Automation by Analogy, in Coq

Alasdair Hill, Katya Komendantskaya

Heriot-Watt University, Scotland

20 March 2018
ML4PG interfaces with proof general to extract features of lemmas from an ITP and uses a machine learning tool such as weka to cluster them.

Feature Extraction

Feature extraction is performed to cluster lemmas on both proof terms and types

ML4PG approach to proof-clustering

We have integrated Proof General with a variety of clustering algorithms:
ML4PG approach to proof-clustering

We have integrated Proof General with a variety of clustering algorithms:

- Unsupervised machine learning technique:
ML4PG approach to proof-clustering

We have integrated Proof General with a variety of clustering algorithms:

- Unsupervised machine learning technique:

  ![Clustering Diagram]

- Engines: Matlab, Weka, Octave, R, ...
ML4PG approach to proof-clustering

We have integrated Proof General with a variety of clustering algorithms:

- **Unsupervised machine learning technique:**

- **Engines:** Matlab, Weka, Octave, R, ...
ML4PG approach to proof-clustering

We have integrated Proof General with a variety of clustering algorithms:

- Unsupervised machine learning technique:

- Engines: Matlab, Weka, Octave, R, ...

- Algorithms: K-means, Gaussian Mixture models, simple Expectation Maximisation, ...
Interaction with ML4PG:

- One interacts with Proof General as usual,
- when one cannot proceed with a proof,
- he calls ML4PG (command line or editor button),
- ML4PG informs the user of similar existing proofs/definitions.
A proof in Coq with ML4PG help
A proof in Coq with ML4PG help

```
Lemma M1_corrected : forall l : list A, l = [] -> tl (tl l) ++ nil = nil.
Proof.
  intro l.
  intro H.
  rewrite H.
  rewrite app_nil_l2.
  simpl; trivial.
  Qed.

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

Lemma M3_3b : forall (a : bool) (l : list bool), l = [a] -> andb (hdb [a]) false.
Proof.
  intro l.
  intro H.
  rewrite H.
  rewrite app_nil_l2.
  simpl; trivial.
  Qed.
```
A proof in Coq with ML4PG help

```coq
Lemma M1_corrected : forall l : list A, l = [] -> tl (tl l) ++ nil = nil.
Proof.
  intro l.
  intro H.
  rewrite H.
  rewrite app_nil_l2.
  simpl; trivial.
Qed.

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

This lemma is similar to the lemmas:
- M1_corrected
- andb_false_r

Similarities:

```
- % response is defined
M3_3b is defined
```

Alasdair (HWU)  Machine Learning for ITP  28 March 2018  7 / 47
A proof in Coq with ML4PG help

```
Lemma M1_corrected : forall l : list A, l = [] -> tl (tl l) ++ nil = nil.
Proof.
  intro l.
  intro H.
  rewrite H.
  rewrite app_nil_l2.
  simpl; trivial.
  Qed.

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.
  trivial.
  Qed.
```
Can clusters help with proof discovery?

Three methods have been created to automatically analogize proofs from these clusters.
These methods look to show that:

- Clusters created by ML4PG contain similar lemmas.
- New proofs can be analogized from these clusters that brute force would be unable to find.
Method:

- For each lemma in cluster copy entire proof and see if it is valid in current lemma.

Example:
Prove lemma:

```
Lemma plus_Sn_m : forall n m: nat, S n + m = S (n + m).
```

Simple Search Example

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

Lemma plus_Sn_m : forall n m:nat, S n + m = S (n + m).
Proof.
  induction a.
  simpl; trivial.
  simpl; trivial.
Qed.

Error.
Searching mulnS, mult_n_O, aux10.
Lemma \( \text{mulnS} : \forall n \ m, n \times S \ m = n + n \times m. \)
Proof.
induction \( n. \)
   trivial. intro \( m. \)
rewrite mulSn. rewrite mulSn. rewrite addSn. rewrite addSn. rewrite addnCA.
rewrite IHn. trivial.
Qed.

Lemma \( \text{plus_Sn_m} : \forall n \ m: \text{nat}, S \ n + m = S \ (n + m). \)
induction \( n. \)
   trivial. intro \( m. \)
rewrite mulSn. rewrite mulSn. rewrite addSn. rewrite addSn. rewrite addnCA.
rewrite IHn. trivial.
Qed.

Error.
Searching mult_n_O, aux10.
Simple Search Example

Lemma \texttt{mult\_n\_0} : \texttt{forall n:nat, O = n \ast 0}.
Proof.
\texttt{induction n}.
\texttt{simpl; trivial}.
\texttt{simpl; trivial}.
Qed.

Lemma \texttt{plus\_S\_n\_m} : \texttt{forall n m:nat, S n + m = S (n + m)}.
Proof.
\texttt{induction n}.
\texttt{simpl; trivial}.
\texttt{simpl; trivial}.
Qed.

Proof Solved.
Simple Search

Success of simple search shows evidence towards the clusters being correct.

For Example:

<table>
<thead>
<tr>
<th>Library</th>
<th>Size</th>
<th>Simple</th>
<th>SimpleBrute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experimental</td>
<td>50</td>
<td>31 (\approx 62%)</td>
<td>40 (\approx 80%)</td>
</tr>
<tr>
<td>Paths (in Coq HoTT library)</td>
<td>41</td>
<td>38 (\approx 93%)</td>
<td>39 (\approx 95%)</td>
</tr>
</tbody>
</table>
## Depth First Search

**Method:**

1. Create list of lists of all tactics used in proofs of other lemmas in clusters.
2. Depth first search the list of tactics until proof is found or no tactics remaining.

**Example:**
Prove lemma:

**Lemma** `M26 : forall a b : nat, (0 - a) * S b = 0`.

With Cluster: M41, M37, M32, M31, M22
Depth First Search Proof Tree

```
intros.

rewrite O_minus.  rewrite <- mult_n_O.

rewrite <- mult_n_O.  rewrite aux12.  rewrite <- mult_O_n.

