Interactive Theorem Provers

- ...are based on higher-order languages/type theory;
- ...provide a rich language to express and formalise theorems (in Mathematics) or software/hardware properties (in Verification).
- As opposed to “First-order Provers”, will never yield complete automation;
- ...but Machine Learning may help to data mine, extract, and generalise proof strategies.

Proof patterns: Example

We illustrate proof patterns by the proof of

\[(B \land A) \rightarrow (A \land B)\].

Tactic patterns. A tactic is a program which performs a proof step in a theorem prover:

- **Lemma**: \[(B \land A) \rightarrow (A \land B)\]
- **apply** (rule impl)
- **apply** (rule conj)
- **apply** (erule conjE)
- **apply** assumption
- **apply** (erule conjE)
- **apply** assumption
done

A tactic level pattern addresses the syntactic combination of the apply steps.

Proof-tree patterns. The complete proof tree can be used to learn strategies:

```
  (B \land A) \impl (A \land B)
    \uparrow
  (A \impl (B \land A))
  (A \land B) \impl (A \land B)
  (B \land A) \impl (A \land B)
```

**Term patterns.** Finally, we can ignore the tactics in the learning and only address how the term-structures of goals change.

Proof strategies

We need to generalise individual proof steps and repeated tactic applications into more general proof strategies. We also need to know when and how to apply a new proof strategy. To illustrate, the underlined step may create this strategy:

\[
\begin{align*}
\text{PRE top_level_symbol} & \in \{\rightarrow, \land\} \\
\text{WHILE top_level_symbol} & \in \{\rightarrow, \land\} \\
\text{apply} \ (\text{rule impl}) & \lor \text{apply} \ (\text{rule conj})
\end{align*}
\]

**Hypothesis**

We believe that our research goal can only be achieved by a combination of symbolic and numeric machine learning techniques. Our hypothesis is:

By utilising numeric machine learning techniques to detect proof patterns and generalise them into proof strategies by symbolic machine learning methods it is possible to automate the proofs of 'similar conjectures'.

Symbolic and statistical proof-pattern recognition

**Statistical Pattern Recognition**

- discover proof families
- discover clusters
- e.g. use of Neural Nets, SVM

**Symbolic Feature Abstraction**

- pre-process of proofs
- use of symbolic technique to create vectors
- e.g. use of ILP

**Symbolic Strategy Generalisation**

- use patterns/cluster to generalise proofs
- find conditions and generalise tactics
- e.g. ILP, HP, anti-unification

**Tactic pattern recognition**

- There is always a finite number of tactics;
- this makes the tactics most obvious candidates for features in pattern-recognition;
- in particular, variable-length Markov Networks seem to be a suitable model.

Additionally, we want our ML tool to take into account and detect:

1. arguments of tactics: because the semantics of a tactic often depends on its inputs;
2. combinations of tactics: various tactic combinations may have different effects and semantics;
3. more abstract families of tactics: different tactics may have same semantics in certain proofs.

**Term pattern recognition**

- Goals and subgoals have term-structures of their own;
- they change as directed by tactics;
- have been studied separately from their tactic relation.

**Geometric Kernels** and methods alike seem to suit best to detect such patterns.

Additionally, we want ML to take into account:

1. local assumptions made and used while completing a proof;
2. global hypotheses and lemmas used;
3. abstract features of term structures and their relative change in proofs.

Proof-tree pattern recognition

- combines effects of term-structures and tactic combinations;
- shows a broader proof strategy.

Additionally, we want our ML tool to take into account and detect:

1. dependencies between tactics and term structures: tactic applications drive the proof; however, sequences of tactics taken independently of goals do not have any meaning;
2. abstracted input and output of tactics – in particular those forming proof families.
3. hierarchies of more abstract tactics (combinations).

Key challenges

- The ability to handle noise. For each proof there will be a large amount of data, and a small part of data that is relevant. However, finding which parts are relevant is very difficult.

- Unpredictable (tactic) behaviour. The smallest change in the goals/configuration could result in drastic changes to tactic behaviour.

- The ability to handle relational data. There is often key relations between goals, their hypotheses and underlying theorems.

- The ability to handle complex and expressive data. Most theorem provers use a higher-order logic, thus, we need intelligent ways of extracting feature vectors.

- The need for generalisation and executable strategies. Our goal is to be able to re-use a learned proof strategy for the same proof family.

Acknowledgments

We would like to thank partners on the AI4FM project and Edinburgh MRG group. The work has been funded by EPSRC grants AI4FM: using AI to aid automation of proof search in formal methods (EP/H024204/1) and machine learning coalgebraic automated proofs (EP/J014222/1).