ML4PG: Machine learning for Proof General

Jónathan Heras and Ekaterina Komendantskaya
{jonathanheras,katya}@computing.dundee.ac.uk

March 12, 2014

Contents

1 Requirements

Before installing ML4PG, you need to download and install the following software.

- Proof General.
- Coq.
- SSReflect (optional).
- graphviz (`sudo apt-get install graphviz`).

2 Installation

Before using ML4PG, it is necessary to configure some variables. First of all, open your *.emacs* file. This file is usually located in */home/user/.emacs*. At the end of the file include the line:

```
(load-file "ML4PG-location/ml4pg.el")
```

where `ML4PG-location` must be changed with the path to the folder where you have downloaded ML4PG.

Now, go to the folder where you have downloaded ML4PG and open the file *ml4pg.el*. The three first lines of this file are three constants.

- *home-dir*: you must replace the current path assigned to this constant with the path where you have downloaded ML4PG.

This finishes the installation of ML4PG.

3 Using ML4PG

To illustrate the use of ML4PG, we will use the file `ml4pg.v` which can be find in the same folder of this manual. This file contains various lemmas about natural numbers and lists.

3.1 Getting started

Open a command line and go to the folder where you have the file `ml4pg.v`. Open the file using `emacs ml4pg.v`. If you have installed everything properly, the image of Figure 1 will appear.

At this point, ML4PG asks you if you are developing your proofs using the plain Coq style or the SSReflect style, in this case we select the Coq mode (c). Once that this is done, the typical Proof General interface appears but including a new menu called *Statistics*, see Figure 2.

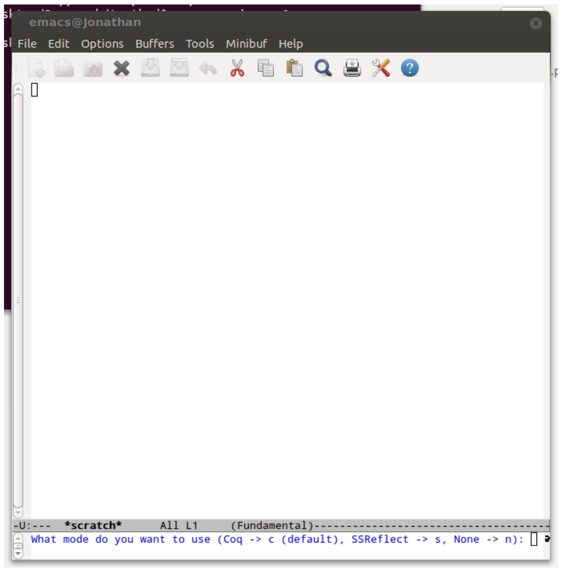
This menu contains several options to configure ML4PG. First of all, you can activate the buttons and the different libraries which can be used pressing the options: *Activate icons* and *Show Available libraries*. Once that you have done this, the interface looks like in Figure 3.

3.2 Extracting feature vectors

Feature vectors can be extracted in two different ways:

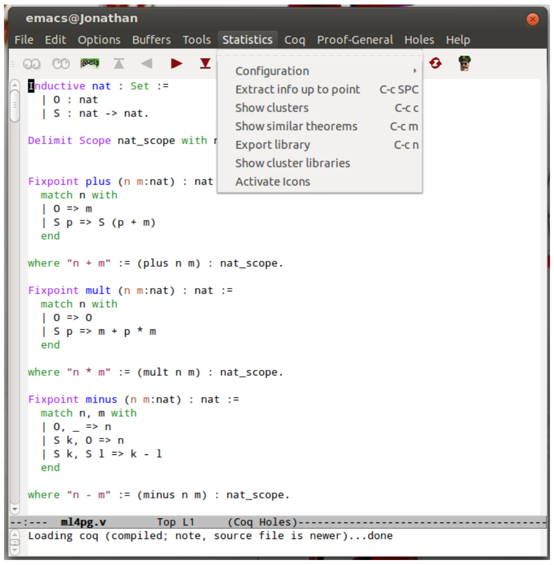
- During the development of the proofs. To this aim, you have to use the shortcut Ctrl-C Ctrl-M to process the next proof command.

- Several proofs at the same time. If you want to extract the feature vectors of several proofs, go to the last proof and use the shortcut Ctrl-C Space. You can also use the *Extract info up to point* option of the statistics menu.



