ML4PG: Machine Learning for Proof General

Ekaterina Komendantskaya and Jónathan Heras

http://www.computing.dundee.ac.uk/staff/katya/ML4PG/

University of Dundee

EPSRC Engineering and Physical Sciences Research Council

Interactive Theorem Provers (ITPs)

...are programming languages applied for:

★ **software verification in industry:**

- ARM microprocessor: 20,000 lines of code;
- verified C compiler: 50,000 lines of code;
- seL4 microkernel: 200,000 lines of code.

★ **and formalisation of mathematics:**

- Four Colour theorem: 60,000 lines of code;
- Feit-Thompson theorem: 170,000 lines;
- Flyspeck project: 325,000 lines of code.

Challenges

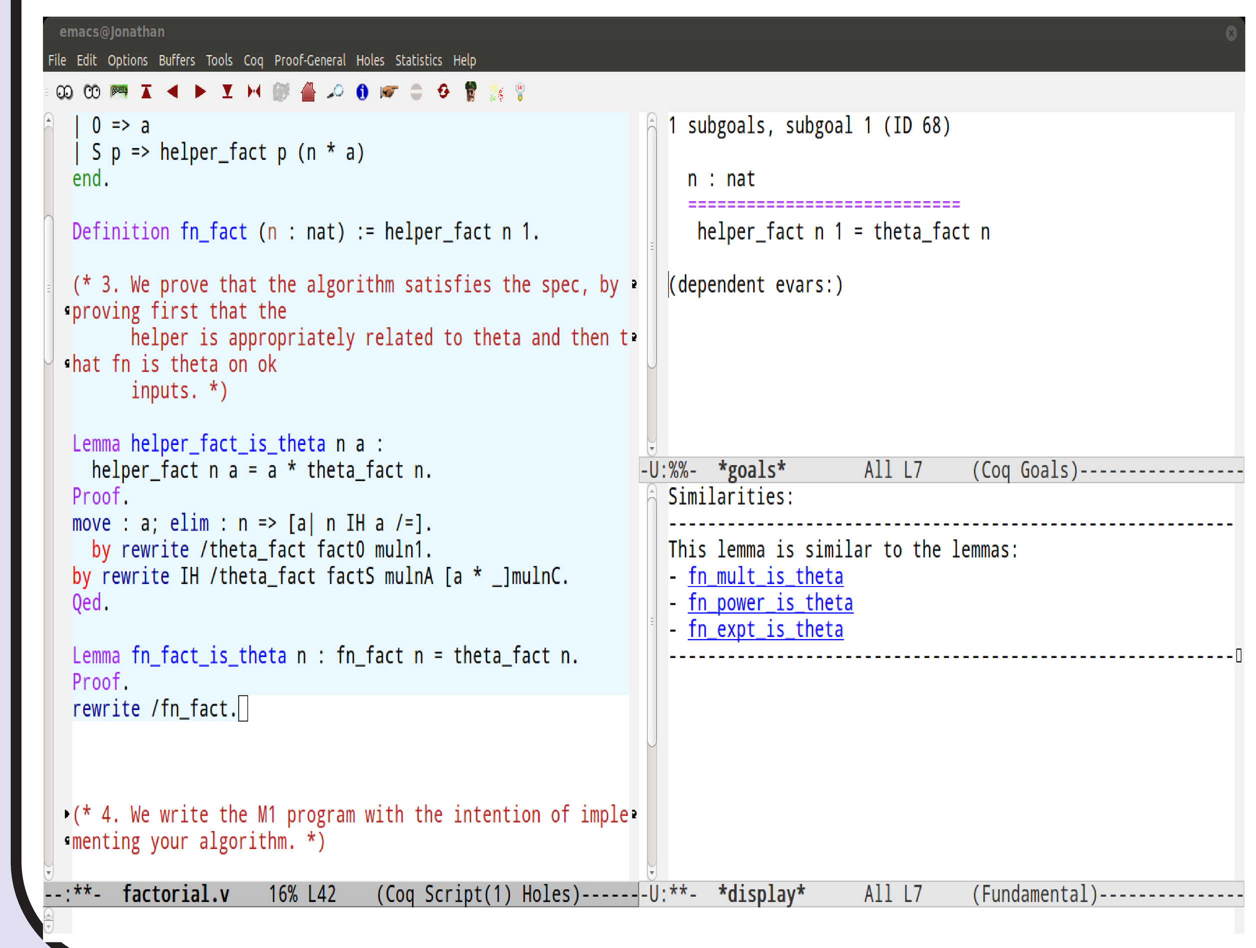
- Manual handling of various proofs, strategies and libraries becomes difficult;
- Team-development is hard, especially since ITPs are sensitive to user notation;
- Comparison of proofs and proof similarities across libraries or in one big library is hard.

