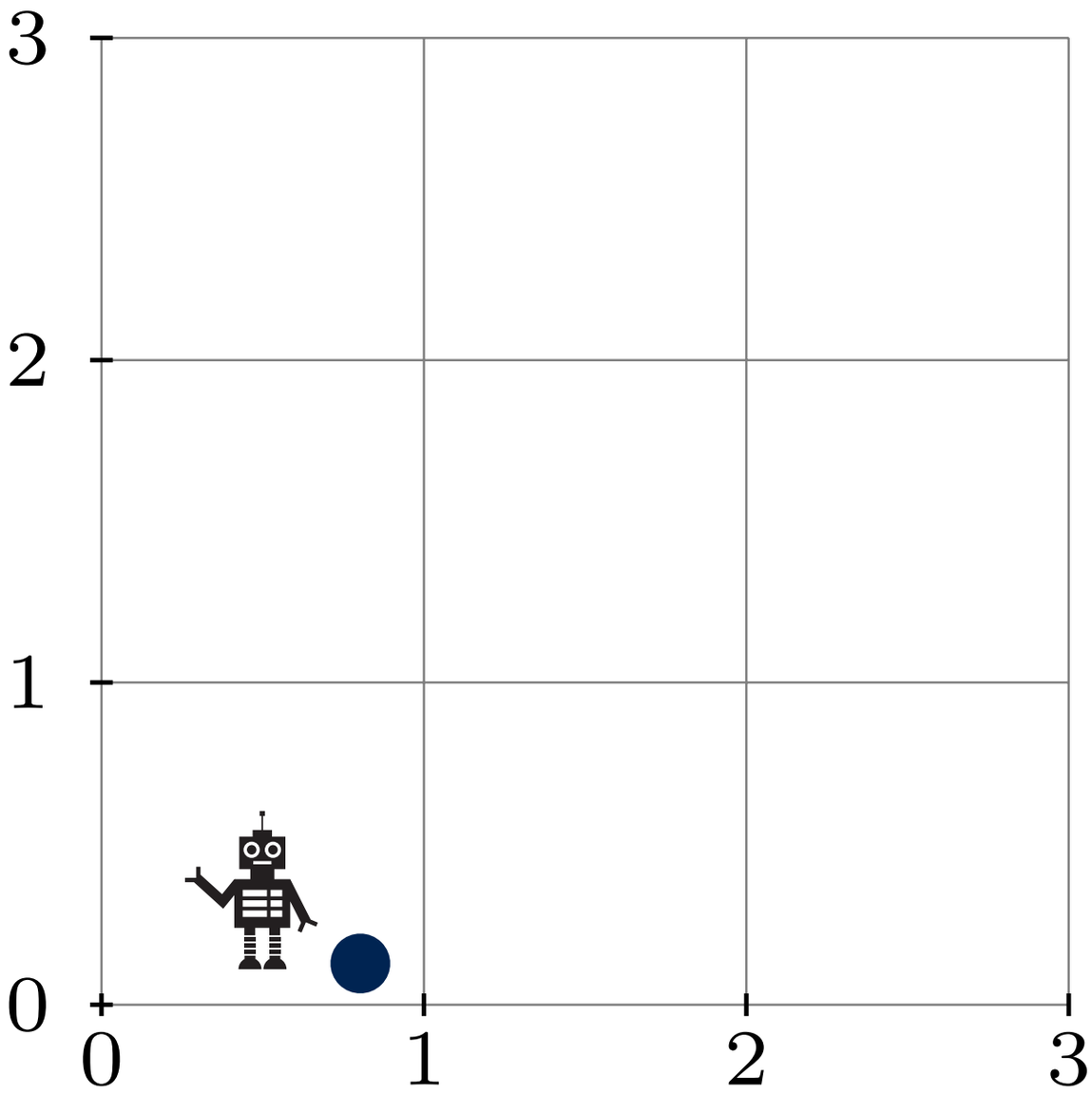
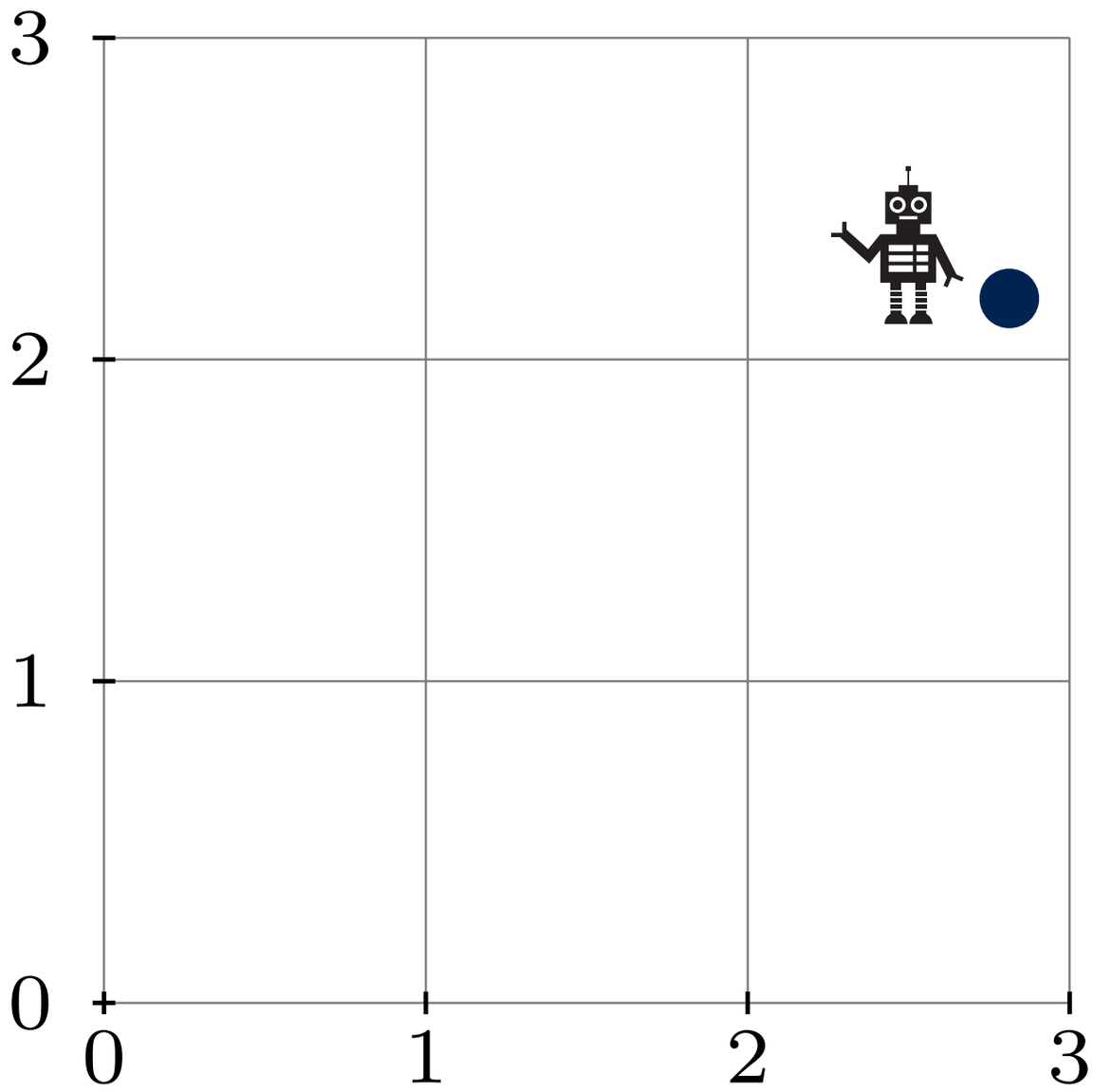
Learning efficient logic programs

