Meta-Interpretive Learning: achievements and challenges

Stephen Muggleton

Department of Computing
Imperial College, London

Motivation

Logic Programming [Kowalski, 1976]

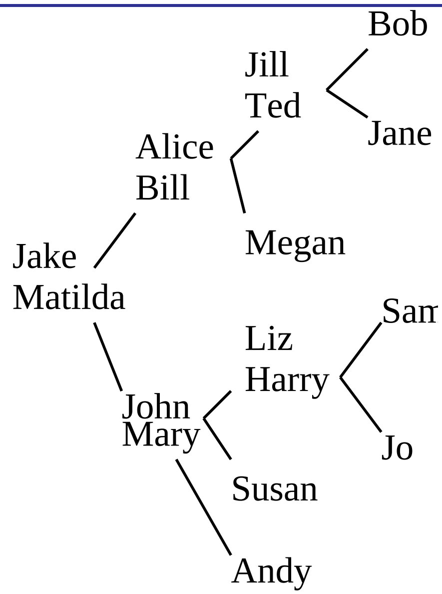
Inductive Logic Programming [Muggleton, 1991]

Machine Learn arbitrary programs

State-of-the-art ILP systems lacked Predicate Invention and Recursion [Muggleton et al, 2011]

Family relations (Dyadic)

Family tree



Target Theory

$$father(ted, bob) \leftarrow$$

$$father(ted, jane) \leftarrow$$

$$parent(X, Y) \leftarrow mother(X, Y)$$

$$parent(X, Y) \leftarrow father(X, Y)$$

$$ancestor(X, Y) \leftarrow parent(X, Y)$$

$$ancestor(X, Y) \leftarrow parent(X, Z),$$
$$ancestor(Z, Y)$$

Generalised Meta-Interpreter

$$prove([], BK, BK).$$

$$prove([Atom|As], BK, BK\_H) :-$$

$$metarule(Name, MetaSub, (Atom \text{ :- } Body), Order),$$

$$Order,$$

$$save\_subst(metasub(Name, MetaSub), BK, BK\_C),$$

$$prove(Body, BK\_C, BK\_Cs),$$

$$prove(As, BK\_Cs, BK\_H).$$

Metarules

Name	Meta-Rule	Order
Instance	$P(X, Y) \leftarrow$	$True$
Base	$P(x, y) \leftarrow Q(x, y)$	$P \succ Q$
Chain	$P(x, y) \leftarrow Q(x, z), R(z, y)$	$P \succ Q, P \succ R$
TailRec	$P(x, y) \leftarrow Q(x, z), P(z, y)$	$P \succ Q$, $x \succ z \succ y$

Meta-Interpretive Learning (MIL)

First-order	Meta-form
Examples ancestor(jake,bob) ⟵ ancestor(alice,jane) ⟵	**Examples** prove([ancestor(jake,bob), ancestor(alice,jane)], ..) ⟵
Background Knowledge father(jake,alice) ⟵ mother(alice,ted) ⟵	**Background Knowledge** instance(father,jake,john) ⟵ instance(mother,alice,ted) ⟵
Instantiated Hypothesis father(ted,bob) ⟵ father(ted,jane) ⟵ p1(X,Y) ⟵ father(X,Y) p1(X,Y) ⟵ mother(X,Y) ancestor(X,Y) ⟵ p1(X,Y) ancestor(X,Y) ⟵ p1(X,Z), ancestor(Z,Y)	**Abduced facts** instance(father,ted,bob) ⟵ instance(father,ted,jane) ⟵ base(p1,father) ⟵ base(p1,mother) ⟵ base(ancestor,p1) ⟵ tailrec(ancestor,p1,ancestor) ⟵

Logical form of Metarules

General form

$$
\begin{aligned}
P(X,Y) &\leftarrow Q(X,Y) \\
P(X,Y) &\leftarrow Q(X,Z), R(Z,Y)
\end{aligned}
$$

Metarule general form used in Family Relations

$$\exists P, Q, ..\forall X, Y, ..P(X, ..) \leftarrow Q(Y, ..), ..$$

Supports predicate/object invention and recursion.