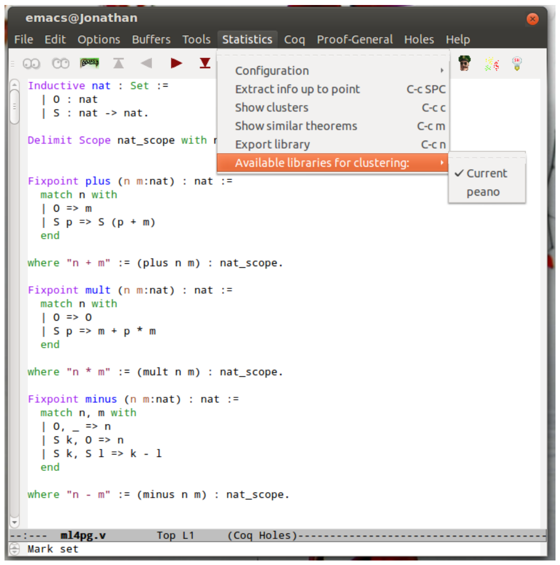
```
-U:---  *scratch*      All L1     (Fundamental)-----------------------------------
What mode do you want to use (Coq -> c (default), SSReflect -> s, None -> n):
```

Figure 1: Initial screen of ML4PG.


emacs@Jonathan
File Edit Options Buffers Tools Statistics Coq Proof-General Holes Help
Configuration
Extract info up to point C-c SPC
Show clusters C-c c
Show similar theorems C-c m
Export library C-c n
Show cluster libraries
Activate Icons

```
Inductive nat : Set :=
  | O : nat
  | S : nat -> nat.

Delimit Scope nat_scope with r


Fixpoint plus (n m:nat) : nat
  match n with
  | O => m
  | S p => S (p + m)
  end

where "n + m" := (plus n m) : nat_scope.

Fixpoint mult (n m:nat) : nat :=
  match n with
  | O => O
  | S p => m + p * m
  end

where "n * m" := (mult n m) : nat_scope.

Fixpoint minus (n m:nat) : nat :=
  match n, m with
  | O, _ => n
  | S k, O => n
  | S k, S l => k - l
  end

where "n - m" := (minus n m) : nat_scope.
--:---  ml4pg.v        Top L1     (Coq Holes)------------------------------------
Loading coq (compiled; note, source file is newer)...done
```

Figure 2: Statistics menu of ML4PG.


emacs@Jonathan
File Edit Options Buffers Tools Statistics Coq Proof-General Holes Help
Configuration
Extract info up to point C-c SPC
Show clusters C-c c
Show similar theorems C-c m
Export library C-c n
Available libraries for clustering:
✓ Current
peano

```
Inductive nat : Set :=
  | O : nat
  | S : nat -> nat.

Delimit Scope nat_scope with r

Fixpoint plus (n m:nat) : nat :=
  match n with
  | O => m
  | S p => S (p + m)
  end

where "n + m" := (plus n m) : nat_scope.

Fixpoint mult (n m:nat) : nat :=
  match n with
  | O => O
  | S p => m + p * m
  end

where "n * m" := (mult n m) : nat_scope.

Fixpoint minus (n m:nat) : nat :=
  match n, m with
  | O, _ => n
  | S k, O => n
  | S k, S l => k - l
  end

where "n - m" := (minus n m) : nat_scope.

--:---  ml4pg.v        Top L1      (Coq Holes)--------------------------------------
Mark set
```

Figure 3: ML4PG interface with all the options active.

Go to the end of `emacs ml4pg.v` file; there, you can see two unfinished proofs: `M3_3b` and `aux7_bis`. Put the cursor at the end of the proof of Lemma `andb_false_r` and use the shortcut Ctrl-C Space or the *Extract info up to point* option of the statistics menu. In this way the information associated with each proof will be extracted and you will be able to use it to obtain proof clusters (groups of similar proofs).

Now, let us explain the functionality of the options included in the Statistics menu.

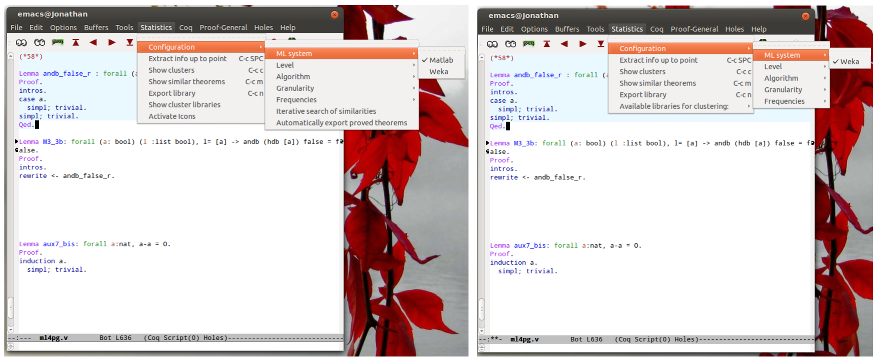
3.3 Configuration menu

The different options to configure the Machine-learning environments were detailed in [4]. All those options can be accessed from the Configuration submenu of the Statistics menu, see Figure 2.

ML sytem: This option allows the user to select the Machine Learning environment that she wants to use. If you downloaded the complete version of ML4PG you will have two options: MATLAB and Weka; otherwise, the only available option is Weka. See Figure 4.

