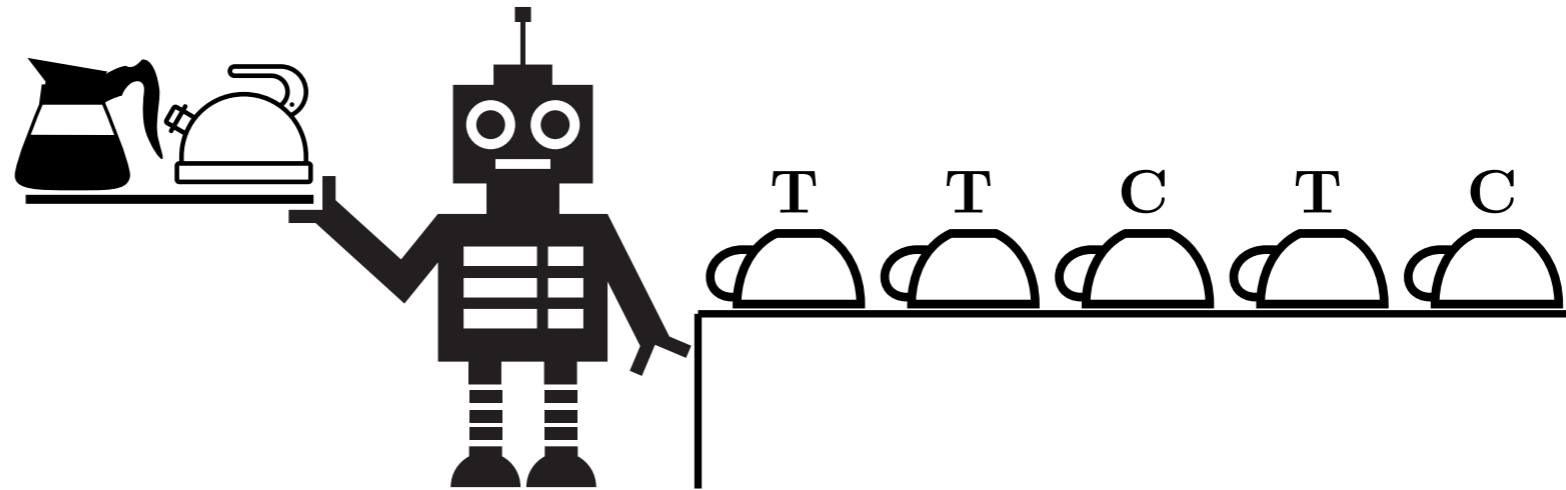


Learning higher-order logic programs through abstraction and invention

[IJCAI16]

Initial state:



Final state:



First-order recursive solution [MLJ2015]

$f(A,B):-f3(A,B),at_end(B).$

$f(A,B):-f3(A,C),f(C,B).$

$f3(A,B):-f2(A,C),move_right(C,B).$

$f2(A,B):-turn_cup_over(A,C),f1(C,B).$

$f1(A,B):-wants_tea(A),pour_tea(A,B).$

$f1(A,B):-wants_coffee(A),pour_coffee(A,B).$

Higher-order solution

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).

Abstraction and invention - robot example

Higher-order definition

$\text{until}(S1, S2, \text{Cond}, \text{Do}) \leftarrow \text{Cond}(S1)$

$\text{until}(S1, S2, \text{Cond}, \text{Do}) \leftarrow \text{Do}(S1, S3), \text{until}(S3, S2).$

Abstraction

$f(A, B) :- \text{until}(A, B, \text{at_end}, f3).$

Invention

$f3(A, B) :- f2(A, C), \text{move_right}(C, B).$

HO predicate	Reductio n
until	1
ifthenesle	1
map	1
filter	2

Previous Metagol (ECAI14,IJCAI15)

```
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).
```

New Metagol with interpreted BK

```
prove([],H,H).
```

```
prove([Atom|Atoms],H1,H2):-
```

```
    prove aux(Atom,H1,H3),
```

```
    prove(Atom,H3,H2).
```

```
prove_aux(Atom,H1,H2):-
```

```
    background((Atom:-Body)),
```

```
    prove(Body,H1,H2).
```

```
prove aux(Atom,H1,H2):-
```

```
    metarule(Name,Subs,(Atom :- Body)),
```

