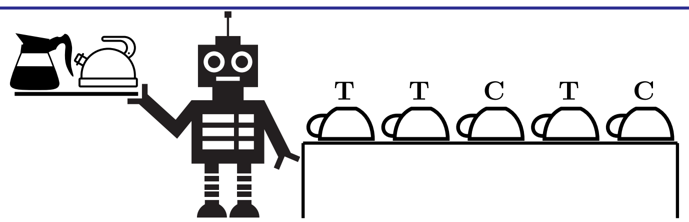
What use is Abstraction in Deep Program Induction?

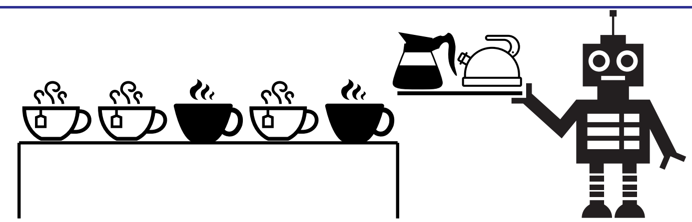
Stephen Muggleton

Department of Computing
Imperial College, London, UK

Robotic waiter



Initial state



Final state

Meta-Interpretive Learning (MLJ, 2015)
Recursive solution

```
f(A,B):-f3(A,B),at_end(B).
f(A,B):-f3(A,C),f(C,B).
f3(A,B):-f2(A,C),move_right(C,B).
f2(A,B):-turn_cup_over(A,C),f1(C,B).
f1(A,B):-wants_tea(A),pour_tea(A,B).
f1(A,B):-wants_coffee(A),pour_coffee(A,B).
```

Meta-Interpretive Learning
Abstraction and Invention

Shorter program

```
f(A,B):-until(A,B,at_end,f3).
f3(A,B):-f2(A,C),move_right(C,B).
f2(A,B):-turn_cup_over(A,C),f1(C,B).
f1(A,B):-ifthenelse(A,B,wants_tea, pour_tea, pour_coffee).
```

Alternation of Abstraction and Invention steps

$\rightarrow$	**Abstract**	$\rightarrow$	**Invent**	$\rightarrow$	**Abstract**
f	until		f3,f2,f1		ifthenelse

Abstraction and Invention - Robot example

Higher-order definition
until(S1,S2,Cond,Do) $\leftarrow$ Cond(S1)
until(S1,S2,Cond,Do) $\leftarrow$ not(Cond(S1)), Do(S1,S2)

Abstraction
f(A,B) $\leftarrow$ until(A,B,at_end,f3)

Invention
f3(A,B) $\leftarrow$ f2(A,C),move_right(C,B)

Metarules

Name	Meta-Rule	Order
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	$P(x, y) \leftarrow Q(x, z), P(z, y)$	$P \succ Q,$ $x \succ z \succ y$
Curry2	$P(x, y) \leftarrow Q(R, x, y)$	$P \succ Q$
HChain	$P(Q, x, y) \leftarrow R(Q, x, z), S(Q, z, y)$	$P \succ R, S \succ R$

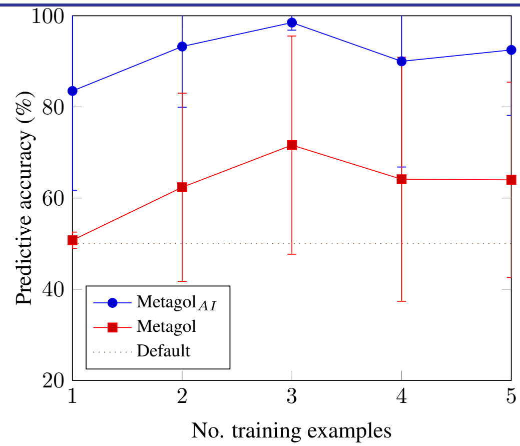
Metagol (ECAI14,IJCAI15,IJCAI16)

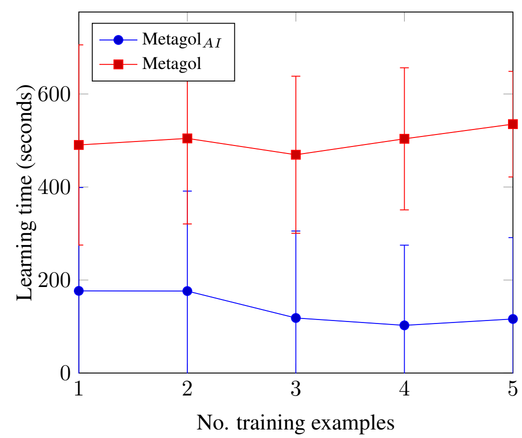
```
prove([],H,H).
prove([Atom|Atoms],H1,H2):-
    prove_aux(Atom,H1,H3),
    prove(Atoms,H3,H2).
prove_aux(Atom,H1,H2):-
    metarule(Name,Subs,(Atom :- Body)),
    new_metasub(H1,sub(Name,Subs)),
    abduce(H1,H3,sub(Name,Subs)),
    prove(Body,H3,H2).
```

Results - Waiter (IJCAI16)

Proposition 1: Sample complexity proportional to program size

Predictive accuracy	Learning time





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.

Conclusions and Further Work

- General method of introducing higher-order constructs such as while, until, ifthenelse, map
- Leads to reduction in program size
- Sample complexity reduction and search space reduction
- Further work - non-functional constructs such as closure to learn

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

- Applications in planning, vision and NLP

Bibliography

- A. Cropper and S.H. Muggleton. Learning higher-order logic programs through abstraction and invention. In Proceedings of the 25th International Joint Conference Artificial Intelligence (IJCAI 2016), pages 1418-1424. IJCAI, 2016.
- A. Cropper, S.H. Muggleton. Learning efficient logical robot strategies involving composable objects. IJCAI 2015.
- D. Lin, E. Dechter, K. Ellis, J.B. Tenenbaum, S.H. Muggleton. Bias reformulation for one-shot function induction. ECAI 2014.