In Family Relations we consider hypotheses in H_2^2, which contains predicates with arity at most 2 and has at most 2 atoms in the body.

Minimising sets of Metarules [ILP 2014]

Set of Metarules	Reduced Set
$P(X,Y) \leftarrow Q(X,Y)$	
$P(X,Y) \leftarrow Q(Y,X)$	$P(X,Y) \leftarrow Q(Y,X)$
$P(X,Y) \leftarrow Q(X,Y), R(Y,X)$	
$P(X,Y) \leftarrow Q(X,Y), R(Y,Z)$	
$P(X,Y) \leftarrow Q(X,Y), R(Z,Y)$	
$P(X,Y) \leftarrow Q(X,Z), R(Z,Y)$	$P(X,Y) \leftarrow Q(X,Z), R(Z,Y)$
..	
$P(X,Y) \leftarrow Q(Z,Y), R(Z,X)$	

Expressivity of H_2^2

Given an infinite signature H_2^2 has Universal Turing Machine expressivity [Tarnlund, 1977].

utm(S,S)	$\leftarrow$	halt(S).
utm(S,T)	$\leftarrow$	execute(S,S1), utm(S1,T).
execute(S,T)	$\leftarrow$	instruction(S,F), F(S,T).

Q: How can we limit H_2^2 to avoid the halting problem?

Metagol implementation (1)

- Ordered Herbrand Base [Knuth and Bendix, 1970; Yahya, Fernandez and Minker, 1994] - guarantees termination of derivations. Lexicographic + interval.
- Episodes - sequence of related learned concepts.
- $0, 1, 2, ..$ clause hypothesis classes tested progressively.
- Log-bounding (PAC result) - $log_2 n$ clause definition needs n examples.
- YAP implementation - https://github.com/metagol/metagol
.

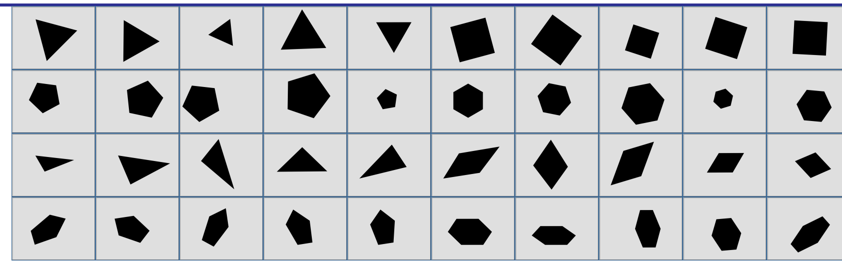
Metagol implementation (2)

- Andrew Cropper's YAP implementation - https://github.com/metagol/metagol
.
- Hank Conn's Web interface - https://github.com/metagol/metagol_web_interface
.
- Live web-interface - http://metagol.doc.ic.ac.uk

Vision applications (1)

Staircase

ILP 2013



Regular Geometric

ILP 2015

stair(X,Y) :- stair1(X,Y).

stair(X,Y) :- stair1(X,Z), **stair(Z,Y)**.

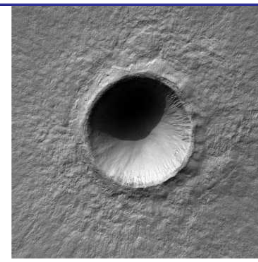
stair1(X,Y) :- vertical(X,Z), horizontal(Z,Y).

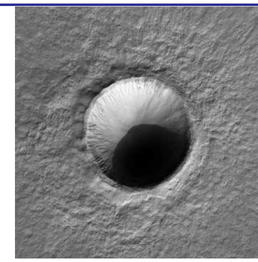
Learned in 0.08s on laptop from single image.