Andrew Cropper and Stephen H. Muggleton

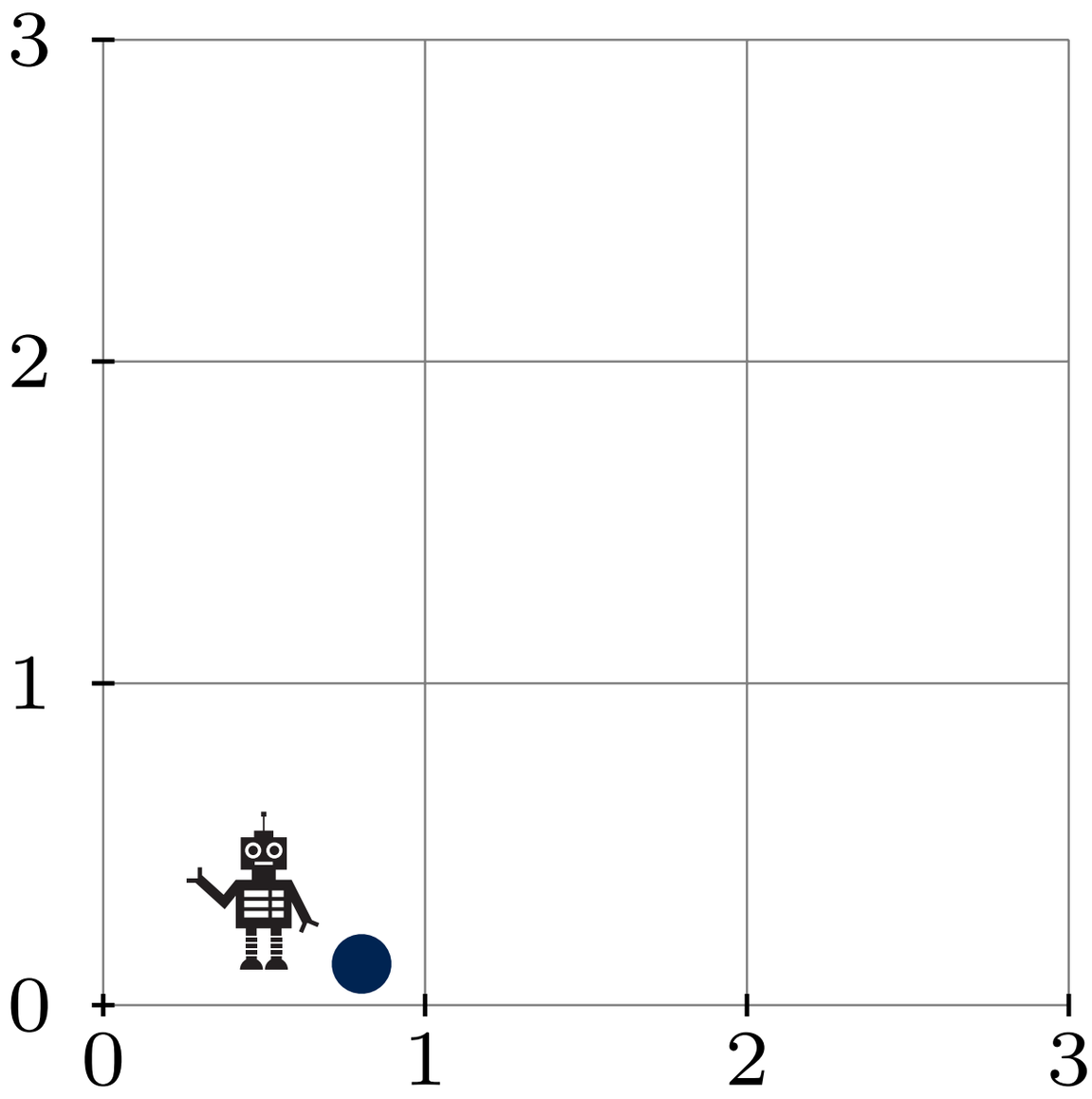
Initial state



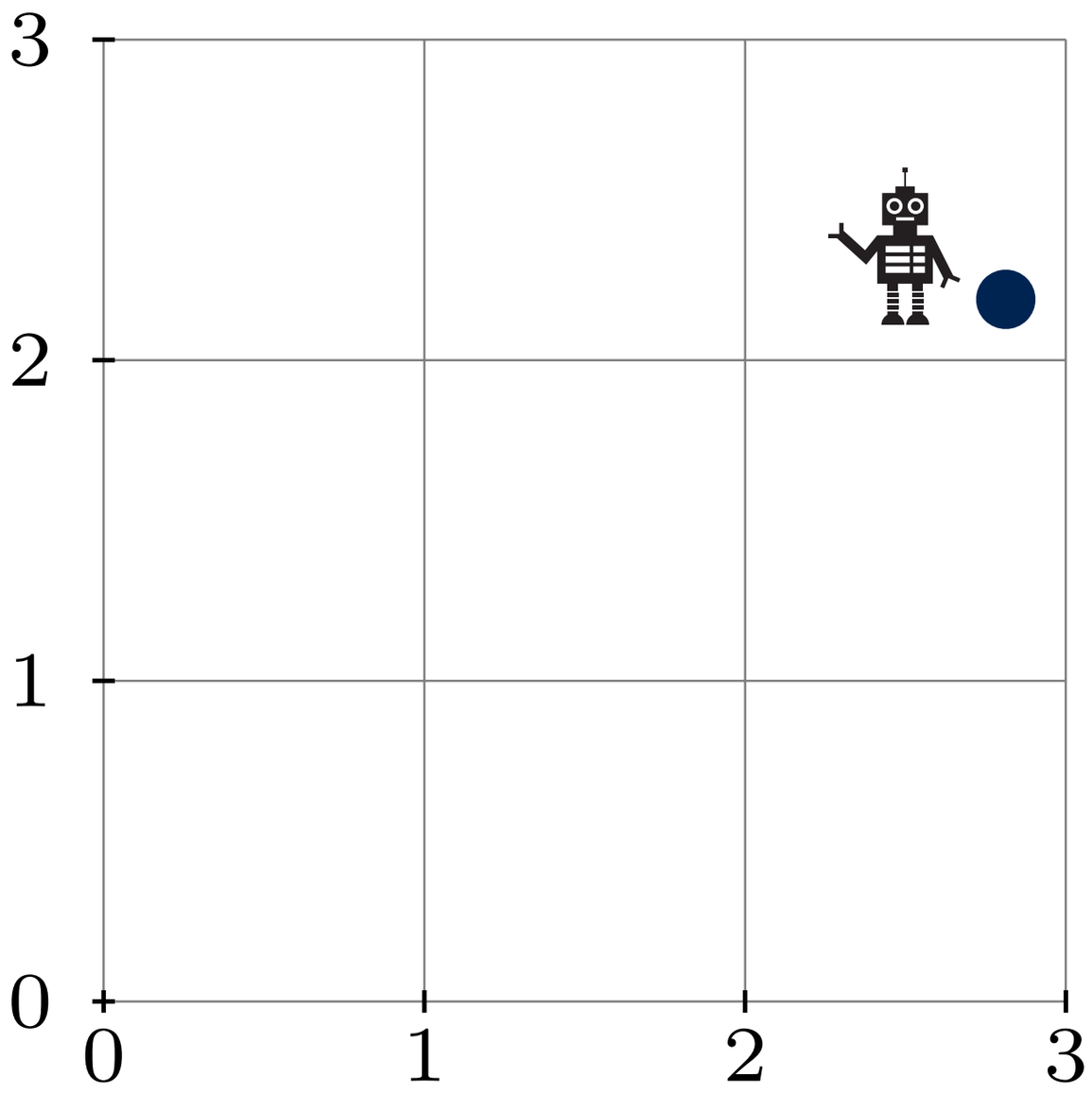
Final state



Initial state



Final state



E+ f([pos(r,1/1),pos(b,1/1)],[pos(r,3/3),pos(b,3/3)])