In our examples, we will use the Weka option.

Level: The user can select three different levels to obtain proof similarities (see Figure 5):


emacs@Jonathan
File Edit Options Buffers Tools Statistics Coq Proof-General Holes Help
Configuration
Extract info up to point C-c SPC
Show clusters C-c c
Show similar theorems C-c m
Export library C-c n
Show cluster libraries
Activate Icons
ML system
Level
Algorithm
Granularity
Frequencies
Iterative search of similarities
Automatically export proved theorems
✓ Matlab
Weka
emacs@Jonathan
File Edit Options Buffers Tools Statistics Coq Proof-General Holes Help
Configuration
Extract info up to point C-c SPC
Show clusters C-c c
Show similar theorems C-c m
Export library C-c n
Available libraries for clustering:
ML system
Level
Algorithm
Granularity
Frequencies
✓ Weka

```
(*58*)

Lemma andb_false_r : forall (a
Proof.
intros.
case a.
  simpl; trivial.
simpl; trivial.
Qed.

Lemma M3_3b: forall (a: bool) (l :list bool), l= [a] -> andb (hdb [a]) false = f
alse.
Proof.
intros.
rewrite <- andb_false_r.



Lemma aux7_bis: forall a:nat, a-a = 0.
Proof.
induction a.
  simpl; trivial.

--:---  ml4pg.v       Bot L636    (Coq Script(0) Holes)----------------------------
```

```
--:**-  ml4pg.v       Bot L636    (Coq Script(0) Holes)----------------------------
```

Figure 4: **Left.** The ML system menu of the complete version of ML4PG. **Right.** The ML system menu of the ML4PG for Weka version.

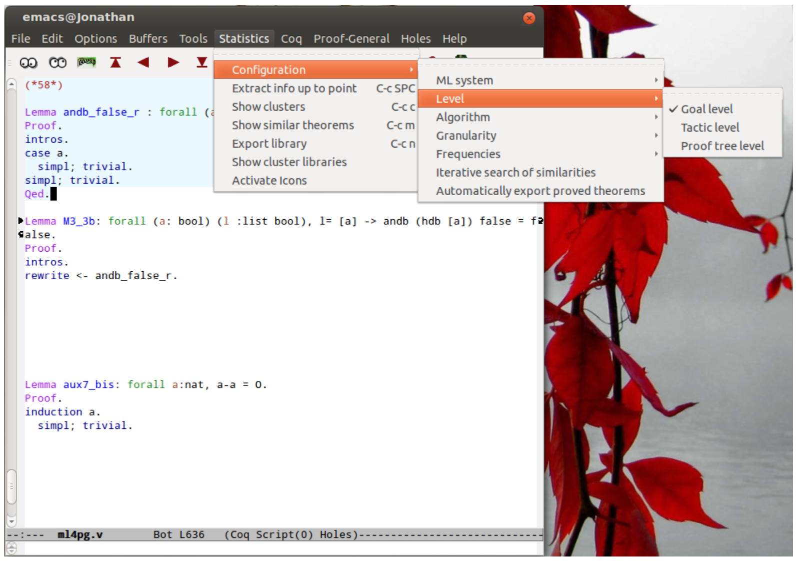
1. Goal-pattern recognition. Sequences of subgoals may show an apparent pattern in the structure of the formulas.
2. Tactic-pattern recognition. Sequences of tactics applied at every level of the proof bear some apparent patterns, as well.
3. Proof tree pattern recognition. There is the level of a proof tree – that shows relations between different proof branches and subgoals.

A detailed description of these three levels was given in [4]. By default, we select the Goal level which has shown the better results up to now.

Algorithms: The user can select different algorithms to obtain proof similarities (all of them behave similar, see [3]). There are different algorithms available depending on the selected Machine Learning system, see Figure 6.

In the case of MATLAB; there are three algorithms available: K-means and Gaussian. In the case of Weka, the algorithms which are available are: K-means, EM and FarthestFirst.

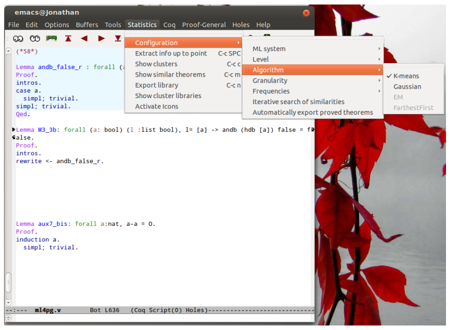
Granularity: In the machine learning literature, there exists a number of heuristics to determine this optimal number of clusters. We used them as an inspiration to formulate our own algorithm for ML4PG, tailored to the interactive proofs. It takes into consideration the size of the proof library and an auxiliary parameter – called granularity. This parameter is used to calculate the optimal number of proof clusters, the process to calculate this optimal number was described in [4]. The user decides the granularity in ML4PG menu (see Figure 7), by selecting a value between 1 and 5, where 1 stands for a low granularity (producing big and general clusters) and 5 stands for a high granularity (producing small and precise clusters).

