ACL2(ml): Machine Learning for ACL2

J. Heras and E. Komendantskaya.
http://staff.computing.dundee.ac.uk/jheras/acl2ml/

January 20, 2014

Abstract

This manual describes ACL2(ml), a machine-learning extension to the Emacs interface for ACL2. This manual is a supplementary document to the paper "Proof-Pattern Recognition and Lemma Discovery in ACL2" [1] presented in LPAR-19, and mainly describe how to install and use ACL2(ml). The description of the underlying methods implemented in ACL2(ml) can be found in [1].

Contents

1 Differences from LPAR version

The current version of ACL2(ml) is a bit different to the one presented in [1]:

- ACL2(ml) can be used to find similar definitions and lemmas for any ACL2 development and stage of the proof.
- The symbolic lemma generation from LPAR-19 paper does not generalize easily: it currently gives nice suggestions in some cases, but no suggestions in other cases [no bugs/errors though]. Nevertheless, we now re-implemented it in Lisp (it was externally implemented in ML before).
- We have designed a weaker symbolic procedure to complement machine learning – and that is generation of lemma pre-conditions, using ACL2 guards.

2 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`.

3 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).

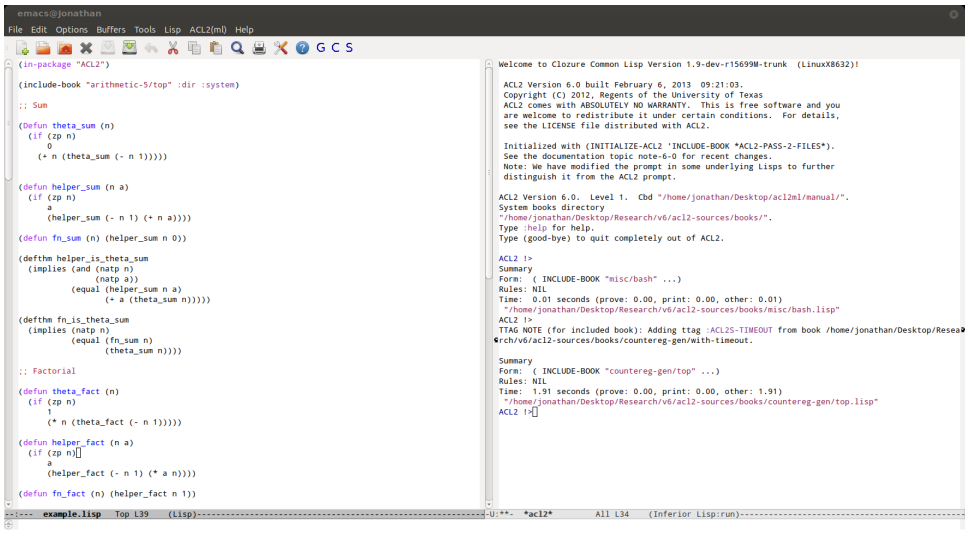

Figure 1: Initial screen of ACL2(ml).

4 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.

4.1 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.

4.2 Advancing in a proof script using ACL2(ml)

You can evaluate ACL2 expressions 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

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).

ACL2(ml).

4.3 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 5.3), libraries that you have selected in the ACL2(ml) menu (see Section 5.4), 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 number of clusters in Section 5.2). 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.

4.4 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

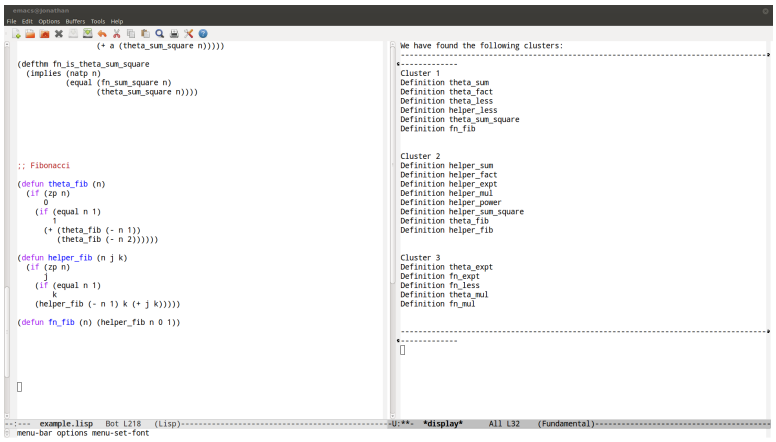

Figure 2: Clusters for the example.lisp library. The Proof General window has been split into two windows positioned side by side: the left one keeps the current script, and the right one shows the families of similar lemmas.

