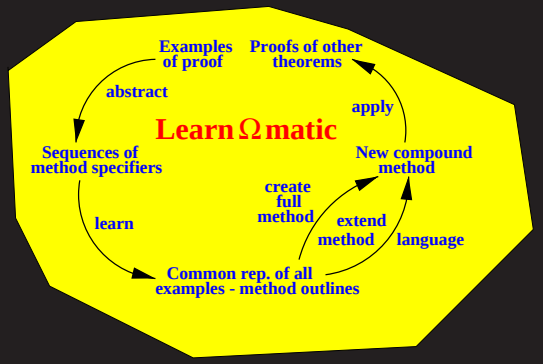
Automatic Learning of Proof Methods in Proof Planning

Mateja Jamnik[1,2], Manfred Kerber[1], Martin Pollet[1,3], Christoph Benzmüller[1,3]

[1] The University of Birmingham, [2] University of Cambridge, [3] University of Saarbrücken

Aims

- Emulate human learning from examples
- Find appropriate learning technique
- Appropriate method representation is crucial
- The big picture:
  - Systems starts with a few inference rules
  - Simple proof methods are then learned
  - Progressively, more complex proof methods learned



From Method Outlines to Usable Methods

- Have well-chosen example proofs
- Abstract them into sequences of rule identifiers
- Learn method outlines using learning algorithm.
- Hence: need to restore missing information to method outlines so proof planner can use the new methods

In ΩMEGA:

- Add preconditions and parameters to method outlines
- Precondition of learned method is true if there is instantiation of method outline where each of its methods is applicable
- But: cannot get precondition without instantiating methods of method outline
- Hence: need to apply methods of method outline to find the one suitable for the new proof situation

Example

Residue Class Theorems:

1. $closed\text{-}under(\mathbb{Z}_3, (\lambda x\, \lambda y_{\blacksquare}\, x \bar{+} (x \bar{+} y)))$
2. $associative\text{-}under(\mathbb{Z}_3, (\lambda x\, \lambda y_{\blacksquare}\, (x \bar{\times} y)))$
3. $commutative\text{-}under(\mathbb{Z}_2, (\lambda x\, \lambda y_{\blacksquare}\, (x \bar{+} y)))$

Traces:

1. $[defn\text{-}exp,\ \forall_i\text{-}sort, \forall_i\text{-}sort,\ convert\text{-}resclass\text{-}to\text{-}num,\ defn\text{-}exp,\ or\text{-}e\text{-}rec,\ simp\text{-}num\text{-}exp,\ simp\text{-}num\text{-}exp,\ \ldots simp\text{-}num\text{-}exp]$
2. $[defn\text{-}exp,\ \forall_i\text{-}sort,\ \forall_i\text{-}sort,\ \forall_i\text{-}sort,\ convert\text{-}resclass\text{-}to\text{-}num,\ or\text{-}e\text{-}rec,\ simp\text{-}num\text{-}exp,\ simp\text{-}num\text{-}exp,\ \ldots simp\text{-}num\text{-}exp]$
3. Similar to 2.

Learned Method Outline:

$\Big[defn\text{-}exp, [\forall_i\text{-}sort]^*, convert\text{-}resclass\text{-}to\text{-}num, [[or\text{-}e\text{-}rec]|[defn\text{-}exp,\ or\text{-}e\text{-}rec]], simp\text{-}num\text{-}exp^*\Big]$

Method Outlines

Primitives:	$\forall p \in P$, let $p \in L$
Sequence:	$\forall l_1, l_2 \in L$, let $[l_1, l_2] \in L$
Disjunction:	$\forall l_1, l_2 \in L$, let $[l_1 \vert l_2] \in L$
Repeat:	$\forall l \in L$, let $l^* \in L$ and $\forall l \in L, \forall n \in \mathbb{N}$, let $l^n \in L$

Learning algorithm

1. Split examples into all possible sublists.
2. For each sublist in each example find consecutive repetitions, (patterns)
3. Find patterns that match in all examples
4. If no matches, no generalisation
5. Generalise with Kleene star or constant.
6. Repeat the process on both sides of pattern.
7. Choose smallest generalisation

Related work:

- Machine learning in machine-oriented theorem proving: Fuchs, Schulz
- Precondition analysis to learn inference schemas: Silver, Desimone
- Analogy: Melis, Whittle; proof reuse: Kolbe, Walter
- Grammatical inference: Muggleton

Further Examples

- Set Theory
- Group Theory

Future Work

- Realisation in λ-Clam
- Include in evolutionary cycle (to get rid of need for well-chosen examples)
- Learn more accurate preconditions

This work was supported by EPSRC grant GR/M22031, "Formation of Methods" & EC IHP grant HPRN-CT-2000-00102, "Calculemus"

Further Information: www.cs.bham.ac.uk/~mmk/projects/MethodFormation/