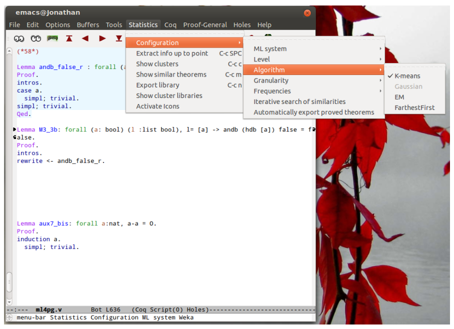

emacs@Jonathan
File Edit Options Buffers Tools Statistics Coq Proof-General Holes Help
Configuration
Extract info up to point C-c SPC
Show clusters C-c c
Show similar theorems C-c m
Export library C-c n
Show cluster libraries
Activate Icons
ML system
Level
Algorithm
Granularity
Frequencies
Iterative search of similarities
Automatically export proved theorems
✓ Goal level
Tactic level
Proof tree level

```
(*58*)

Lemma andb_false_r : forall (a
Proof.
intros.
case a.
  simpl; trivial.
simpl; trivial.
Qed.

Lemma M3_3b: forall (a: bool) (l :list bool), l= [a] -> andb (hdb [a]) false = f
alse.
Proof.
intros.
rewrite <- andb_false_r.



Lemma aux7_bis: forall a:nat, a-a = 0.
Proof.
induction a.
  simpl; trivial.
```

```
--:---  ml4pg.v        Bot L636    (Coq Script(0) Holes)-----------------------------
```

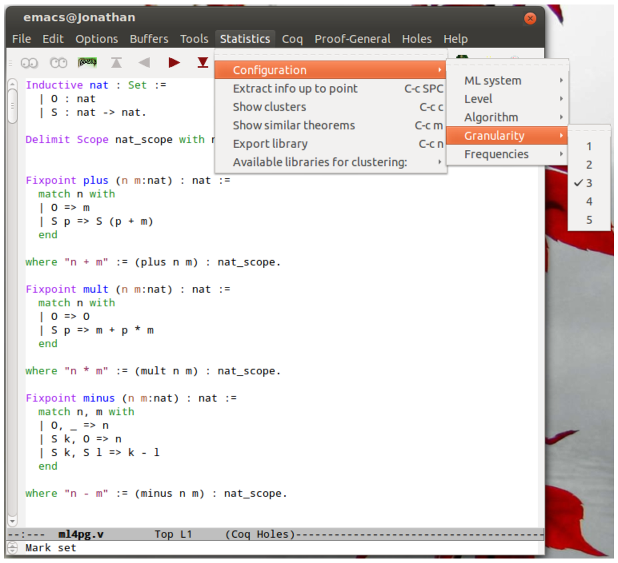
Figure 5: ML4PG levels menu.


emacs@Jonathan
File Edit Options Buffers Tools Statistics Coq Proof-General Holes Help
Configuration
Extract info up to point C-c SPC
Show clusters C-c c
Show similar theorems C-c m
Export library C-c n
Show cluster libraries
Activate Icons
ML system
Level
Algorithm
Granularity
Frequencies
Iterative search of similarities
Automatically export proved theorems
✓ K-means
Gaussian
EM
FarthestFirst


emacs@Jonathan
File Edit Options Buffers Tools Statistics Coq Proof-General Holes Help
Configuration
Extract info up to point C-c SPC
Show clusters C-c c
Show similar theorems C-c m
Export library C-c n
Show cluster libraries
Activate Icons
ML system
Level
Algorithm
Granularity
Frequencies
Iterative search of similarities
Automatically export proved theorems
✓ K-means
Gaussian
EM
FarthestFirst

```
menu-bar Statistics Configuration ML system Weka
```

Figure 6: **Left.** The ML algorithms menu with the MATLAB selection. **Right.** The ML algorithms menu with the Weka selection.


emacs@Jonathan
File Edit Options Buffers Tools Statistics Coq Proof-General Holes Help
Configuration
Extract info up to point C-c SPC
Show clusters C-c c
Show similar theorems C-c m
Export library C-c n
Available libraries for clustering:
ML system
Level
Algorithm
Granularity
Frequencies
1
2
✓ 3
4
5

```
Inductive nat : Set :=
  | O : nat
  | S : nat -> nat.

Delimit Scope nat_scope with r


Fixpoint plus (n m:nat) : nat :=
  match n with
  | O => m
  | S p => S (p + m)
  end

where "n + m" := (plus n m) : nat_scope.

Fixpoint mult (n m:nat) : nat :=
  match n with
  | O => O
  | S p => m + p * m
  end

where "n * m" := (mult n m) : nat_scope.

Fixpoint minus (n m:nat) : nat :=
  match n, m with
  | O, _ => n
  | S k, O => n
  | S k, S l => k - l
  end

where "n - m" := (minus n m) : nat_scope.

--:---  ml4pg.v        Top L1     (Coq Holes)------------------------------------
Mark set
```