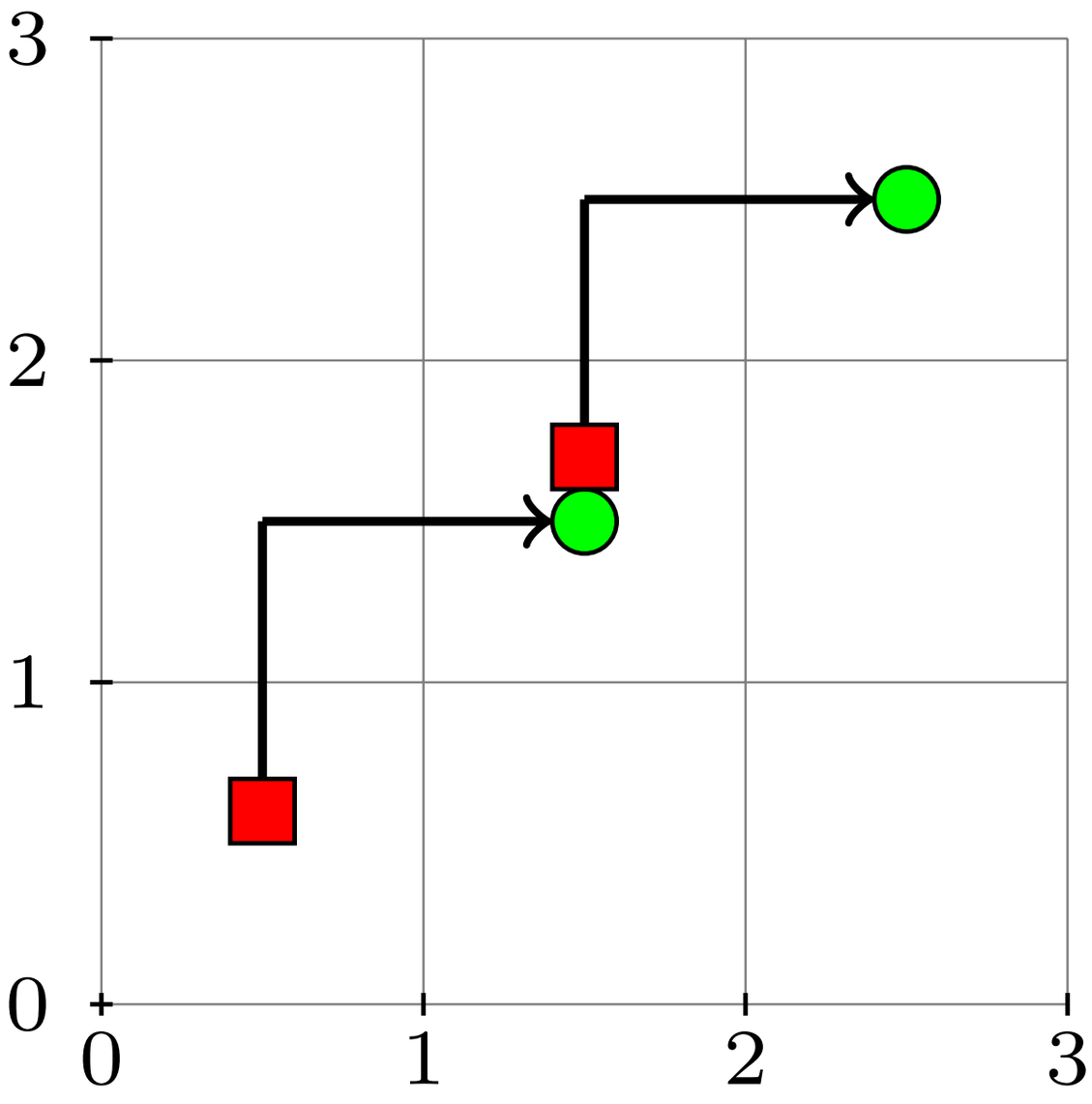
B north/2, south/2, east/2, west/2, grab/2, drop/2

```
f(X,Y):-f3(X,Z),f3(Z,Y).
f3(X,Y):-f2(X,Z),drop(Z,Y).
f2(X,Y):-grab(X,Z),f1(Z,Y).
f1(X,Y):-north(X,Z),east(Z,Y).
```

```
f(X,Y):-f3(X,Z),drop(Z,Y).
f3(X,Y):-grab(X,Z),f2(Z,Y).
f2(X,Y):-f1(X,Z),f1(Z,Y).
f1(X,Y):-north(X,Z),east(Z,Y).
```

