Top Program Construction and Reduction for Polynomial-Time Meta-Interpretive Learning*

Stassa Patsantzis, Stephen Muggleton
Imperial College London

Top Program Construction (simplified)

Input: Positive and negative examples E^+, E^-; Background knowledge B; Metarules $\mathcal{M}$.

Output: The Top Program, $\top$, a set of clauses each of which entails at least one positive and exactly 0 negative examples.

Construction: Clauses in $\top$ are instances of the metarules derived by SLD-refutation of examples in E^+ with clauses in B and $\mathcal{M}$.

Algorithm:

1. Generalisation step: Construct each clause that entails one or more examples in E^+ and add that clause to $\top$.

2. Specialisation step: Remove from $\top$ each clause that entails one or more examples in E^-.

3. Return $\top$.

Motivation

Top Program Construction avoids an exponential-time search of a possibly large Hypothesis Space. Instead, the Top Program is *constructed* in time polynomial to the set of constructible clauses.

Implementation

Top Program Construction is implemented in Prolog in the new Meta-interpretive Learning system Louise:

https://github.com/stassa/louise/

Louise constructs a Top Program and then reduces it to remove logically redundant clauses.

Top program construction

MIL problem

$E^+ = \{path(a,b) \leftarrow, path(a,c) \leftarrow\}$
$E^- = \{\leftarrow path(1,2), \leftarrow path(1,3)\}$
$B = \{edge\_alpha(a,b), edge\_alpha(b,c),$
$edge\_alnum(a,b), edge\_alnum(b,c),$
$edge\_alnum(1,2), edge\_alnum(2,3)\}$
$\mathcal{M} = \{P(x,y) \leftarrow Q(x,y),$
$P(x,y) \leftarrow Q(x,z), R(z,y)\}$

A Meta-Interpretive Learning problem consists of examples, background definitions and second-order metarules.

Generalisation step

$path(x,y) \leftarrow edge\_alnum(x,y)*$
$path(x,y) \leftarrow edge\_alpha(x,y)$
$path(x,y) \leftarrow path(x,y)$
$path(x,y) \leftarrow edge\_alnum(x,z), edge\_alnum(z,y)*$
$path(x,y) \leftarrow edge\_alnum(x,z), edge\_alpha(z,y)$
$path(x,y) \leftarrow edge\_alpha(x,z), edge\_alnum(z,y)$
$path(x,y) \leftarrow edge\_alpha(x,z), edge\_alpha(z,y)$
$path(x,y) \leftarrow path(x,z), edge\_alnum(z,y)$
$path(x,y) \leftarrow path(x,z), edge\_alpha(z,y)$

In the Generalisation step clauses are constructed by refutation of examples by SLD-resolution with clauses in B and $\mathcal{M}$.

Left, clauses marked with * entail negative examples.

Specialisation step

$path(x,y) \leftarrow edge\_alpha(x,y)$
$path(x,y) \leftarrow path(x,y)$
$path(x,y) \leftarrow edge\_alnum(x,z), edge\_alpha(z,y)$
$path(x,y) \leftarrow edge\_alpha(x,z), edge\_alnum(z,y)$
$path(x,y) \leftarrow edge\_alpha(x,z), edge\_alpha(z,y)$
$path(x,y) \leftarrow path(x,z), edge\_alnum(z,y)$
$path(x,y) \leftarrow path(x,z), edge\_alpha(z,y)$

Clauses entailing negative examples are removed in the Specialisation step.