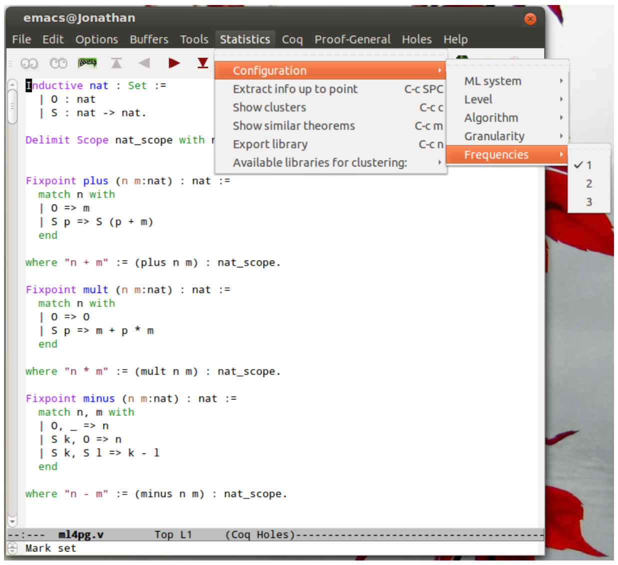
Figure 7: ML4PG granularity menu.

Frequencies: Clustering techniques divide data into n groups of similar objects (called clusters), where the value of n is a "learning" parameter provided by the user together with other inputs to the clustering algorithms. Increasing the value of n means that the algorithm will try to separate objects into more classes, and, as a consequence, each cluster will contain examples with higher correlation. The frequencies of clusters can serve for analysis of their reliability. Results of one run of a clustering algorithm may differ from another, even on the same data set. This is due to the fact that clustering algorithms randomly choose examples to start from, and then, form clusters relative to those examples. However, it may happen that certain clusters are found repeatedly – and frequently – in different runs; then, we can use these frequencies to determine the reliable clusters. The frequencies can be determined using the threshold presented in Figure 8, a detailed description of this parameter was given in [4].

3.4 Show clusters

The option *Show Clusters* of the Statistics menu shows clusters when a library is clustered irrespective of the current proof goal. An example using the `ml4pg.v` library with the options:

- ML system: Weka,
- Algorithm: K-means,
- Level: Goal-level,


emacs@Jonathan
File Edit Options Buffers Tools Statistics Coq Proof-General Holes Help
Configuration
Extract info up to point C-c SPC
Show clusters C-c c
Show similar theorems C-c m
Export library C-c n
Available libraries for clustering:
ML system
Level
Algorithm
Granularity
Frequencies
✓ 1
2
3

```
Inductive nat : Set :=
  | O : nat
  | S : nat -> nat.

Delimit Scope nat_scope with r

Fixpoint plus (n m:nat) : nat :=
  match n with
  | O => m
  | S p => S (p + m)
  end

where "n + m" := (plus n m) : nat_scope.

Fixpoint mult (n m:nat) : nat :=
  match n with
  | O => O
  | S p => m + p * m
  end

where "n * m" := (mult n m) : nat_scope.

Fixpoint minus (n m:nat) : nat :=
  match n, m with
  | O, _ => n
  | S k, O => n
  | S k, S l => k - l
  end

where "n - m" := (minus n m) : nat_scope.

--:---  ml4pg.v        Top L1      (Coq Holes)--------------------------------------
Mark set
```

Figure 8: ML4PG frequencies menu.

- Granularity: 3,
- Frequencies: 1.

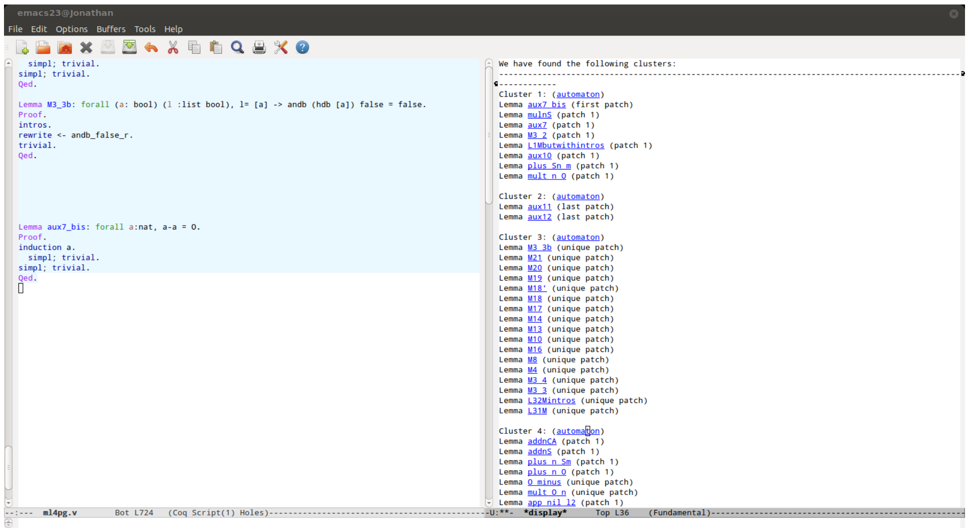
is shown in Figure 9.