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.

4.5 Generating preconditions of theorems using guards

This section describes a new functionality of ACL2(ml) that was not included in [1]. Guards (`http://www.cs.utexas.edu/~moore/acl2/current/manual/index.html?topic=ACL2____GUARD`) are a mechanism to restrict the domain of a function. Guards can be used to generate preconditions to lemma statements on the basis of the functions included in a theorem and this is the last functionality that we have included in ACL2(ml). To use this mechanism the user does not need to define the guards of his functions, but ACL2(ml) computes some by-default guards on the basis of the guards of the functions that are used in the new definition.

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 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

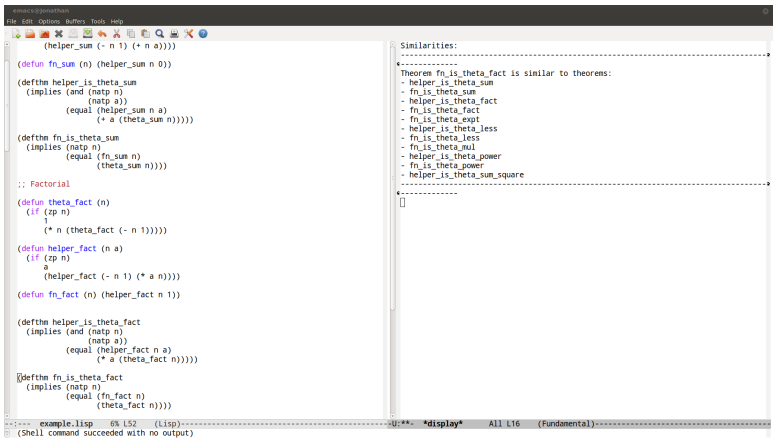

Figure 3: Lemmas that are similar to `fn_is_theta_fact`.

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.

5 Configuration of ACL2(ml)

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

5.1 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. (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.

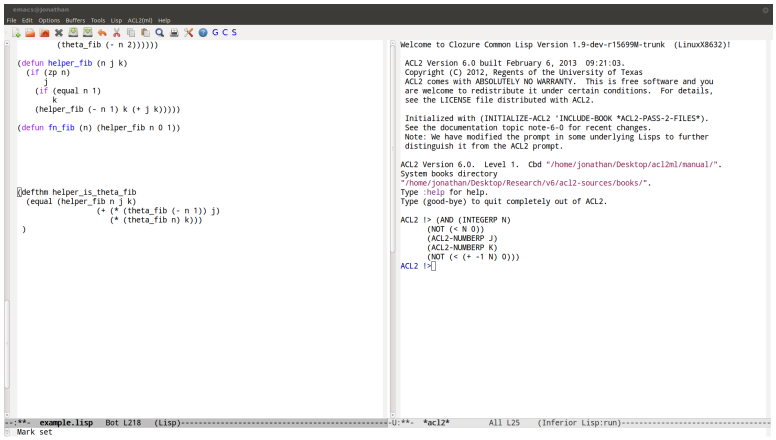

Figure 4: Guards for `helper_is_theta_fib`.

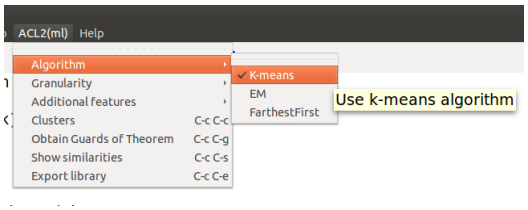

Figure 5: The Algorithms submenu.

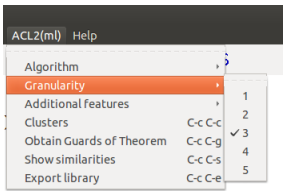

Figure 6: ACL2(ml) granularity menu.

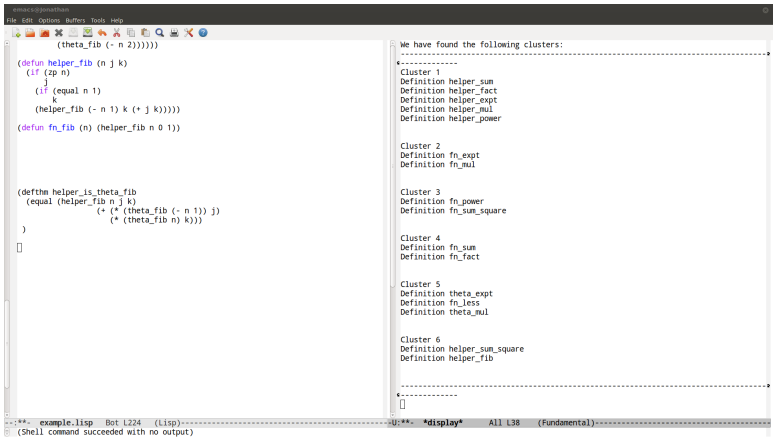

Figure 7: Clusters for the example.lisp library using granularity 5.