trivial.
```
Lemma M26 : \( \forall a \ b : \text{nat}, (0 - a) \ast S \ b = 0. \)

Proof.

\textit{intros}.
Lemma M26 : \( \forall a, b : \text{nat}, (0 - a) \ast S b = 0 \).

Proof.
\begin{align*}
\text{intros.} \\
\text{rewrite } O\_\text{minus}. \\
\end{align*}
Lemma M26 : \( \forall a \; b : \text{nat}, (0 - a) \ast S \; b = 0. \)

Proof.

\begin{itemize}
\item intros.
\item rewrite 0_minus.
\item rewrite <- mult_n_0.
\end{itemize}

Error.
**Lemma** M26 : \( \forall a \ b : \text{nat}, \ (0 - a) \ * \ S \ b = 0. \)

**Proof.**

\[
\begin{align*}
    &\text{intros.} \\
    &\text{rewrite 0_minus.} \\
    &\text{rewrite <- aux12.}
\end{align*}
\]

**Error.**
Lemma M26 : forall a b: nat, (O - a) * S b = 0.

Proof.
  intros.
  rewrite O_minus.
  rewrite <- mult_O_n.
Lemma M26 : \( \forall a \ b: \text{nat}, (0 - a) \star S b = 0. \)

Proof.

intros.
rewrite O_minus.
rewrite \(<\cdot \) mult_0_n.
trivial.
Qed.

Proof Solved.
Context Mining Search

Method:

1. Extract each lemma removing internal variable references.
2. Perform a depth first search on the extracted lemmas using variables from the context instead of the internal ones.
3. If there is a reference to an external lemma all other lemmas in its cluster are also tried.
Example:
Prove lemma:

Lemma M23 : forall a: nat, (a + 0) * S 0 = a.

With Cluster: andb_false_r, aux11, M1_corrected, aux12, mulSn, addSn, plus_0_n, app_nil_l2b, app_nil_l, mulnS, aux7, addnCA, addnS
Context Mining Search Example

How context mining search represents the proof found:

(1 . "induction")
(semi (0 . "simpl") (0 . "trivial"))
(semi (0 . "simpl") (0 . "trivial"))
(ext "rewrite" . "addSn")
(ext "rewrite" . "addnCA")
(1 . "rewrite")
(0 . "trivial")
Context Mining Search Example

(1. "induction")

One variable used in tactic. Possible variables from context: a

Lemma M23 : forall a: nat, (a + 0) * S 0 = a.
Proof.
induction a.
Context Mining Search Example

(semi (0 . "simpl") (0 . "trivial"))
No variables used in tactics and tactics are separated by a semi colon.

Lemma M23 : forall a: nat, (a + 0) * S 0 = a.
Proof.
induction a.
simpl; trivial.
(semi (0 . "simpl") (0 . "trivial"))
No variables used in tactics and tactics are separated by a semi colon.

Lemma M23 : forall a: nat, (a + 0) * S 0 = a.
Proof.
 induction a.
simpl; trivial.
simpl; trivial.
Context Mining Search Example

(ext "rewrite" . "addSn")

External rewrite with no arrows referenced.

Perform rewrite on variables in addSn clusters: addSn, andb_false_r, M23, aux11, M1_corrected, aux12, mulSn, plus_0_n, app_nil_l2b, app_nil_l

Lemma M23 : forall a: nat, (a + 0) * S 0 = a.
Proof.
induction a.
simpl; trivial.
simpl; trivial.
rewrite addSn.

Error.
Context Mining Search Example

Remaining lemmas: andb_false_r, M23, aux11, M1_corrected, aux12, mulSn, plus_0_n, app_nil_l2b, app_nil_l

Lemma M23 : forall a: nat, (a + 0) * S 0 = a.

Proof.
induction a.
simpl; trivial.
simpl; trivial.
rewrite andb_false_r.

Error.
Context Mining Search Example

Remaining lemmas: M23, aux11, M1_corrected, aux12, mulSn, plus_0_n, app_nil_l2b, app_nil_l

Lemma M23 : forall a: nat, (a + 0) * S 0 = a.

Proof.
  induction a.
  simpl; trivial.
  simpl; trivial.
  rewrite M23.

Error.
Context Mining Search Example

Remaining lemmas: aux11, M1_corrected, aux12, mulSn, plus_0_n, app_nil_l2b, app_nil_l

Lemma M23 : \forall a : \text{nat}, (a + 0) * S 0 = a.
Proof.
induction a.
simpl; trivial.
simpl; trivial.
rewrite aux11.

Error.
Context Mining Search Example

Remaining lemmas: M1_corrected, aux12, mulSn, plus_0_n, app_nil_l2b, app_nil_l

Proof.
induction a.
simpl; trivial.
simpl; trivial.
rewrite M1_corrected.

Error.
Context Mining Search Example

Remaining lemmas: aux12, mulSn, plus_0_n, app_nil_l2b, app_nil_l

Lemma M23 : forall a: nat, (a + 0) * S 0 = a.
Proof.
induction a.
simpl; trivial.
simpl; trivial.
rewrite aux12.

Error.
Context Mining Search Example

Remaining lemmas: mulSn, plus_0_n, app_nil_l2b, app_nil_l

Lemma M23 : forall a: nat, (a + 0) * S 0 = a.