ML4PG can cluster families of similar definitions, lemma statements and proofs. These three functionalities can be called using the clustering button of the Proof General toolbar.

3.5 Show similar theorems

The example above shows one mode of working with ML4PG: that is, when a library is clustered irrespective of the current proof goal. However, it may be useful to use this technology to aid the interactive proof development. In which case, we can cluster libraries relative to a few initial proof steps for the current proof goal. An example using the `ml4pg.v` library with the options:

- ML system: Weka,
- Algorithm: FarthestFirst,
- Level: Goal-level,
- Granularity: 4,
- Frequencies: 2.

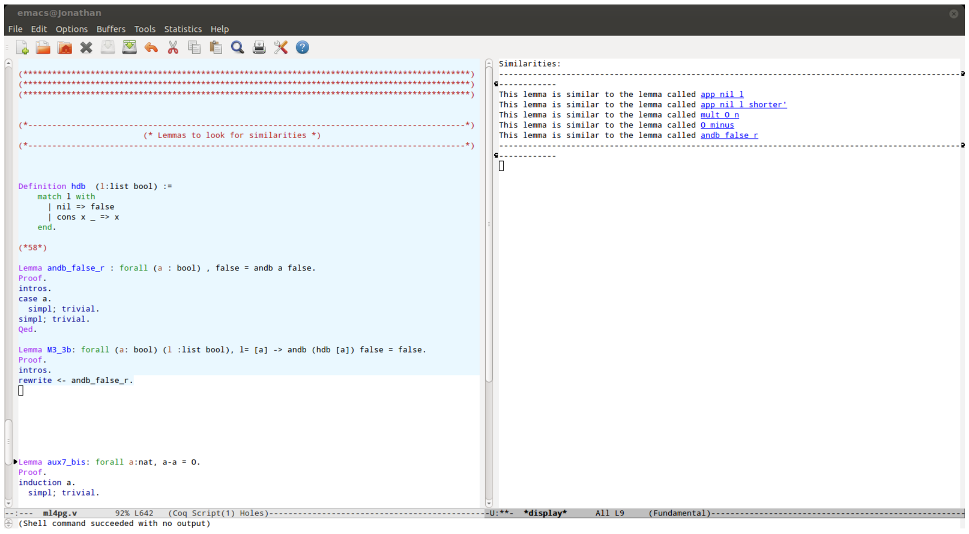


```
  simpl; trivial.
simpl; trivial.
Qed.

Lemma M3_3b: forall (a: bool) (l :list bool), l= [a] -> andb (hdb [a]) false = false.
Proof.
intros.
rewrite <- andb_false_r.
trivial.
Qed.




Lemma aux7_bis: forall a:nat, a-a = 0.
Proof.
induction a.
  simpl; trivial.
simpl; trivial.
Qed.
```

```
We have found the following clusters:
------------------------------------------------------------------------------------------
------------
Cluster 1: (automaton)
Lemma aux7 bis (first patch)
Lemma mulnS (patch 1)
Lemma aux7 (patch 1)
Lemma M3 2 (patch 1)
Lemma L1Mbutwithintros (patch 1)
Lemma aux10 (patch 1)
Lemma plus Sn m (patch 1)
Lemma mult n O (patch 1)

Cluster 2: (automaton)
Lemma aux11 (last patch)
Lemma aux12 (last patch)

Cluster 3: (automaton)
Lemma M3 3b (unique patch)
Lemma M21 (unique patch)
Lemma M20 (unique patch)
Lemma M19 (unique patch)
Lemma M18' (unique patch)
Lemma M18 (unique patch)
Lemma M17 (unique patch)
Lemma M14 (unique patch)
Lemma M13 (unique patch)
Lemma M10 (unique patch)
Lemma M16 (unique patch)
Lemma M8 (unique patch)
Lemma M4 (unique patch)
Lemma M3 4 (unique patch)
Lemma M3 3 (unique patch)
Lemma L32Mintros (unique patch)
Lemma L31M (unique patch)

Cluster 4: (automaton)
Lemma addnCA (patch 1)
Lemma addnS (patch 1)
Lemma plus n Sm (patch 1)
Lemma plus n O (patch 1)
Lemma O minus (unique patch)
Lemma mult O n (unique patch)
Lemma app nil l2 (patch 1)
```

```
--:---  ml4pg.v       Bot L724    (Coq Script(1) Holes)---------------------------------------------------------------------------------U:**-  *display*     Top L36     (Fundamental)-------------------------------------------------------
```

Figure 9: Clusters for the ml4pg library. The Proof General window has been split into two windows positioned side by side: the left one keeps the current proof script, and the right one shows the clusters. If the user clicks on the name of a theorem showed in the right screen, such a window is split horizontally and a brief description of the selected theorem is shown.