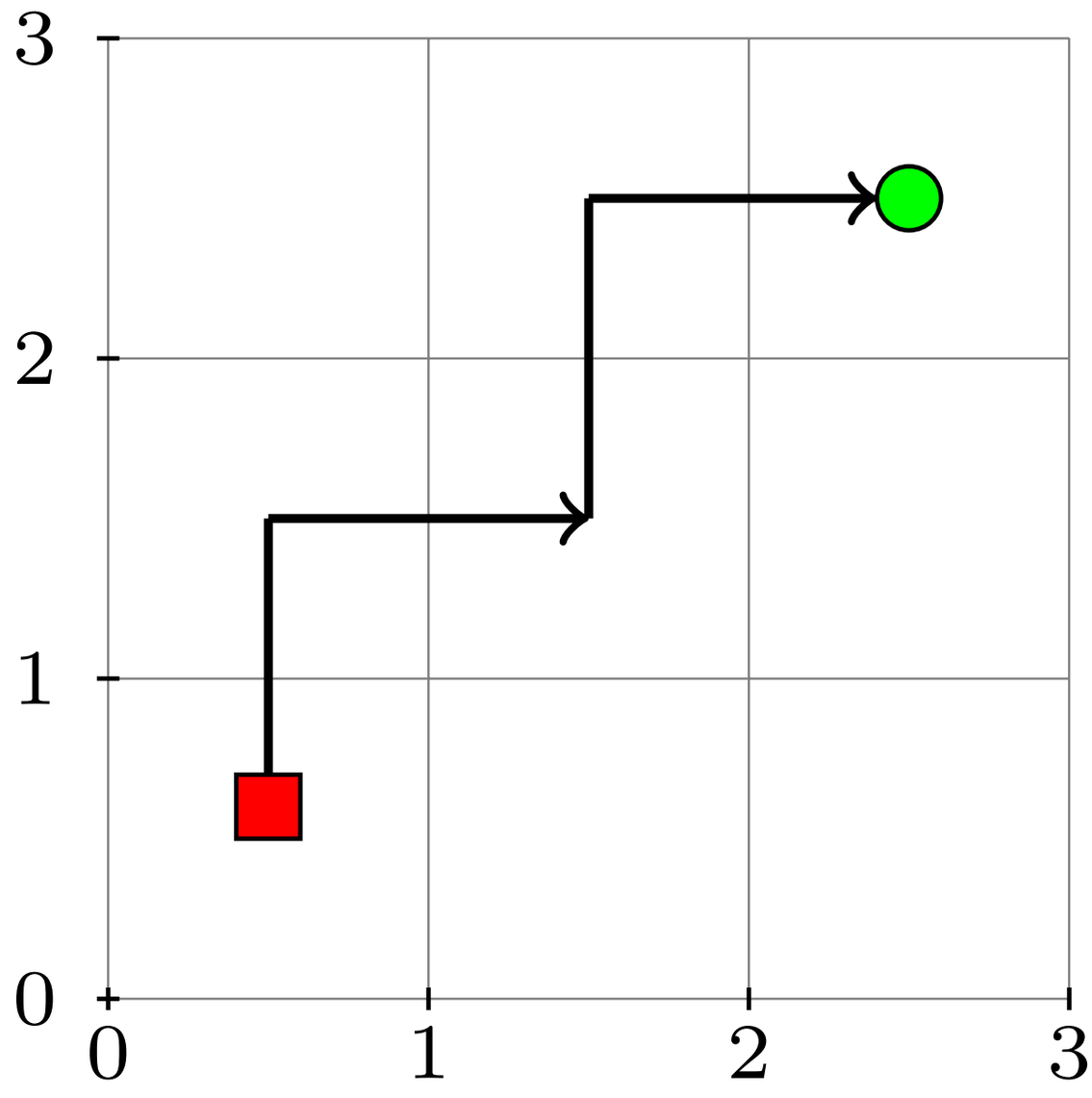
grab drop

Inefficient




```
f(X,Y):-f3(X,Z),f3(Z,Y).
f3(X,Y):-f2(X,Z),drop(Z,Y).
f2(X,Y):-grab(X,Z),f1(Z,Y).
f1(X,Y):-north(X,Z),east(Z,Y).
```

Efficient




```
f(X,Y):-f3(X,Z),drop(Z,Y).
f3(X,Y):-grab(X,Z),f2(Z,Y).
f2(X,Y):-f1(X,Z),f1(Z,Y).
f1(X,Y):-north(X,Z),east(Z,Y).
```

MetagolO / Metaopt

1. Find a program **P** with minimal textual complexity
2. Repeat until convergence:
   A. Find program **P′** such that cost(**P′**) < cost(**P**)
   B. Set **P** to **P′**