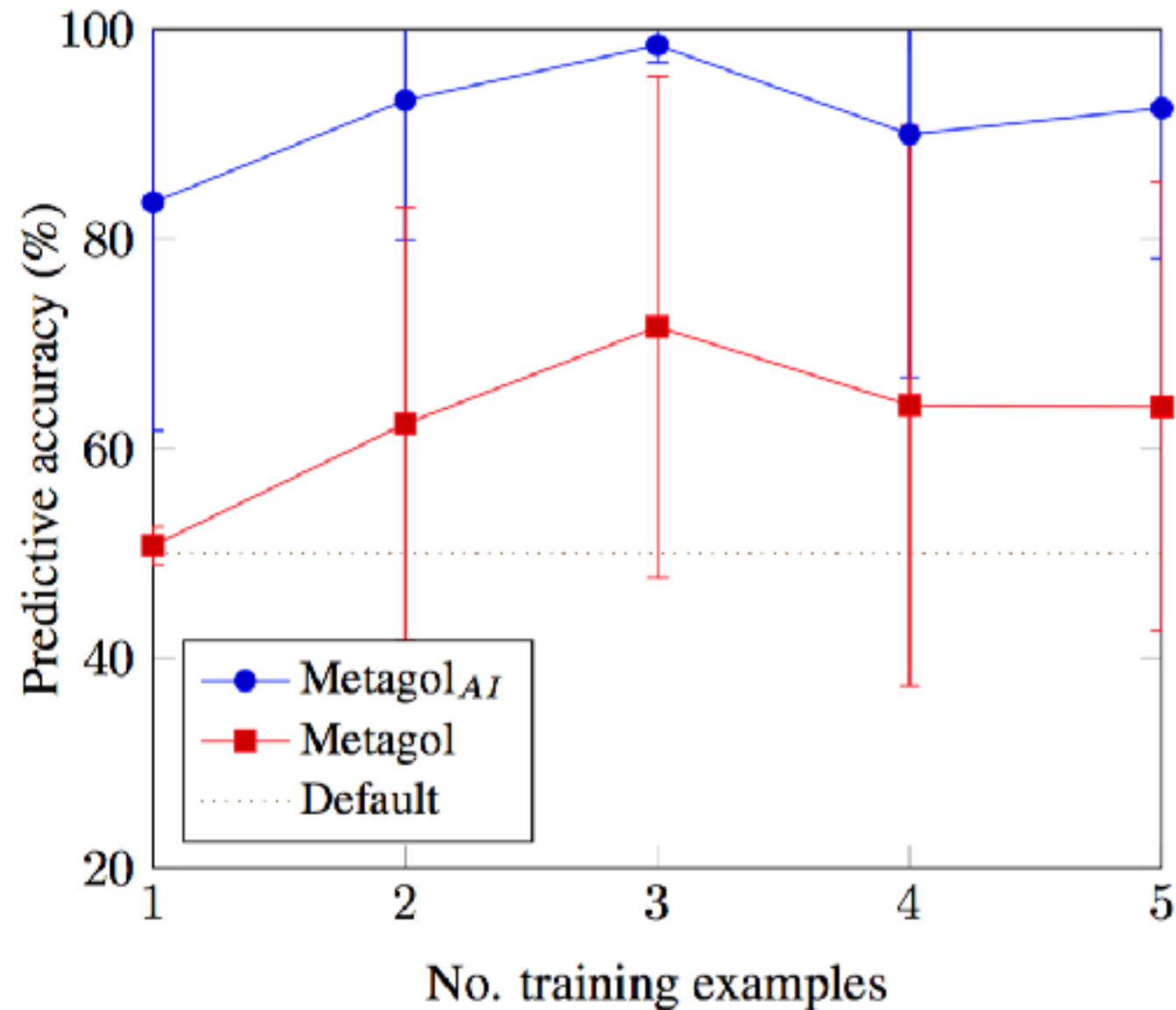
```
    new metasub(H1,sub(Name,Subs)),
```

```
    abduce(H1,H3,sub(Name,Subs)),
```