Proof.
induction a.
simpl; trivial.
simpl; trivial.
rewrite mulSn.

Error.
Context Mining Search Example

Remaining lemmas: plus_0_n, app_nil_l2b, app_nil_l

Lemma M23 : \( \forall a : \text{nat}, (a + 0) \times \text{S} 0 = a. \)
Proof.
induction a.
simpl; trivial.
simpl; trivial.
rewrite plus_0_n.
Context Mining Search Example

(\text{ext "rewrite" . "addnS"})
External rewrite with no arrows referenced. Perform rewrite on variables in \text{addnCA}, \text{M23}, \text{mulnS}, \text{aux7}, \text{addnS}

\textbf{Lemma M23} : \text{forall } a: \text{nat}, (a + 0) \times S 0 = a.
\textbf{Proof.}
\text{induction } a.
\text{simpl; trivial.}
\text{simpl; trivial.}
\text{rewrite plus\_0\_n.}
\text{rewrite addnCA.}

\text{Error.}
Context Mining Search Example

Remaining lemmas: M23, mulnS, aux7, addnS

Lemma M23 : \( \forall a : \text{nat}, (a + 0) \times S 0 = a \).

Proof.

induction \( a \).
simpl; trivial.
simpl; trivial.
rewrite plus_0_n.
rewrite M23.

Error.
Remaining lemmas: mulnS, aux7, addnS

Lemma M23 : forall a: nat, (a + 0) * S 0 = a.
Proof.
induction a.
simpl; trivial.
simpl; trivial.
rewrite plus_0_n.
rewrite mulnS.

Error.
Remaining lemmas: aux7, addnS

Lemma M23 : forall a: nat, (a + 0) * S 0 = a.
Proof.
induction a.
simpl; trivial.
simpl; trivial.
rewrite plus_0_n.
rewrite aux7.

Error.
Remaining lemmas: addnS

Lemma M23 : \( \forall a : \mathbb{N}, (a + 0) \times S 0 = a \).  
Proof.  
  induction \( a \).  
  simpl; trivial.  
  simpl; trivial.  
  rewrite plus_0_n.  
  rewrite addnS.
(1. "rewrite")
Two variables available IHa and a. Trying rewrite on both.

Lemma M23 : forall a: nat, (a + 0) * S 0 = a.
Proof.
induction a.
simpl; trivial.
simpl; trivial.
rewrite plus_0_n.
rewrite addnS.
rewrite IHa.
(0. "trivial")
No variables used in tactic

Lemma M23 : forall a: nat, (a + 0) * S 0 = a.
Proof.
induction a.
simpl; trivial.
simpl; trivial.
rewrite plus_0_n.
rewrite addnS.
rewrite IHa.
trivial.
Qed.

Proof Solved.
CompCert Proof

Lemma  iregn_of_eq :
  \( \forall r \ r', \ iregn_of \ r = \text{OK} \ r' \rightarrow \ preg_of \ r = \text{IR} \ r' \).

Proof.
  unfold iregn_of; intros. destruct (preg_of r); inv H; auto.
Qed.
Another Context Mining Search Example

CompCert Proof

Lemma ireg_of_eq :
  \forall r \ r', \ ireg_of \ r = OK \ r' \ \rightarrow \ preg_of \ r = IR \ r'.
Proof.
  unfold \ ireg_of; \ intros. \ destruct \ (preg_of \ r); \ inv \ H; \ auto.
Qed.

Context Mining Proof

Lemma ireg_of_eq :
  \forall r \ r', \ ireg_of \ r = OK \ r' \ \rightarrow \ preg_of \ r = IR \ r'.
Proof.
  intros.
  destruct \ r, \ r'; \ inv \ H; \ auto.
Qed.
Context Mining Advantages

- Makes use of clustering to find additional lemmas to rewrite and apply.
- Stops errors due to using incorrect variable name.
- Finds brand new proof which cannot be found by brute force.
### Method Results

This table only counts lemmas that are in a cluster.

<table>
<thead>
<tr>
<th>Library</th>
<th>Size</th>
<th>Simple</th>
<th>DFS</th>
<th>CMS</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experimental</td>
<td>50</td>
<td>≈ 62%</td>
<td>≈ 66%</td>
<td>≈ 76%</td>
<td>≈ 80%</td>
</tr>
<tr>
<td>Paths (in Coq HoTT library)</td>
<td>41</td>
<td>≈ 93%</td>
<td>≈ 93%</td>
<td>≈ 80%</td>
<td>≈ 93%</td>
</tr>
</tbody>
</table>

Pending: CompCert
An add on for Proof General has been created for automatic analogizing of Coq Proofs.

Three methods for analogizing Coq proofs from ML4PG clusters in proof general have been created.

Clustering performed by ML4PG has been shown to find similar lemmas.

More complex searching algorithms can be run on these clusters to find new proofs.
An add on for Proof General has been created for automatic analogizing of Coq Proofs.

Three methods for analogizing Coq proofs from ML4PG clusters in proof general have been created.

Clustering performed by ML4PG has been shown to find similar lemmas.

More complex searching algorithms can be run on these clusters to find new proofs.

Further Work?