Typed Meta-Interpretive Learning for Proof Strategies

Colin Farquhar, Gudmund Grov,
Andrew Cropper, Stephen Muggleton
& Alan Bundy

Imperial College
London

Motivation

- Use of theorem proving software increasingly widespread in industry
- Manual intervention often required; time-consuming and requires a skilled user
- “Stuck” proofs often cluster into groups which can be addressed using the same strategy

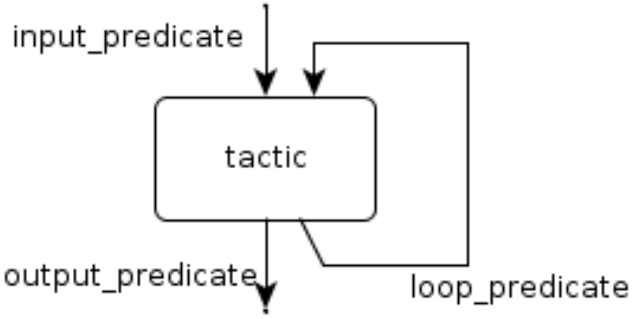
Motivation

- We demonstrate that *meta-interpretive learning* (MIL) can be used to learn these strategies in order to re-apply them elsewhere
- These strategies have a high degree of branching and thus a large search space
- We introduce *typed MIL* and hypothesise:

"Typed MIL learns more deterministic proof strategies than untyped MIL"

PSGraph

- Graphical representation of proof strategies [2]
- Nodes represent tactics, predicates on wires direct goals between nodes and ensure correctness



An example PSGraph with edges labelled by predicates

Metagol

- We use the *Metagol* [3] implementation of MIL
- Encode tactics and goal data as background information
- Successful proof evaluations given as positive examples
- Metarules applied to background to find a definition which satisfies examples

Metagol

- Metagol allows *predicate invention*, i.e. missing definitions can be found using available background information
- We can use this to find wire predicate definitions:

```
strategy(A,B) :- strategy_1(A), tactic(A,B).

    strategy_1(A) :- has_symbol(A,C).
```

Metagol

- Two separate learning problems: structure and conditions
- To learn both we must either restrict wire predicates or learn a strategy with a high degree of branching
- To resolve this we introduce *typed MIL*

Typed MIL

- All predicates and arguments are “tagged” with a constant representing their type:

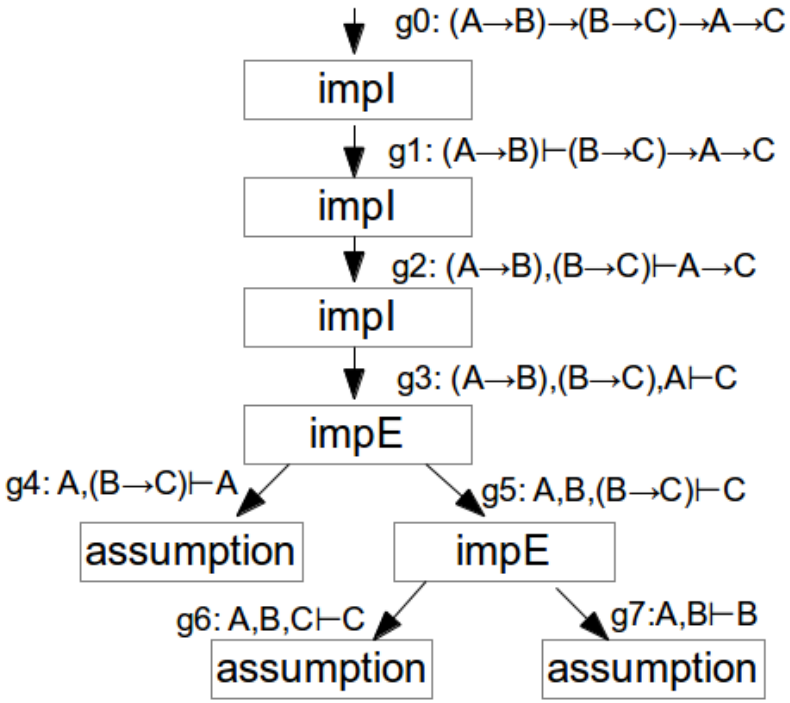
$$P(X,Y) \Rightarrow P{:}t_1(X{:}t_2,Y{:}t_3).$$

- To work in Metagol we treat the predicate type as an extra argument:

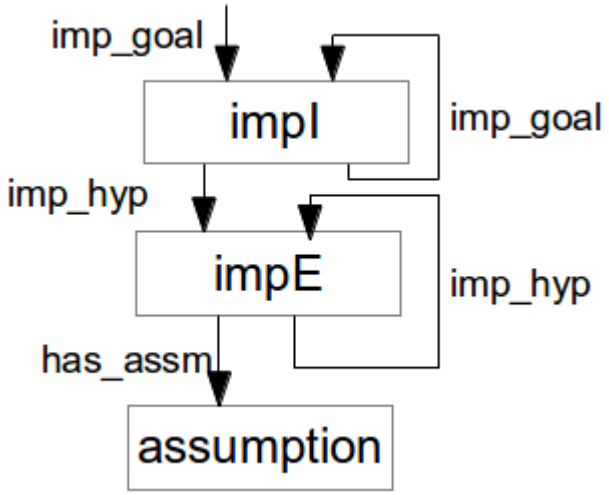
$$P{:}t_1(X,Y) \Rightarrow P(t_1,X,Y).$$

An Example Strategy

(A → B) → (B → C) → A → C



Proof Tree



PSGraph

Learned Definitions

Untyped Strategy

```
strat_i(A,B) :- erule_impE(A,C),assumption(C,B).
strat_i(A,B) :- erule_impE(A,C),strat_i(C,B).
strat_i(A,B) :- rule_impI(A,C),strat_i(C,B).
```

Typed Strategy

```
strat_i(psgraph,A,B) :-
      assm_type(wpred,A),assumption(tactic,A,B).
strat_i(psgraph,A,B) :-
      strat_i_1(psgraph,A,C),strat_i(psgraph,C,B).
strat_i_1(psgraph,A,B) :-
      impE_type(wpred,A),erule_impE(tactic,A,B).
strat_i_1(psgraph,A,B) :-
      impI_type(wpred,A),rule_impI(tactic,A,B).
```

Results

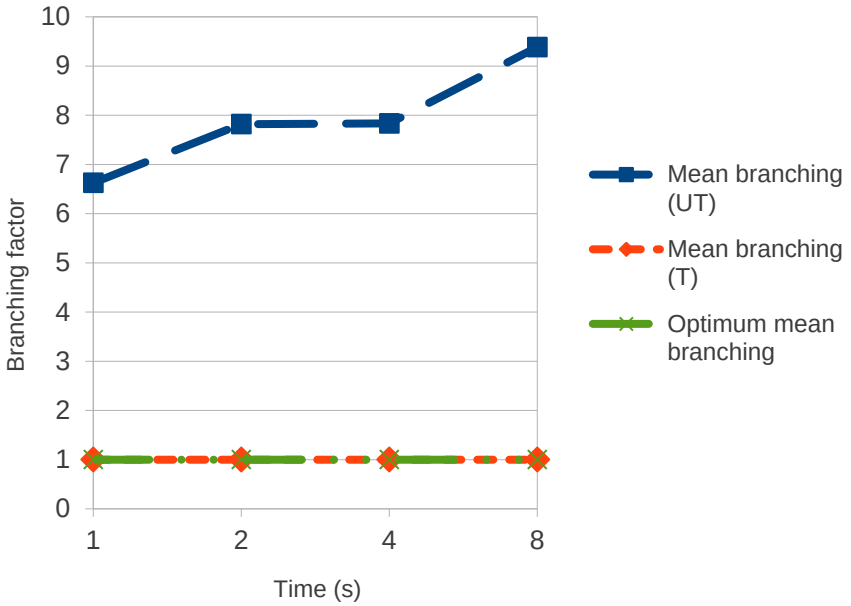
- Learned strategies from 15 propositional logic proofs
- Metagol run for 1, 2, 4 and 8 seconds

	Successes	Mean nodes	Mean clauses	Mean branches	Mean evaluations
Untyped	13	4	3	9	1
Typed	7	4	4	1	2

Average learning results across all experiments

Results

- Consider branching factor σ for each strategy.
- Large σ indicates large search space
- Untyped: $\sigma > 1$ and increases
- Typed: $\sigma = 1$ and constant



Mean branching factor of strategies learned using untyped (UT) and typed (T) MIL compared to the optimum branching factor

Conclusions

- Typed strategies have less branching than untyped strategies
- Thus typed strategies are more deterministic
- Initial results using this approach have been very promising

Further Work

- Developing strategies from multiple proofs – already underway. Will use examples from group theory.
- Moving on to larger, more complex proofs, e.g. rippling [1]. A simplified version can be learned with untyped MIL, can we learn a full version with types?

References

[1] A. Bundy, *Rippling: meta-level guidance for mathematical reasoning,* Vol. 56. Cambridge University Press, 2005.

[2] G. Grov, A. Kissinger and Y. Lin, *A graphical language for proof strategies*, in *LPAR*, volum 8312 of *LNCS*, pages 324-339. Springer, 2013

[3] D. Lin, E. Dechter, K. Ellis, J. B. Tenenbaum and S. H. Muggleton, *Bias reformation for one-shot function induction*, in *ECAI*, pages 525-530, 2014

This work has been supported by EPSRC grant EP/J001058/1 and the first author is supported by a James Watt scholarship. The fourth author acknowledges support from his Royal Academy of Engineering/Syngenta Research Chair