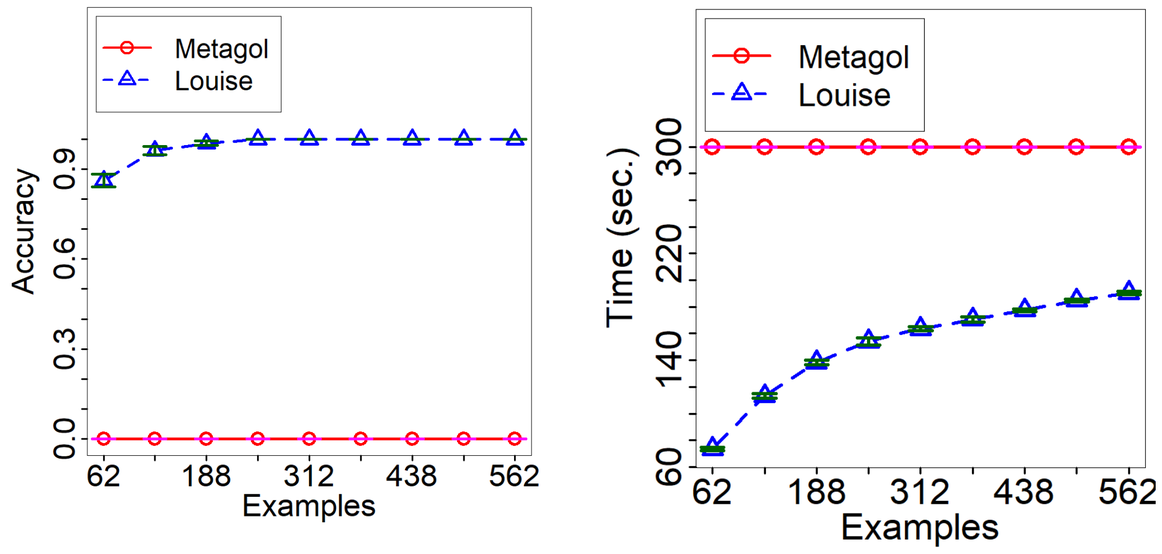
Top program reduction

$path(x,y) \leftarrow edge\_alpha(x,y)$
$path(x,y) \leftarrow edge\_alnum(x,z), edge\_alpha(z,y)$
$path(x,y) \leftarrow path(x,z), edge\_alnum(z,y)$
$path(x,y) \leftarrow path(x,z), edge\_alpha(z,y)$

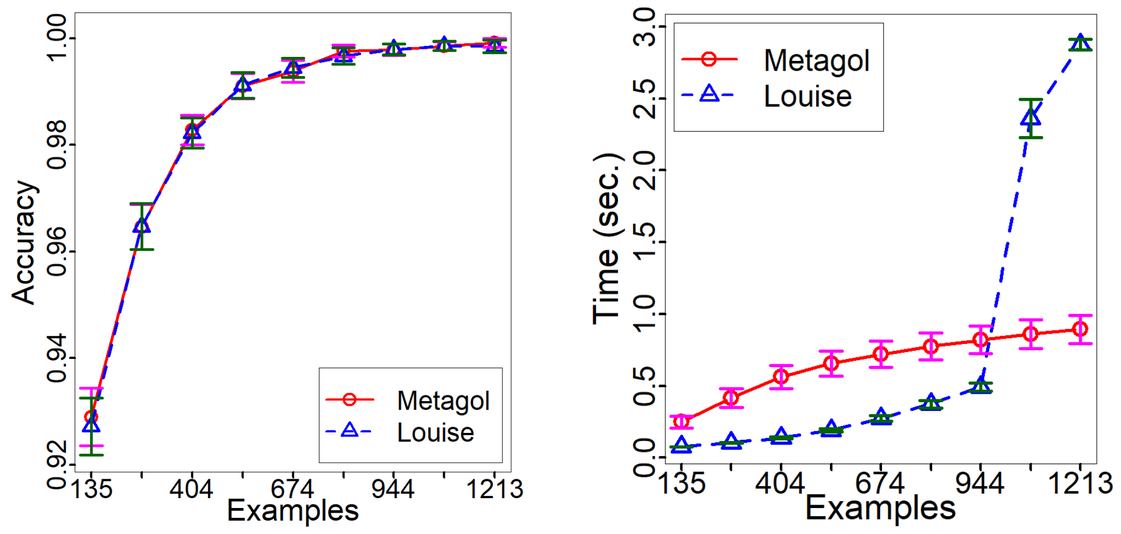
Logically redundant clauses are removed from the Top program by Plotkin's program reduction algorithm.

