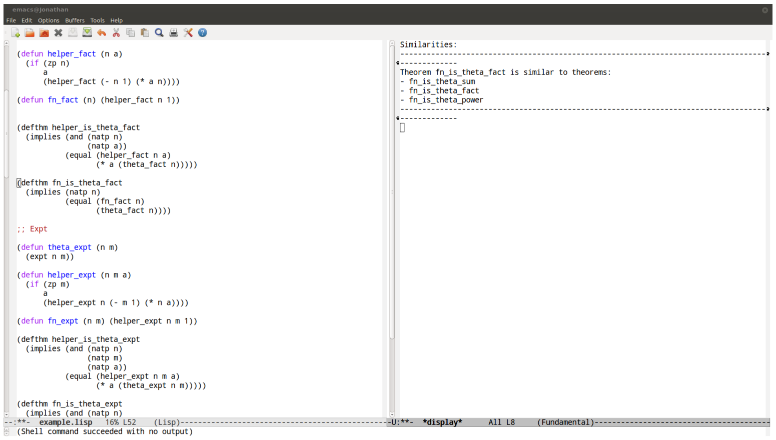
```
        (theta_fib (- n 2))))))

(defun helper_fib (n j k)
  (if (zp n)
      j
    (if (equal n 1)
        k
    (helper_fib (- n 1) k (+ j k)))))

(defun fn_fib (n) (helper_fib n 0 1))




(defthm helper_is_theta_fib
  (equal (helper_fib n j k)
                  (+ (* (theta_fib (- n 1)) j)
                     (* (theta_fib n) k)))
 )
```

```
--:**-  example.lisp   Bot L224   (Lisp)----------------------------------------
(Shell command succeeded with no output)
```

```
We have found the following clusters:
-------------------------------------------------------------------------------
-------------
Cluster 1
Definition helper_sum
Definition helper_fact
Definition helper_expt
Definition helper_mul
Definition helper_power


Cluster 2
Definition fn_expt
Definition fn_mul


Cluster 3
Definition fn_power
Definition fn_sum_square


Cluster 4
Definition fn_sum
Definition fn_fact


Cluster 5
Definition theta_expt
Definition fn_less
Definition theta_mul


Cluster 6
Definition helper_sum_square
Definition helper_fib


-------------------------------------------------------------------------------
-------------
```

```
-U:**-  *display*     All L38    (Fundamental)----------------------------------------
```

Figure 7: Clusters for the example.lisp library using granularity 5.



```

(defun helper_fact (n a)
  (if (zp n)
      a
      (helper_fact (- n 1) (* a n))))

(defun fn_fact (n) (helper_fact n 1))


(defthm helper_is_theta_fact
  (implies (and (natp n)
                (natp a))
           (equal (helper_fact n a)
                  (* a (theta_fact n)))))

(defthm fn_is_theta_fact
  (implies (natp n)
           (equal (fn_fact n)
                  (theta_fact n))))

;; Expt

(defun theta_expt (n m)
  (expt n m))

(defun helper_expt (n m a)
  (if (zp m)
      a
      (helper_expt n (- m 1) (* n a))))

(defun fn_expt (n m) (helper_expt n m 1))

(defthm helper_is_theta_expt
  (implies (and (natp n)
                (natp m)
                (natp a))
           (equal (helper_expt n m a)
                  (* a (theta_expt n m)))))

(defthm fn_is_theta_expt
  (implies (and (natp n)
```

```
--:**-  example.lisp   16% L52   (Lisp)----------------------------------------
(Shell command succeeded with no output)
```