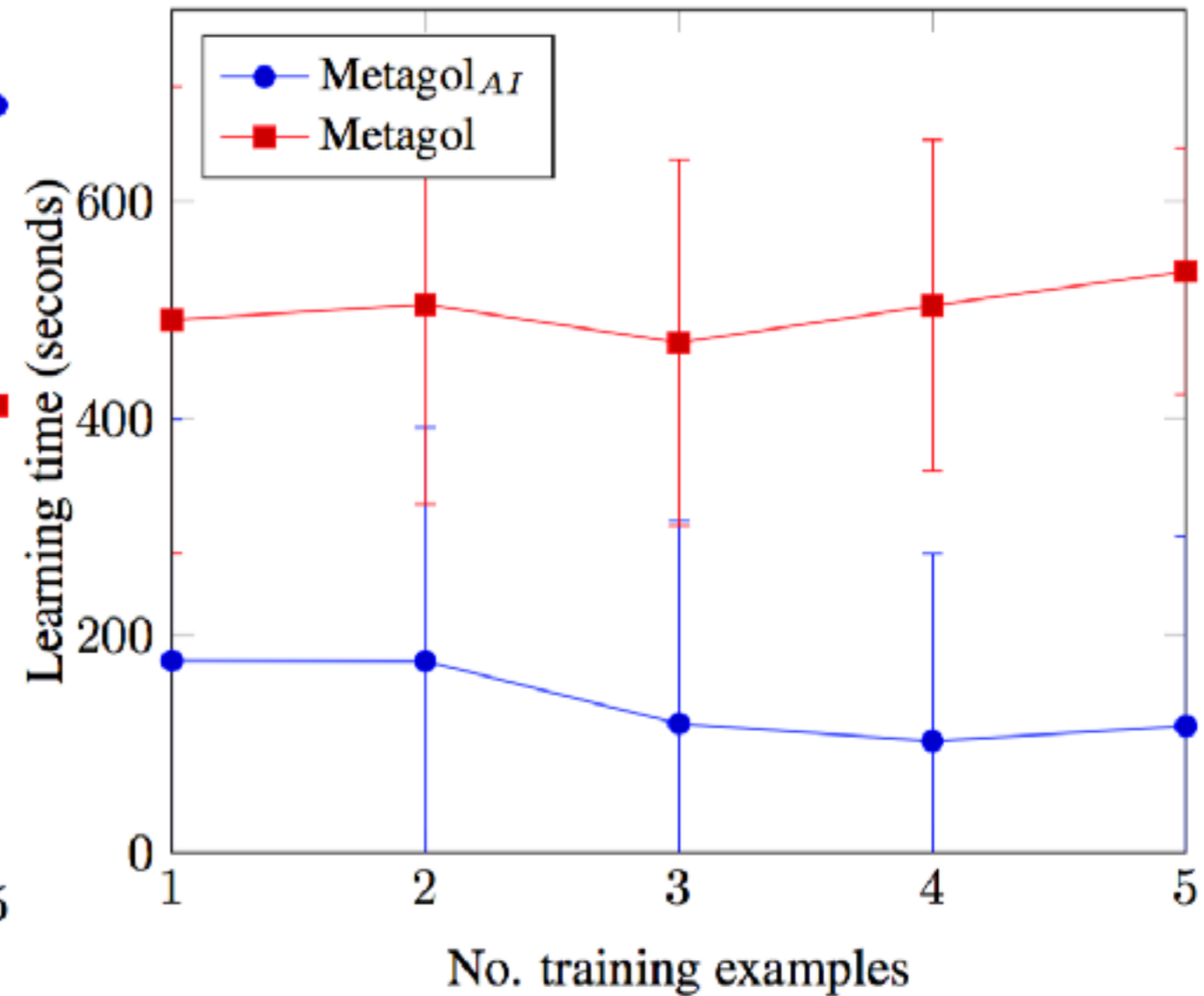
```
    prove(Body,H3,H2).
```


Waiter results

Proposition 1: Sample complexity proportional to program size

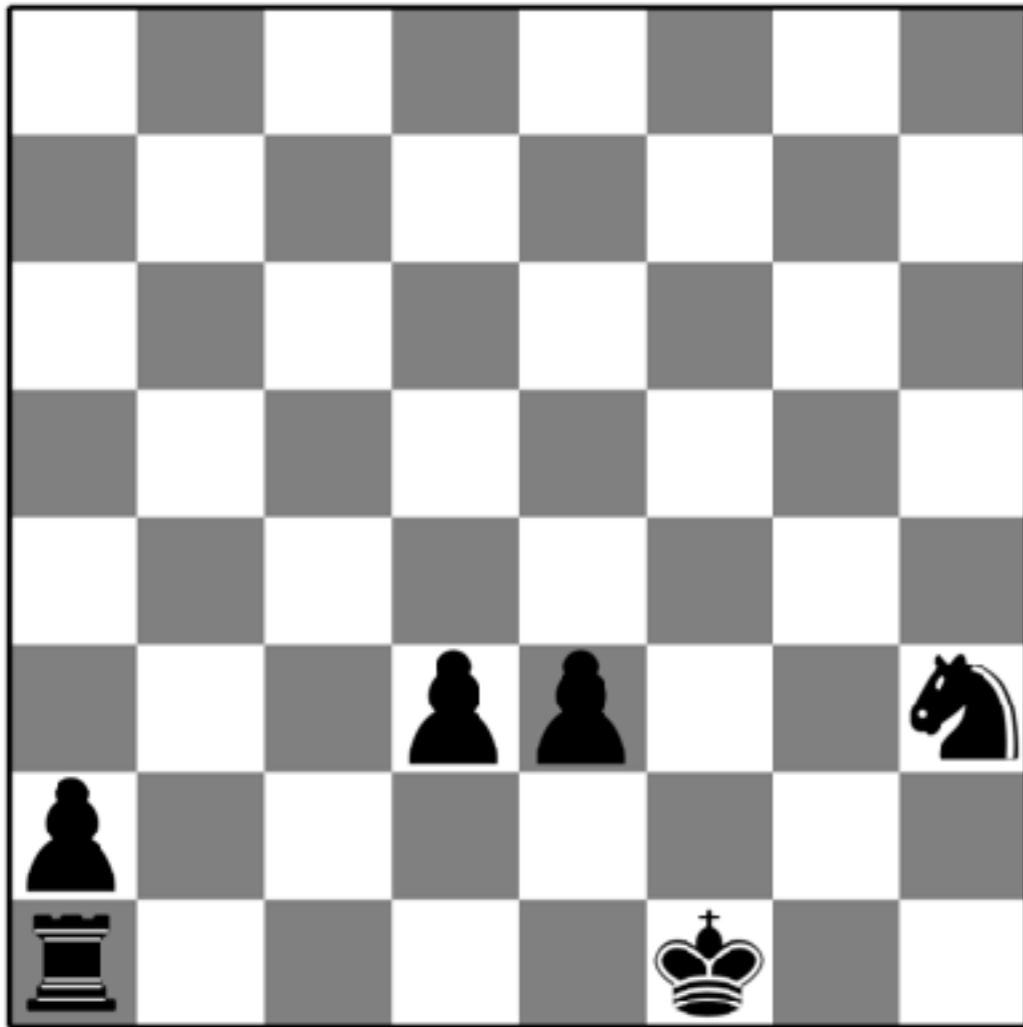


(a) Predictive accuracies

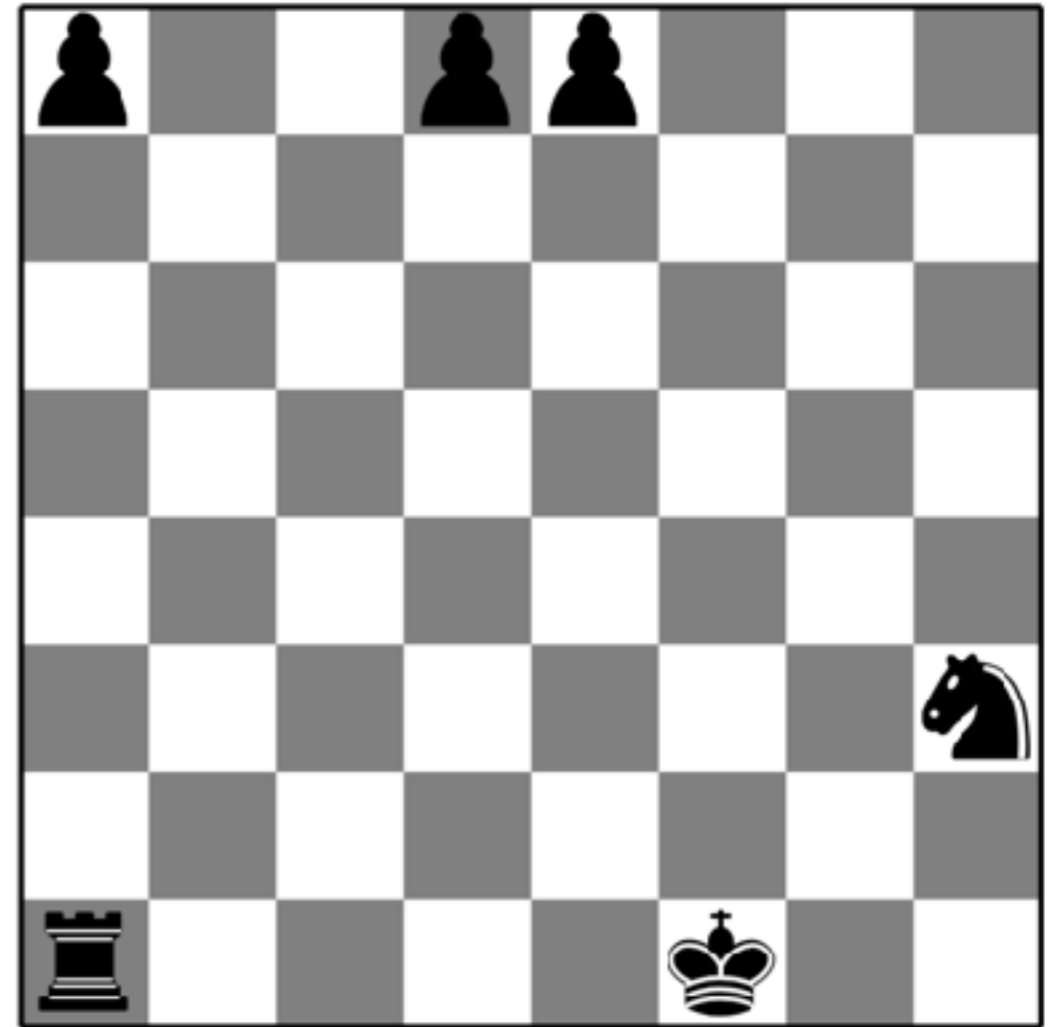


(b) Learning times

Chess experiment

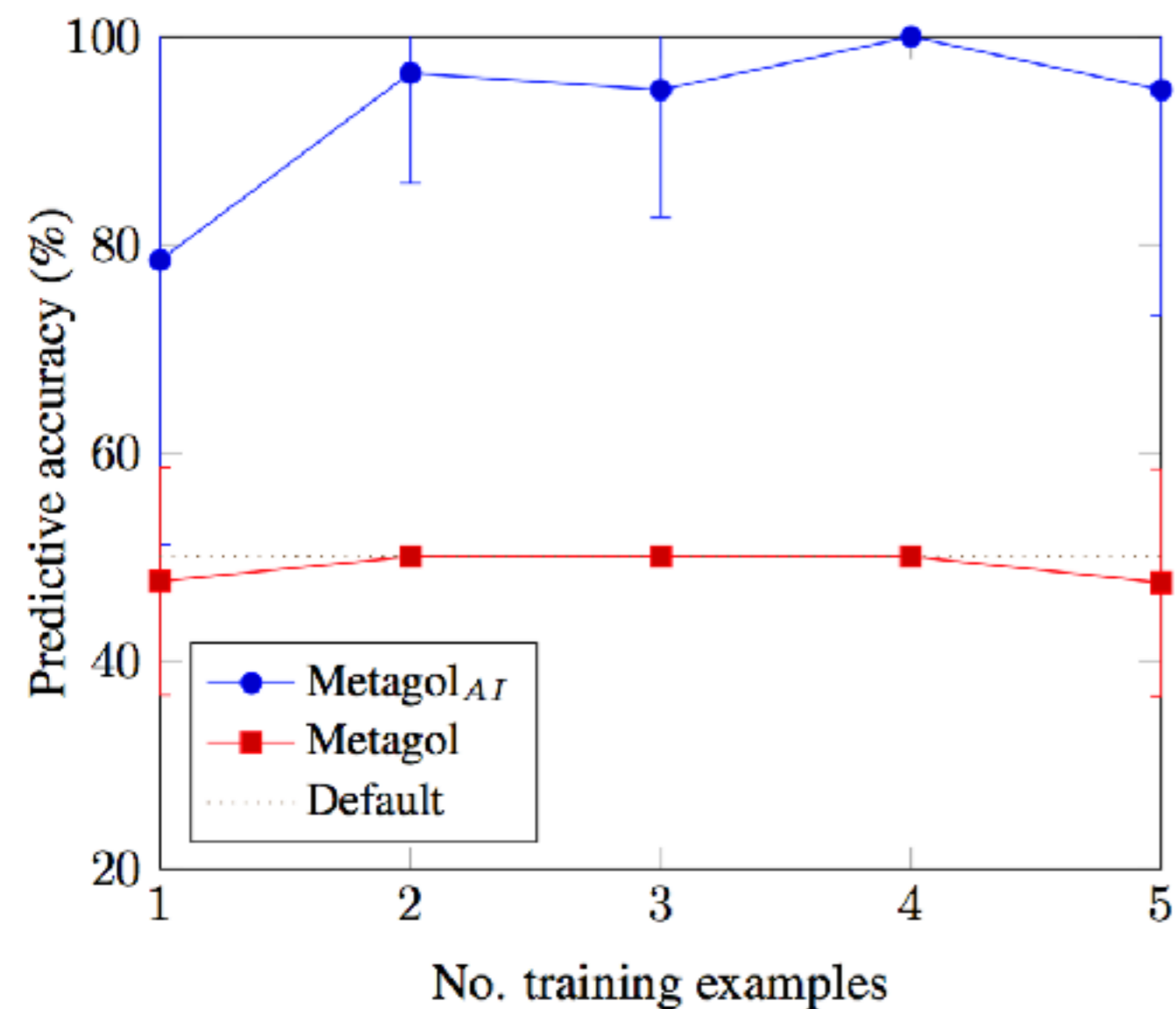