Converges on minimal worst-case program

Action	Cost
grab/2	2
drop/2	2
north/2	1
south/2	1
east/2	1
west/2	1

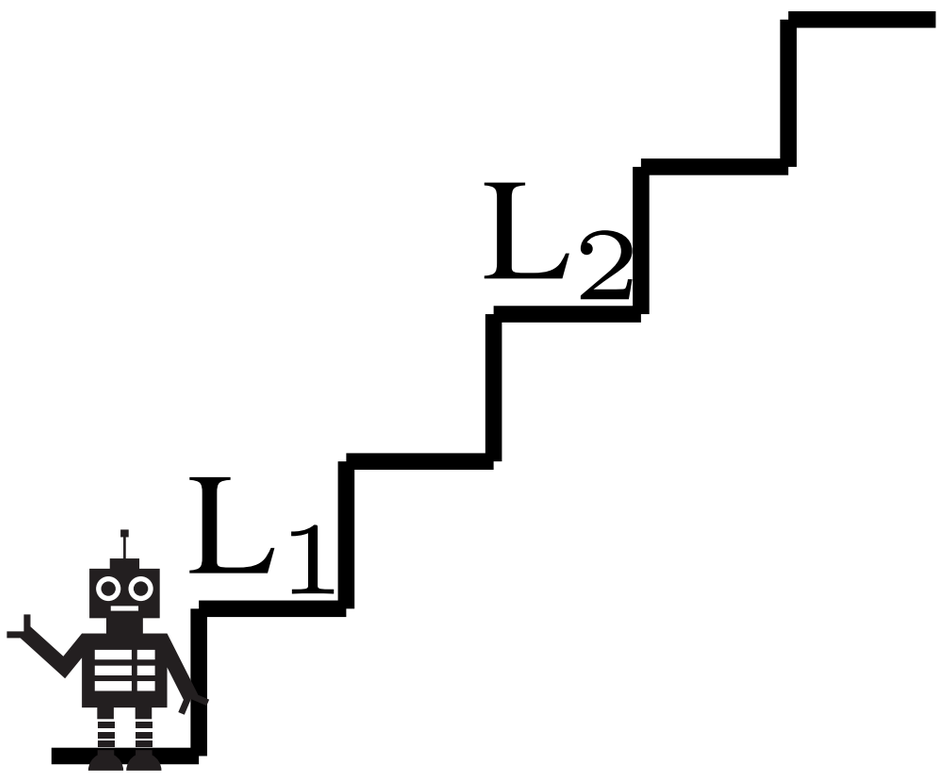
Metagol

```
f(X,Y):-f3(X,Z),f3(Z,Y).
f3(X,Y):-f2(X,Z),drop(Z,Y).
f2(X,Y):-grab(X,Z),f1(Z,Y).
f1(X,Y):-north(X,Z),east(Z,Y)
```

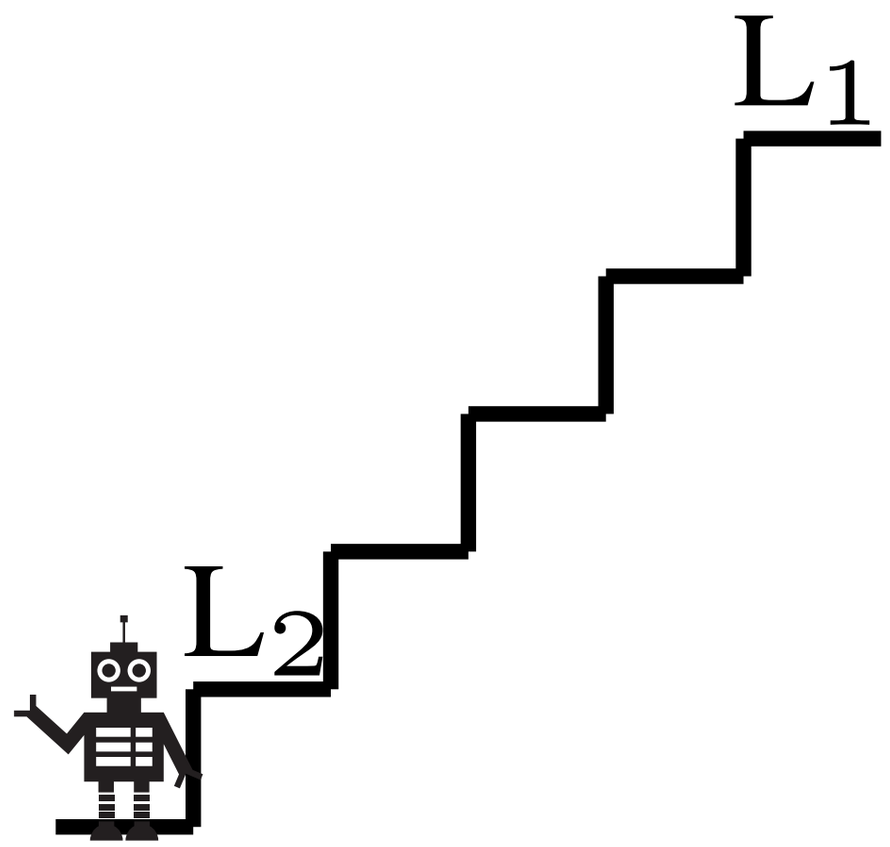
Metaopt

```
f(X,Y):-f3(X,Z),drop(Z,Y).
f3(X,Y):-grab(X,Z),f2(Z,Y).
f2(X,Y):-f1(X,Z),f1(Z,Y).
f1(X,Y):-north(X,Z),east(Z,Y)
```