Note Predicate invention and **recursion**.

Vision applications (2) - ILP2017 - Object invention

Example Mars Images



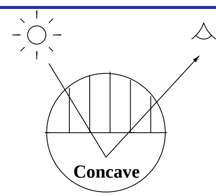


lit(obj1,north).   lit(obj1,south).

Background Knowledge

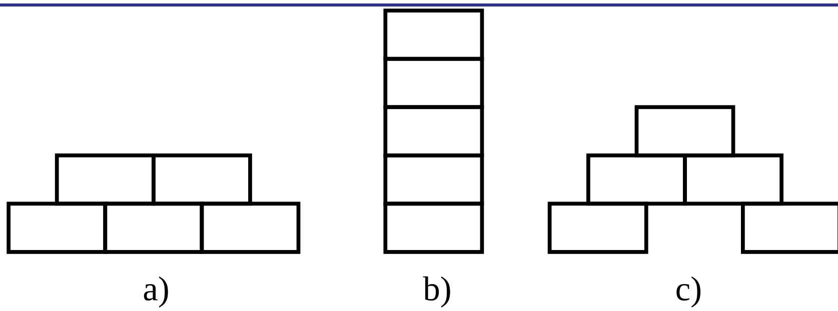
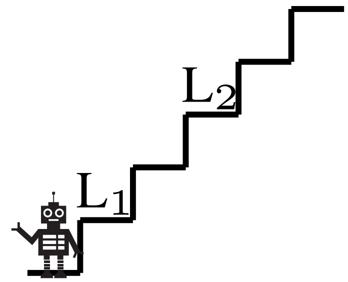
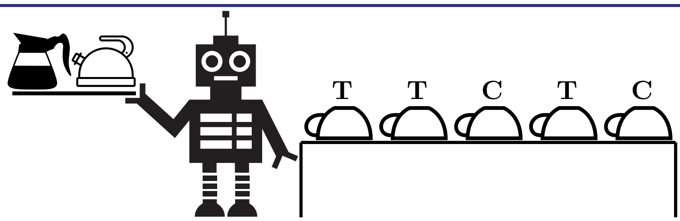
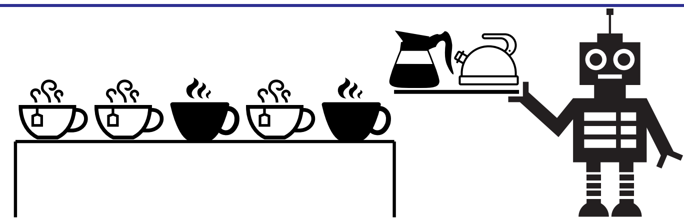
```
light_path(X,X).
light_path(X,Y) :- reflect(X,Z), light_path(Z,Y).
highlight(X,Y):- contains(X,Y), brighter(Y,X), light(L),
  light_path(L,Y), reflector(Y), light(Y,O), observer(O).
hl_angle(obj1,hlight,south). % highlight angle
opposite(north,south). opposite(south,north).
```

Hypothesis Image1




```
lit(A,B):- lit1(A,C), lit3(A,B,C).
lit1(A,B):- highlight(A,B), lit2(A), lit4(B).
lit3(A,B,C):- hl_angle(A,B,D), opposite(D,C).
lit2(obj1). % concave
lit4(hlight). % highlight
light(light1). observer(observer1). reflector(hlight).
reflect(obj1,hlight). reflect(hlight, observer1).
```