(a) Initial state

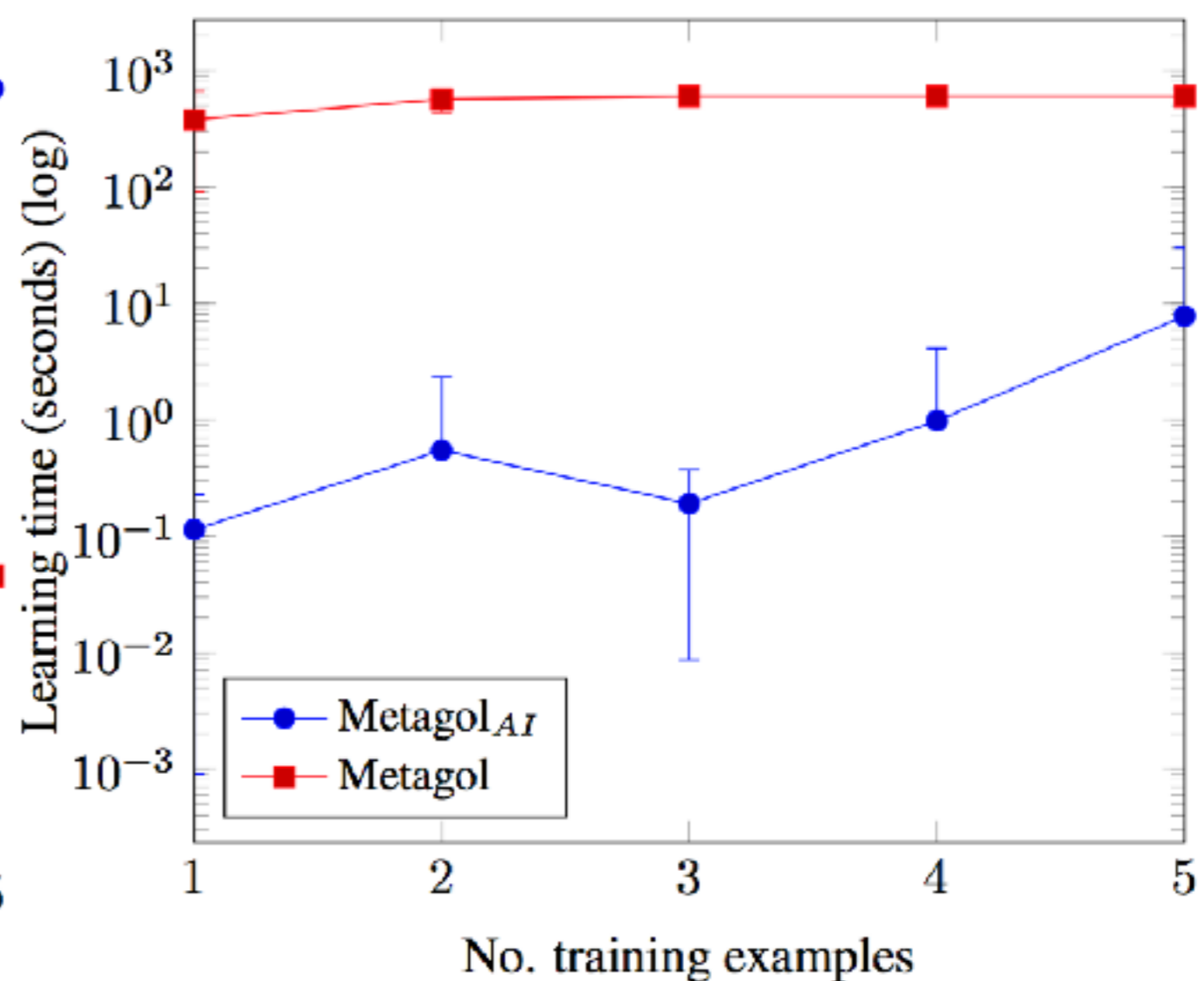


(b) Final state

Chess results



(a) Predictive accuracies



(b) Learning times

Programming example 1

Input	Output
[[i,j,c,a,i],[2,0,1,6]]	[[i,j,c,a],[2,0,1]]
[[1,1],[a,a],[x,x]]	[[1],[a],[x]]
[[1,2,3,4,5],[1,2,3,4,5]]	[[1,2,3,4],[1,2,3,4]]