Robot postman [IJCAI15]



Initial state



Final state

Actions: go_to_bottom/2, go_to_top/2, find_next_sender/2, find_next_recipient/2, take_letter/2, give_letter/2, bag_letter/2

Robot postman [IJCAI15]

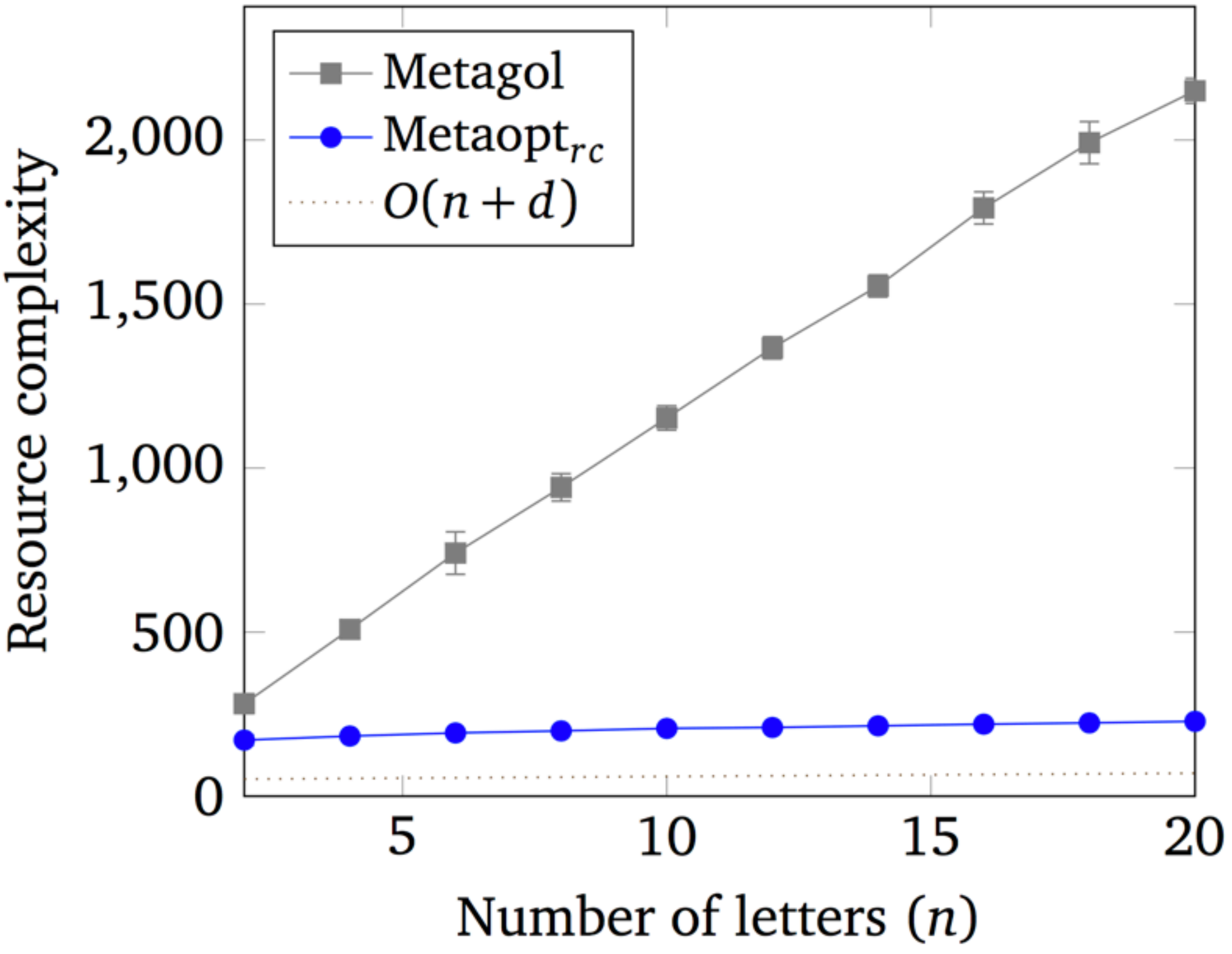
Metagol

```
f(A,B):-f2(A,C),f(C,B).
f(A,B):-f1(A,C),go_start(C,B).
f2(A,B):-f1(A,C),go_start(C,B).
f1(A,B):-find_next_sender(A,C),take_letter(C,B).
```

Metaopt

```
f(A,B):-f2(A,C),f2(C,B).
f2(A,B):-f1(A,C),f2(C,B).
f2(A,B):-f1(A,C),go_start(C,B).
f1(A,B):-find_next_sender(A,C),bag_letter(C,B).
```

Robot postman [IJCAI15]



Robot sorter [IJCAI15]

Input	Output
[9,13,1,8,4]	[1,4,8,9,13]
[1,18,20,6,15,5]	[1,5,6,15,18,20]
[12,16,18,6,15,3,5]	???
[16,1,4,12,3,18,2,14]	???
[12,17,5,13,6,4,14,2,15]	???