5.2 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 5.2) 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.

5.3 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.

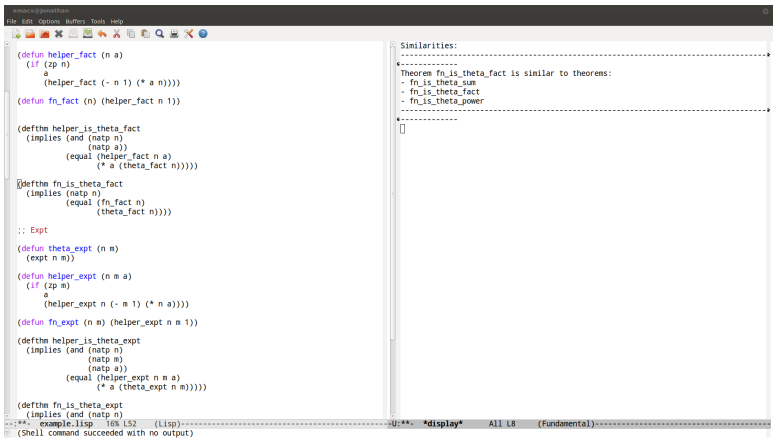

Figure 8: Lemmas that are similar to `fn_is_theta_fact` using granularity 5.

5.4 Available libraries for clustering

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

5.4.1 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.

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.

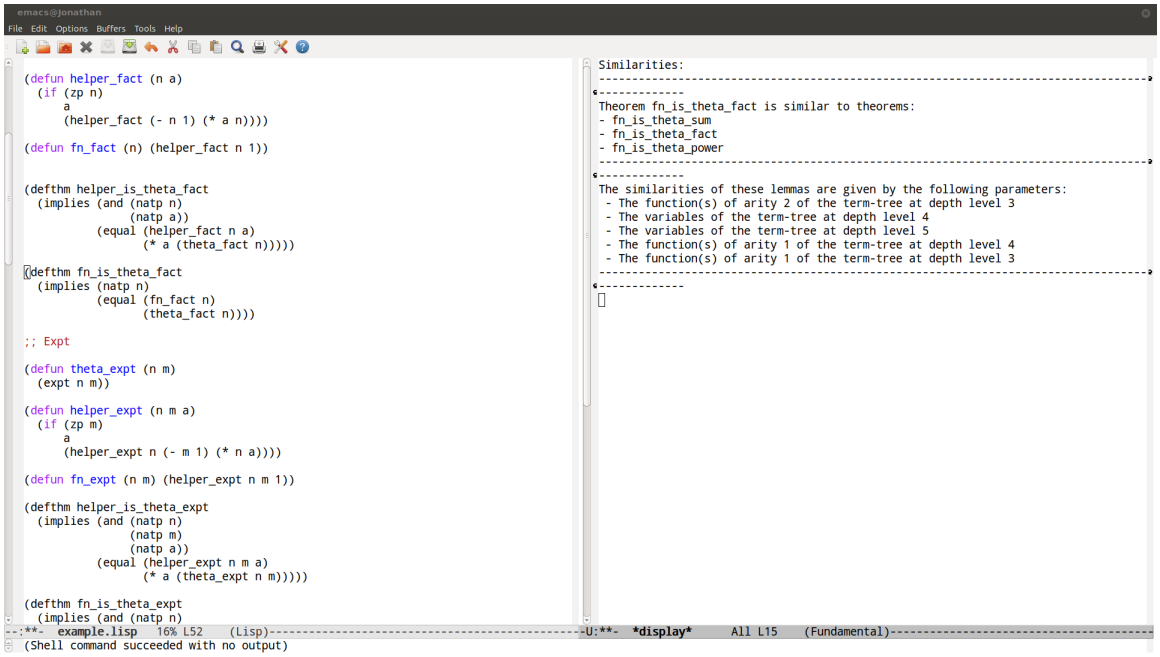

Figure 9: Features that are used to group lemmas in a cluster.