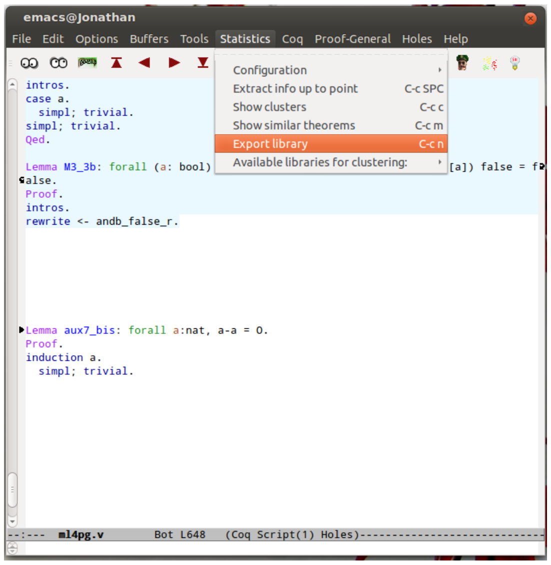


```
(**************************************************************************************)
(**************************************************************************************)
(**************************************************************************************)


(*------------------------------------------------------------------------------------*)
                        (* Lemmas to look for similarities *)
(*------------------------------------------------------------------------------------*)



Definition hdb  (l:list bool) :=
    match l with
      | nil => false
      | cons x _ => x
    end.

(*58*)

Lemma andb_false_r : forall (a : bool) , false = andb a false.
Proof.
intros.
case a.
  simpl; trivial.
simpl; trivial.
Qed.

Lemma M3_3b: forall (a: bool) (l :list bool), l= [a] -> andb (hdb [a]) false = false.
Proof.
intros.
rewrite <- andb_false_r.





Lemma aux7_bis: forall a:nat, a-a = 0.
Proof.
induction a.
  simpl; trivial.
```

```
Similarities:
------------------------------------------------------------------------------------------
------------
This lemma is similar to the lemma called app nil l
This lemma is similar to the lemma called app nil l shorter'
This lemma is similar to the lemma called mult O n
This lemma is similar to the lemma called O minus
This lemma is similar to the lemma called andb false r
------------------------------------------------------------------------------------------
------------
```

```
--:---  ml4pg.v       92% L642    (Coq Script(1) Holes)---------------------------------------------------------------------------------U:**-  *display*     All L9      (Fundamental)-------------------------------------------------------
(Shell command succeeded with no output)
```

Figure 10: On the right side, several suggestions provided by ML4PG. If the user clicks on the name of one of the suggested lemmas, a brief description about it is shown.


emacs@Jonathan
File Edit Options Buffers Tools Statistics Coq Proof-General Holes Help
Configuration
Extract info up to point C-c SPC
Show clusters C-c c
Show similar theorems C-c m
Export library C-c n
Available libraries for clustering:

```
intros.
case a.
  simpl; trivial.
simpl; trivial.
Qed.

Lemma M3_3b: forall (a: bool)                    [a]) false = f
alse.
Proof.
intros.
rewrite <- andb_false_r.




Lemma aux7_bis: forall a:nat, a-a = 0.
Proof.
induction a.
  simpl; trivial.
```

```
--:---  ml4pg.v        Bot L648   (Coq Script(1) Holes)----------------------------
```

Figure 11: ML4PG export menu.

and with the few steps included about the proof of `M3_3b` is shown in Figure 10.

This functionality can also invoked using the right most button of the Proof General toolbar.

3.6 Export Library

Using the Export library option, the user can export the library for further use (see Figure 11) with the Available libraries for clustering option.

4 Experimental features

In this section, we describe some experimental features of ML4PG to visualise ML4PG's results.