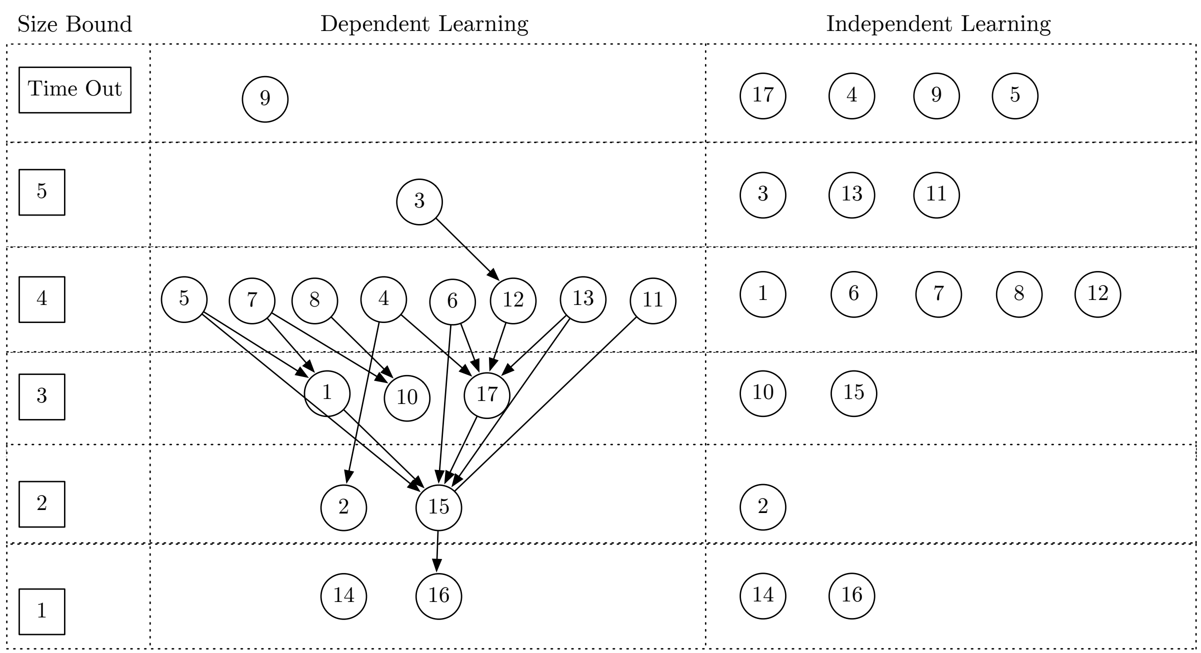
Robotic applications

	
Building a Stable Wall IJCAI 2013	Learning Efficient Strategies IJCAI 2015
	
Initial state IJCAI 2016	Final state Abstraction and Invention

Language applications

Formal grammars [MLJ 2014]

Dependent string transformations [ECAI 2014]



Chain of programs from dependent learning

f_{03}(A,B) :- f_{12_1}(A,C), f_{12}(C,B).

f_{12}(A,B) :- f_{12_1}(A,C), f_{12_2}(C,B).

f_{12_1}(A,B) :- f_{12_2}(A,C), $skip1$(C,B).

f_{12_2}(A,B) :- f_{12_3}(A,C), $write1$(C,B,'.').

f_{12_3}(A,B) :- $copy1$(A,C), f_{17_1}(C,B).

f_{17}(A,B) :- f_{17_1}(A,C), f_{15}(C,B).

f_{17_1}(A,B) :- f_{15_1}(A,C), f_{17_1}(C,B).

f_{17_1}(A,B) :- $skipalphanum$(A,B).

f_{15}(A,B) :- f_{15_1}(A,C), f_{16}(C,B).

f_{15_1}(A,B) :- $skipalphanum$(A,C), $skip1$(C,B).