Our solution: ML4PG

Proof General [1] is a user interface for several existing ITPs. **ML4PG** [4] is a machine-learning extension to **Proof General** [1] that:

- finds common proof-patterns in proofs across various scripts, libraries, users and notations;
- provides proof-hints, especially in industrial cases where routine similar cases are frequent, and effort is distributed across several programmers.

ML4PG session for Coq/SSReflect



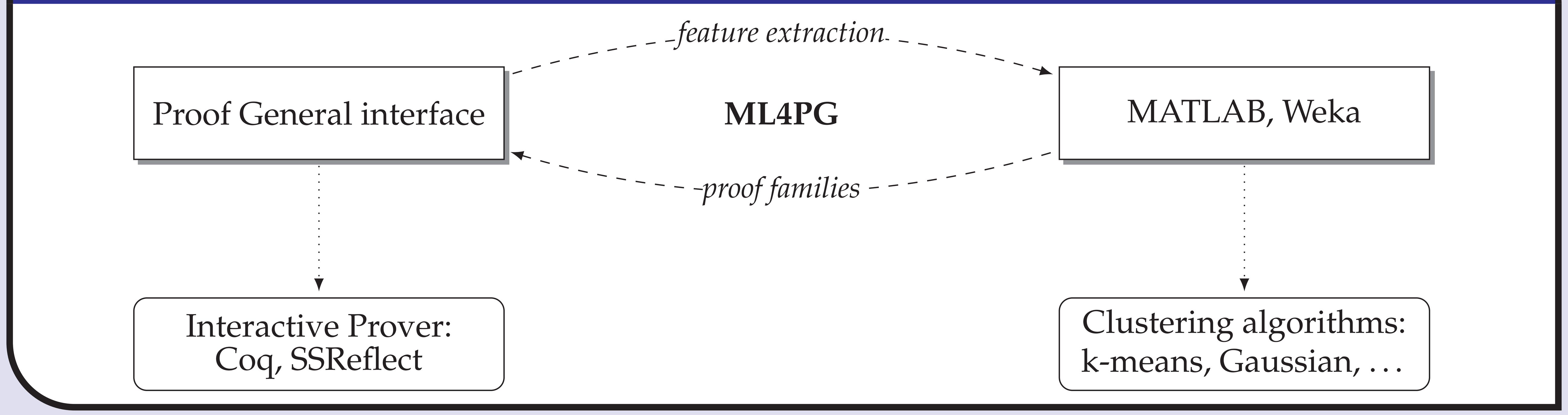
Step 1: Feature extraction

- ML4PG works on the background of Proof General, and extracts statistical features from interactive proofs in Coq [2] and SSReflect [3];
- The features reflect shapes of lemmas, structure of proofs, and patterns of user interaction with the ITP.
- Proof trace method captures statistical relation between several proof steps.