[[1,2],[1,2,3],[1,2,3,4],[1,2,3,4,5]]	[[1],[1,2],[1,2,3],[1,2,3,4]]

f(A,B):-map(A,B,f3).

f3(A,B):-f2(A,C),**f1**(C,B).

f2(A,B):-**f1**(A,C),tail(C,B).

f1(A,B):-reduceback(A,B,concat).

f3 = droplast

f1 = reverse

Programming example 2

Input	Output
[[i,j,c,a,i],[2,0,1,6]]	[[i,j,c,a]]
[[1,1],[a,a],[x,x]]	[[1],[a]]
[[1,2,3,4,5],[1,2,3,4,5]]	[[1,2,3,4]]
[[1,2],[1,2,3],[1,2,3,4],[1,2,3,4,5]]	[[1],[1,2],[1,2,3]]

$f(A,B):-f4(A,C),\mathbf{f3}(C,B).$

$f4(A,B):-map(A,B,\mathbf{f3}).$

$\mathbf{f3}(A,B):-f2(A,C),\mathbf{f1}(C,B).$

$f2(A,B):-\mathbf{f1}(A,C),tail(C,B).$

$\mathbf{f1}(A,B):-reduceback(A,B,concat).$

$\mathbf{f4} = \text{droplasts}$



Conclusions

- 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

Future work

- Invent the higher-order abstractions
- Applications in planning, vision and NLP

Bibliography

<https://github.com/metagol>

- A. Cropper, S.H. Muggleton. Learning efficient logical robot strategies involving composable objects. IJCAI 2015.
- 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.