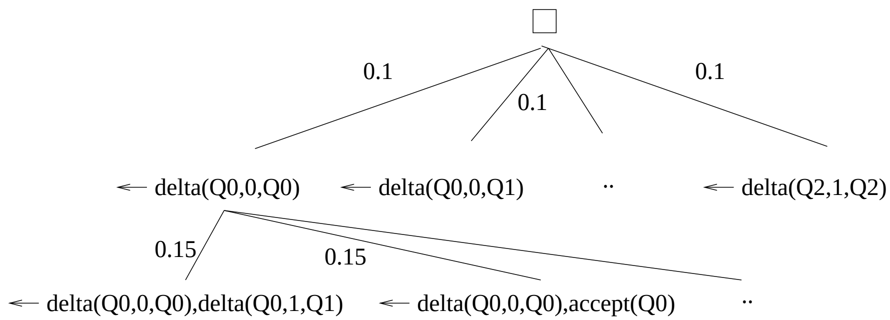
f_{16}(A,B) :- $copyalphanum$(A,C), $skiprest$(C,B).

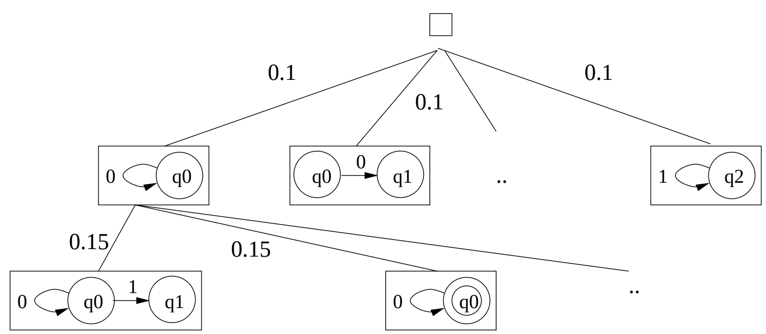
Other applications

Learning proof tactics [ILP 2015]

Learning data transformations [ILP 2015]

Bayesian Meta-Interpretive Learning

<table>
<tr>
<td>Clauses</td>
<td>


</td>
</tr>
<tr>
<td>Finite
State
Acceptors
(FSAs)</td>
<td>


</td>
</tr>
</table>

Related work

Predicate Invention. Early ILP [Muggleton and Buntine, 1988; Rouveirol and Puget, 1989; Stahl 1992]

Abductive Predicate Invention. Propositional Meta-level abduction [Inoue et al., 2010]

Meta-Interpretive Learning. Learning regular and context-free grammars [Muggleton et al, 2013]

Higher-order Logic Learning. Without background knowledge [Feng and Muggleton, 1992; Lloyd 2003]

Higher-order Datalog. HO-Progol learning [Pahlavi and Muggleton, 2012]

Conclusions and Challenges

- New form of Declarative Machine Learning [De Raedt, 2012]
- H^2_2 is tractable and Turing-complete fragment of High-order Logic
- Knuth-Bendix style ordering guarantees termination of queries
- Beyond classification learning - strategy learning

Challenges

- Generalise beyond Dyadic logic
- Deal with classification noise
- Active learning
- Efficient problem decomposition
- Meaningful invented names and types

Bibliography

- A. Cropper, S.H. Muggleton. Learning efficient logical robot strategies involving composable objects. IJCAI 2015.
- A. Cropper and S.H. Muggleton. Learning higher-order logic programs through abstraction and invention. IJCAI 2016.
- W-Z Dai, S.H. Muggleton, Z-H Zhou. Logical vision: Meta-interpretive learning from real images. ILP 2017.
- S.H. Muggleton, D. Lin, A. Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. Machine Learning, 2015.
- D. Lin, E. Dechter, K. Ellis, J.B. Tenenbaum, S.H. Muggleton. Bias reformulation for one-shot function induction. ECAI 2014.

Draughtsman's assistant demo

Learning from drawings Use simplified version of Postscript language with primitives *draw, turn90, aturn90* in image array.

One-shot learning Each drawing learned from single example using Metarules and Higher-order definitions.

Learn symbols as programs For instance, the letter **L** as a drawing.

Learn numbers as higher-order definitions For instance, the number two (three, four) applied to **L** gives two (three, four) **L**'s.

Incremental learning Larger programs learned by building on previously learned programs.