Step 2: Machine learning tools

- As higher-order proofs in general can take an infinite variety of shapes and sizes, ML4PG does not use any a priori given training labels.
- it uses unsupervised learning (clustering) algorithms implemented in MATLAB and Weka; and allows the user to adjust learning parameters, e.g. the size and proximity of clusters.
- The output shows families of related proofs.

ML4PG overview



Case Study: Verification of Java Virtual Machine with ML4PG

Java Virtual Machine (JVM) is a stack-based abstract machine which can execute Java bytecode. We modelled a subset of the JVM in Coq, verifying the interpreter for JVM programs. This work is inspired by the ACL2 proofs about JVMs [5].

```
static int factorial(int n)
{
  int a = 1;
  while (n != 0){
    a = a * n;
    n = n-1;
  }
  return a;
}
```

```
0  : iconst 1
1  : istore 1
2  : iload 0
3  : ifeq 13
4  : iload 1
5  : iload 0
6  : imul
7  : istore 1
8  : iload 0
9  : iconst 1
10 : isub
11 : istore 0
12 : goto 2
13 : iload 1
14 : ireturn
```

JVM model:

counter: 0

stack: | | | | ... |

local variables: | 5 | | | | ... |

```
Fixpoint helper_fact (n a : nat) :=
match n with
| 0 => a
| S p => helper_fact p (n * a)
end.

Definition fn_fact (n : nat) :=
      helper_fact n 1.
```

Case study: ML4PG Role in Proof Pattern Discovery

As part of JVM verification process, we needed to prove in Coq the following lemma:

Lemma 1 (Factorial JVM lemma) $\forall n \in \mathbb{N}$, *running the bytecode associated with the factorial program with* n *as input, the Coq JVM produces a state which contains* $n!$ *on top of the stack.*

After processing the proof statistics of 150 lemmas in the library, ML4PG correctly suggested to reuse the proof strategy from similar (already proven) lemmas concerning different operations:

$\star$ multiplication JVM lemma $\star\star$ power JVM lemma $\star\star\star$ exponentiation JVM lemma

Step 3: Interaction with ML4PG

- ML4PG automatically sends the gathered statistics to a chosen machine-learning interface and triggers execution of a clustering algorithm of the user's choice;
- it does some gentle post-processing of the results given by the machine-learning tool, and displays families of related proofs to the user.

Discovered Proof Families

Proof hints provided by ML4PG for the proof of **Lemma 1**: all proof families below contain proofs of already proven $\star$, $\star\star$, and $\star\star\star$.

Clustering algorithm:	$g=1$	$g=2$	$g=3$
Gaussian	9	0	0
K-means (MATLAB)	20	**3**	**3**
K-means (Weka)	29	10	2
FarthestFirst	27	24	0

Size of dataset: $\approx$ 150 lemmas. The granularity parameter g ranges from 1 (producing big and general clusters) to 5 (producing small and precise clusters).

Benefits of ML4PG

- ... can be switched on/off on user's demand;
- ... does not assume any machine-learning knowledge from the user;
- ... allows the user to make choices regarding approach to levels of proofs, size and granularity of clusters, and particular statistical algorithms;
- ...tolerant to mixing and matching different proof libraries and different notation used in proofs across different users.

References

[1] D. Aspinall. *Proof General: A Generic Tool for Proof Development*. In TACAS'00, LNCS 1785, pp. 38–43. 2000.

[2] Y. Bertot and P. Castéran. *Coq'Art: the Calculus of Constructions*. Springer-Verlag. 2004.

[3] G. Gonthier and A. Mahboubi. *An introduction to small scale reflection*. J. of Formalized Reasoning, 3(2):95–152. 2010.

[4] E. Komendantskaya, J. Heras, and G. Grov. *Machine learning in Proof General: interfacing interfaces*. To be published in EPTCS Post-proceedings of UITP'12. 2013.

[5] J S. Moore. Models, Algebras and Logic of Engineering Software, chapter Proving Theorems about Java and the JVM with ACL2, pages 227–290. IOS Press, 2004.