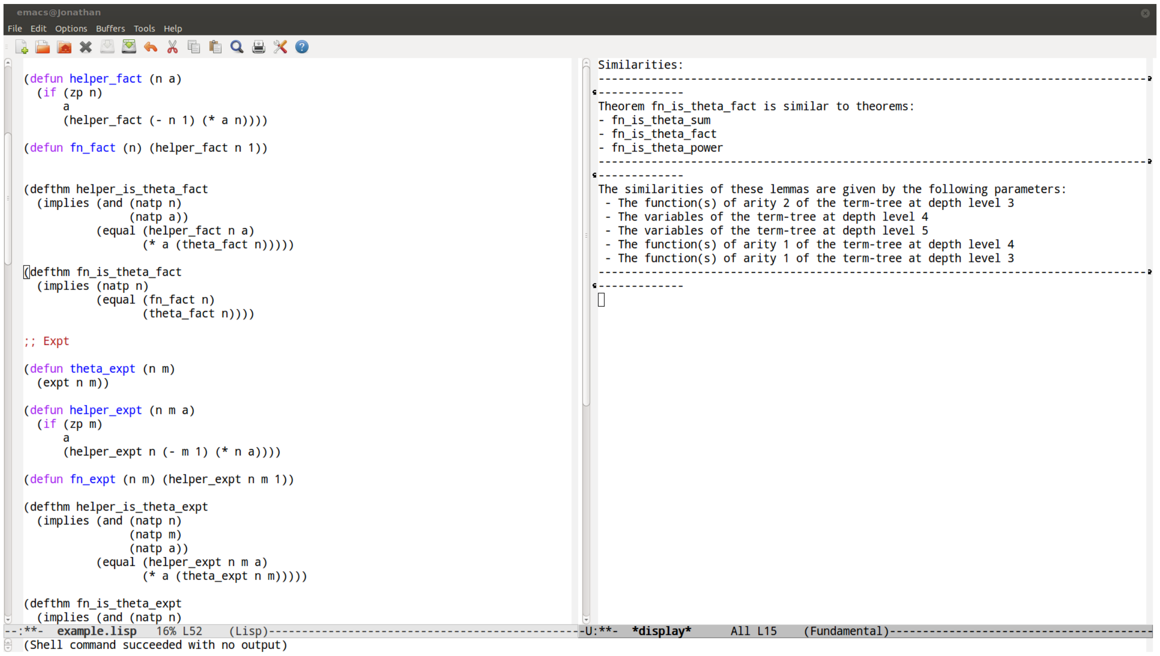
```
Similarities:
-------------------------------------------------------------------------------
-------------
Theorem fn_is_theta_fact is similar to theorems:
- fn_is_theta_sum
- fn_is_theta_fact
- fn_is_theta_power
-------------------------------------------------------------------------------
-------------
```

```
-U:**-  *display*     All L8    (Fundamental)----------------------------------------
```

Figure 8: Lemmas that are similar to `fn_is_theta_fact` using granularity 5.

4.4.1 Explain cluster similarities

If this option is activated, ACL2(ml) shows the reason because the different lemmas are grouped in the same cluster, see Figure 9.


emacs@Jonathan
File Edit Options Buffers Tools Help
--:**- example.lisp   16% L52   (Lisp)
-U:**- *display*   All L15   (Fundamental)

```lisp
(defun helper_fact (n a)
  (if (zp n)
      a
      (helper_fact (- n 1) (* a n))))

(defun fn_fact (n) (helper_fact n 1))


(defthm helper_is_theta_fact
  (implies (and (natp n)
                (natp a))
           (equal (helper_fact n a)
                  (* a (theta_fact n)))))

(defthm fn_is_theta_fact
  (implies (natp n)
           (equal (fn_fact n)
                  (theta_fact n))))

;; Expt

(defun theta_expt (n m)
  (expt n m))

(defun helper_expt (n m a)
  (if (zp m)
      a
      (helper_expt n (- m 1) (* n a))))

(defun fn_expt (n m) (helper_expt n m 1))

(defthm helper_is_theta_expt
  (implies (and (natp n)
                (natp m)
                (natp a))
           (equal (helper_expt n m a)
                  (* a (theta_expt n m)))))

(defthm fn_is_theta_expt
  (implies (and (natp n)
```

```
(Shell command succeeded with no output)
```

```
Similarities:
--------------------------------------------------------------------------------
-------------
Theorem fn_is_theta_fact is similar to theorems:
- fn_is_theta_sum
- fn_is_theta_fact
- fn_is_theta_power
--------------------------------------------------------------------------------
-------------
The similarities of these lemmas are given by the following parameters:
 - The function(s) of arity 2 of the term-tree at depth level 3
 - The variables of the term-tree at depth level 4
 - The variables of the term-tree at depth level 5
 - The function(s) of arity 1 of the term-tree at depth level 4
 - The function(s) of arity 1 of the term-tree at depth level 3
--------------------------------------------------------------------------------
-------------
```

Figure 9: Features that are used to group lemmas in a cluster.

References

[1] J. Heras, E. Komendantskaya, M. Johansson, and E. Maclean. Proof-Pattern Recognition and Lemma Discovery in ACL2. In *19th Logic for Programming Artificial Intelligence and Reasoning (LPAR-19)*, volume 8312 of *Lecture Notes in Computer Science*, pages 389–406, 2013.