Comparing Louise to Metagol

Experiment 1: Grid world navigation

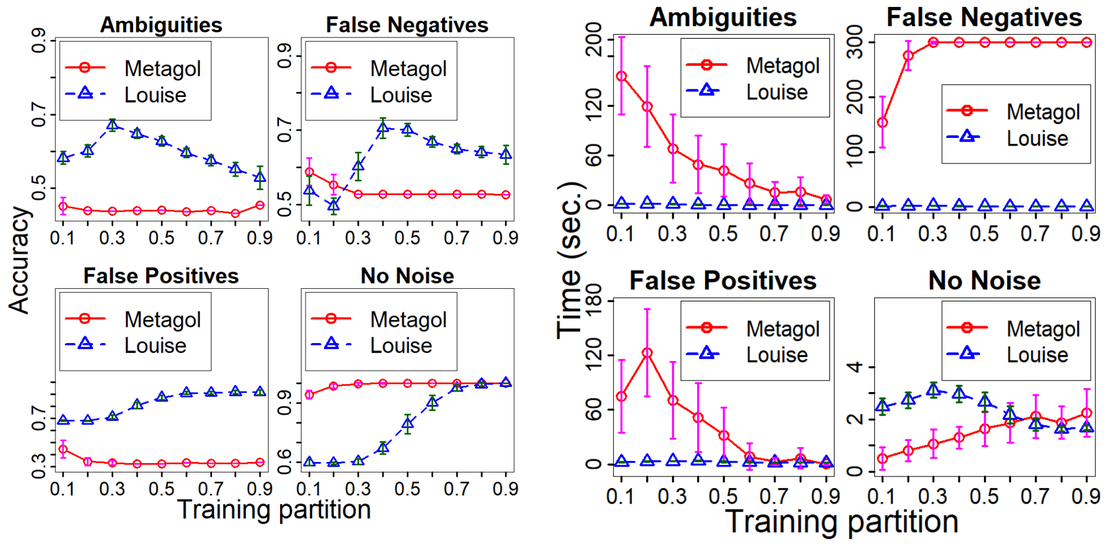


Experiment 2: Learning a CNL grammar (M:tG Fragment)

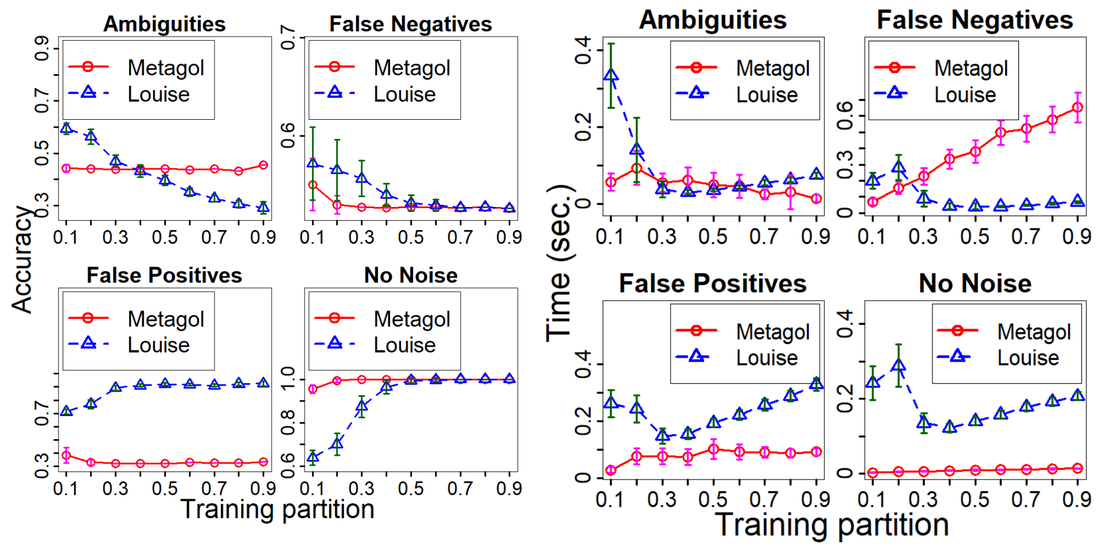


Experiment 3: Learning with mislabelled examples

Coloured graph connectedness – Some irrelevant background knowledge.



Coloured graph connectedness – No Irrelevant background knowledge.



* Recipient of the best ILP student paper award (journal track)