Machine Learning for ACL2

Jónathan Heras and Katya Komendantskaya

Motivation

Communities working on automated theorem provers and interactive theorem provers have accumulated big corpora of electronic proof libraries. However, the size of the libraries, as well as their technical and notational sophistication often stand on the way of efficient knowledge re-use. Very often, it is easier to start a new library from scratch rather than search the existing proof libraries for potentially common heuristics and techniques. Proof-pattern recognition is the area where statistical machine-learning is likely to make an impact.

ACL2 has elements of both interactive and automated provers. ACL2 is automatic in the sense that once started on a problem, it proceeds without human assistance. However, non-trivial results are not usually proved in the first attempt, and the user, inspired by the failed proof generated by ACL2, has to lead the prover to a successful proof. The main reasons why ACL2 gets stuck during a proof are (in the likelihood order):

R.1. Some auxiliary results have not been proved yet – the user is in charge of providing those auxiliary lemmas.

R.2. A concrete result is not used in a proof – the user should explicitly indicate the results that he wants to use.

R.3. Some lemmas produce endless proof-loops – the user needs to disable those rules.

R.4. ACL2 does not use the correct inductive scheme – the user will provide a proper scheme.

Final product

In this project, we intend to create a tool that uses machine-learning techniques to make the development of ACL2 proofs easier and faster. We have already implemented a similar *statistical machine-learning* interface in the theorem prover *Coq*.[1]

For ACL2, we will mainly focus on dealing with the problems related to reasons R.1 and R.2 – we do not consider either reason R.3 (this problem can be easily handled inspecting the trace of rules applied by ACL2) or reason R.4 (there already exist efficient tools for this purpose). With our tool, we will try to fulfill the following objectives organised in priority order.

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

O.1. After a proof-attempt has failed, the user will be able to call our *statistical machine-learning* hint generator, and it will display to the user a family of statistically similar theorems/lemmas. Inspecting those theorems, the user can get some insight on the auxiliary lemmas which should be introduced by analogy with the previous proofs, in order to prove the current goal (reason R.1).

O.2. On the top of the output produced in objective O.1, we can apply *symbolic machine-learning methods* to automatically generate the auxiliary lemmas (reason R.1) – this tool could be applied on user's demand or automatically after a failed proof. We currently investigate this idea jointly with a research group of A. Bundy in the University of Edinburgh.

O.3. In addition to families of similar proofs, the tool will be able to find relevant lemmas (already available in the system) to use in the proof of the current goal (cf. R.2). A filter of relevant lemmas can also be applied to fasten ACL2 proofs since we could disable rewrite rules which are not useful for the current goal but are tried during the proof.

Current prototype

A first prototype is available at `http://www.computing.dundee.ac.uk/staff/jheras/ml4acl2`. This prototype finds families of related proofs on the basis of lemma shapes and is a first step towards objective O.1. In the development of this tool, we already implement some important ideas that will contribute towards the final product:

- integration of ACL2 interface (emacs) with statistical machine-learning environments *Matlab* and *Weka*,
- automated extraction of relevant statistical features from ACL2 proofs, and
- manipulation of machine-learning parameters within the ACL2 interface.

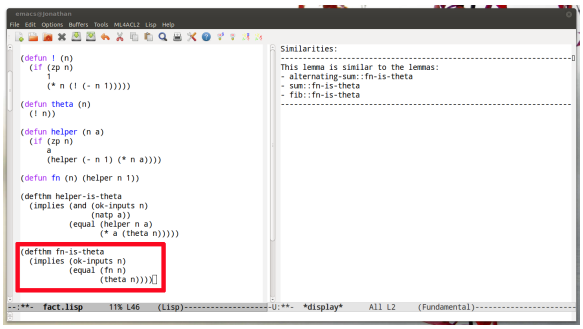

Figure 1: *Screenshot of the prototype implementation of the tool that data-mines proofs in ACL2. It shows the family of proofs similar to the chosen lemma* `fn-is-theta`*, as it is displayed to the ACL2 user, after the ACL2 proofs are clustered in machine-learning environments MATLAB and Weka.*