4.1 Automaton for clustering

To facilitate the understanding of ML4PG output, ML4PG can display the proof pattern in the form of an automaton, showing the correlating features that made the pattern. This functionality can be accessed once the clusters are shown as explained in the previous subsection. For each cluster, an automaton button is available, see Figure 9. If the user clicks on that button, an automaton representation of the pattern is generated. For instance, using the configuration of the previous subsection, ML4PG generates the automaton of Figure 12 for the

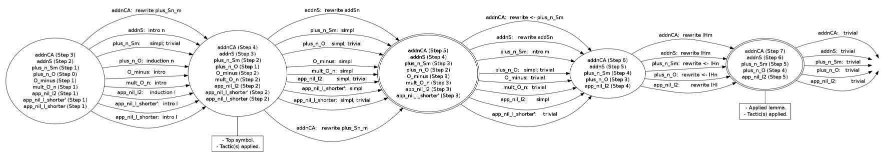


Figure 12: Automaton for the lemmas addnCA, addnS, plus_n_Sm, plus_n_O, O_minus, mult_O_n, app_nil_l2, app_nil_l_shorter', app_nil_l_shorter

cluster containing lemmas addnCA, addnS, plus_n_Sm, plus_n_O, O_minus, mult_O_n, app_nil_l2, app_nil_l_shorter', app_nil_l_shorter.

A detailed description of the automaton is given in [2, 3].

4.2 Similarity graphs

ML4PG can generate 3 similarity graphs: for definitions, lemma statements, and proofs. These features can be accessed from the "Generate Similarity Graph" submenu of the ML4PG menu. This tool can be used to visualise the different clusters of the library, see Figure 13.

The current version of the similarity graphs generate a png file and also a webpage (as further work, we plan to link the ellipses included in the generated webpage to the referred theorem). A comparison of these similarity graphs with the dependency graphs available at Coq can be seen in [1].

4.3 Visualisation of term trees of Lemma statements

ML4PG can show the term tree of lemma statement that has been already proven. This functionality can be accessed from the option "Term tree of a lemma statement" from the "Generate Similarity Graph" submenu of the ML4PG menu. For instance, the term tree of Lemma andb_false_r of the ml4pg.v library is given in Figure 14.

A discussion about these term trees can be seen in [1, 2].

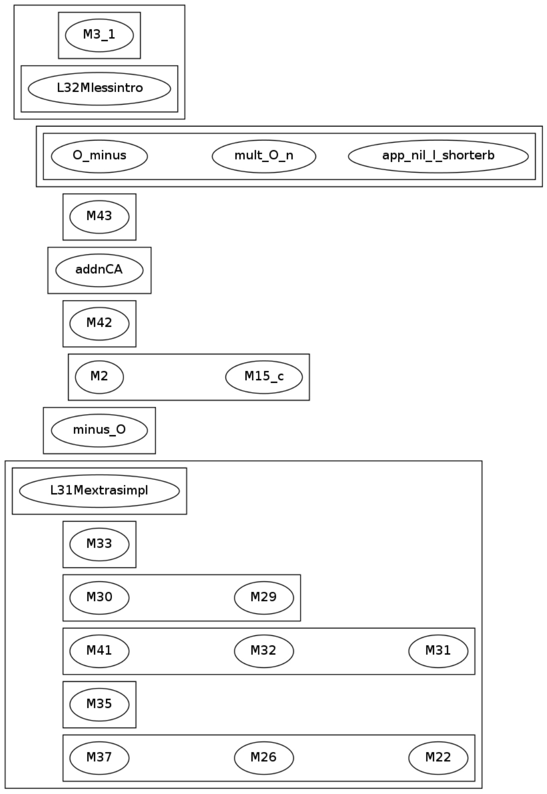


Figure 13: Visualisation of the proof-clusters of the `ml4pg.v` library. The boxes of the diagram represent the different clusters, and the ellipses inside the boxes are the lemmas of each cluster. We run twice the clustering algorithm to generate the visualisation of term-clusters: first with granularity value 3 and then with granularity value 5. If a cluster that appears with granularity value 3 is split in smaller clusters when increasing the granularity value, we represent this situation using boxes inside boxes.

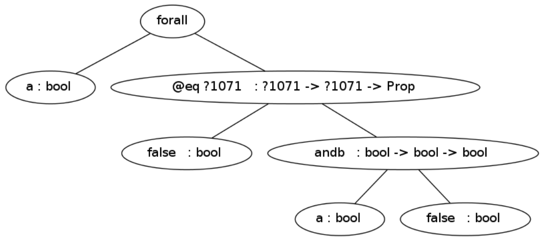

forall
a : bool
@eq ?1071 : ?1071 -> ?1071 -> Prop
false : bool
andb : bool -> bool -> bool
a : bool
false : bool

Figure 14: Visualisation of the term tree of the lemma `andb_false_r`.

References

[1] J. Heras and E. Komendantskaya. HoTT formalisation in Coq: Dependency Graphs & ML4PG, 2014. `http://arxiv.org/abs/1403.2531`.

[2] J. Heras and E. Komendantskaya. Proof Pattern Search in Coq/SSReflect, 2014. `http://arxiv.org/abs/1402.0081`.

[3] J. Heras and E. Komendantskaya. Recycling Proof Patterns in Coq: Case Studies. *Journal Mathematics in Computer Science, accepted*, 2014.

[4] E. Komendantskaya, J. Heras, and G. Grov. Machine learning in proof general: interfacing interfaces. 2012.