Robot sorter [IJCAI15]

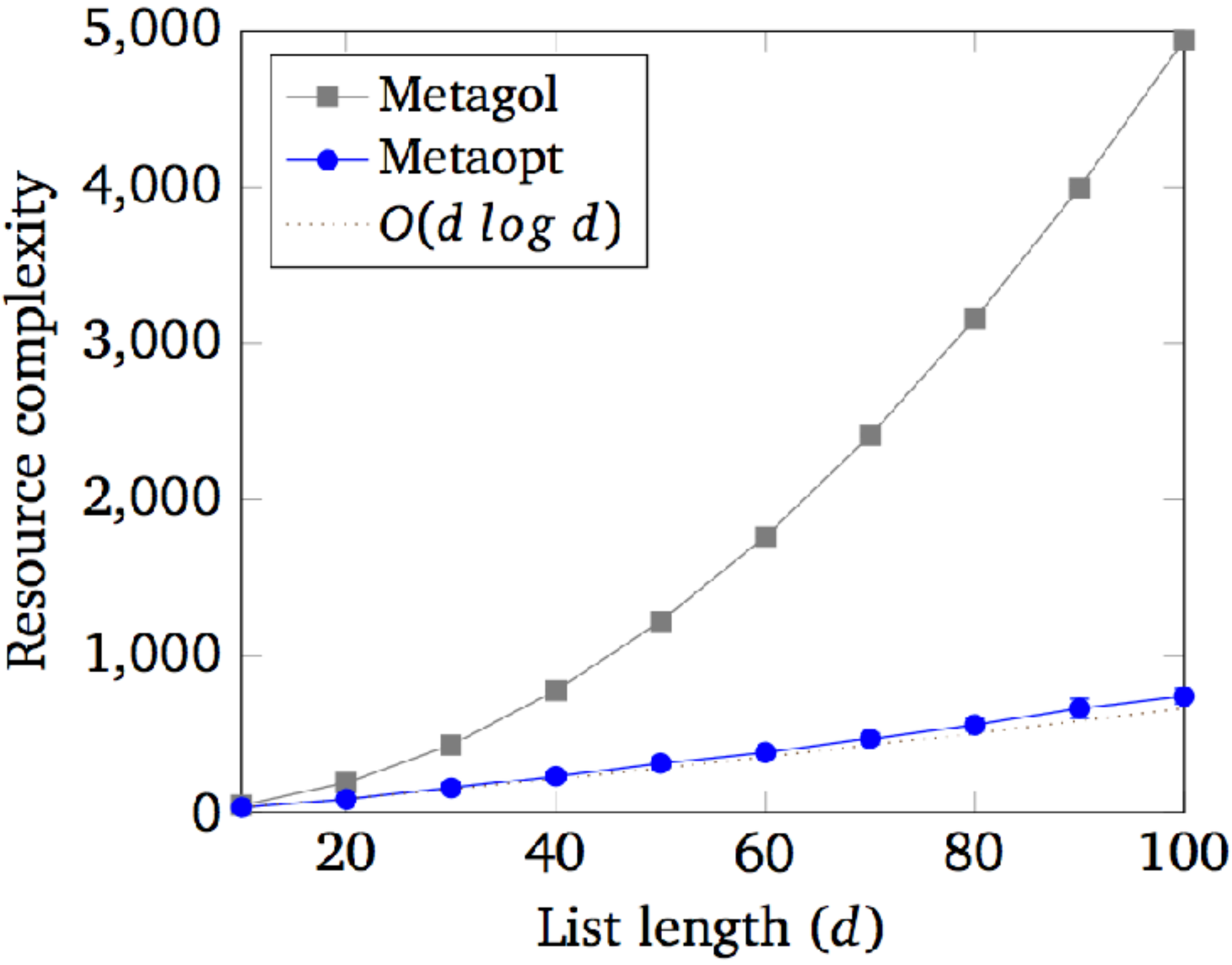
Metagol

```
f(A,B):-f1(A,C),f(C,B).
f(A,B):-f1(A,C),go_start(C,B).
f1(A,B):-comp_adjacent(A,C),f(C,B).
f1(A,B):-decrement_end(A,C),go_start(C,B).
```

Metaopt

```
f(A,B):-f1(A,C),f(C,B).
f(A,B):-decrement_end(A,C),combine(C,B).
f1(A,B):-pick_up_left(A,C),split(C,B).
f1(A,B):-combine(A,C),go_start(C,B).
```

Robot sorter [IJCAI15]



Find duplicate

Input	Output
[m,a,c,h,i,n,e],	e
[l,e,a,r,n,i,n,g]	g
[a,l,g,o,r,i,t,h,m]	???

Find duplicate

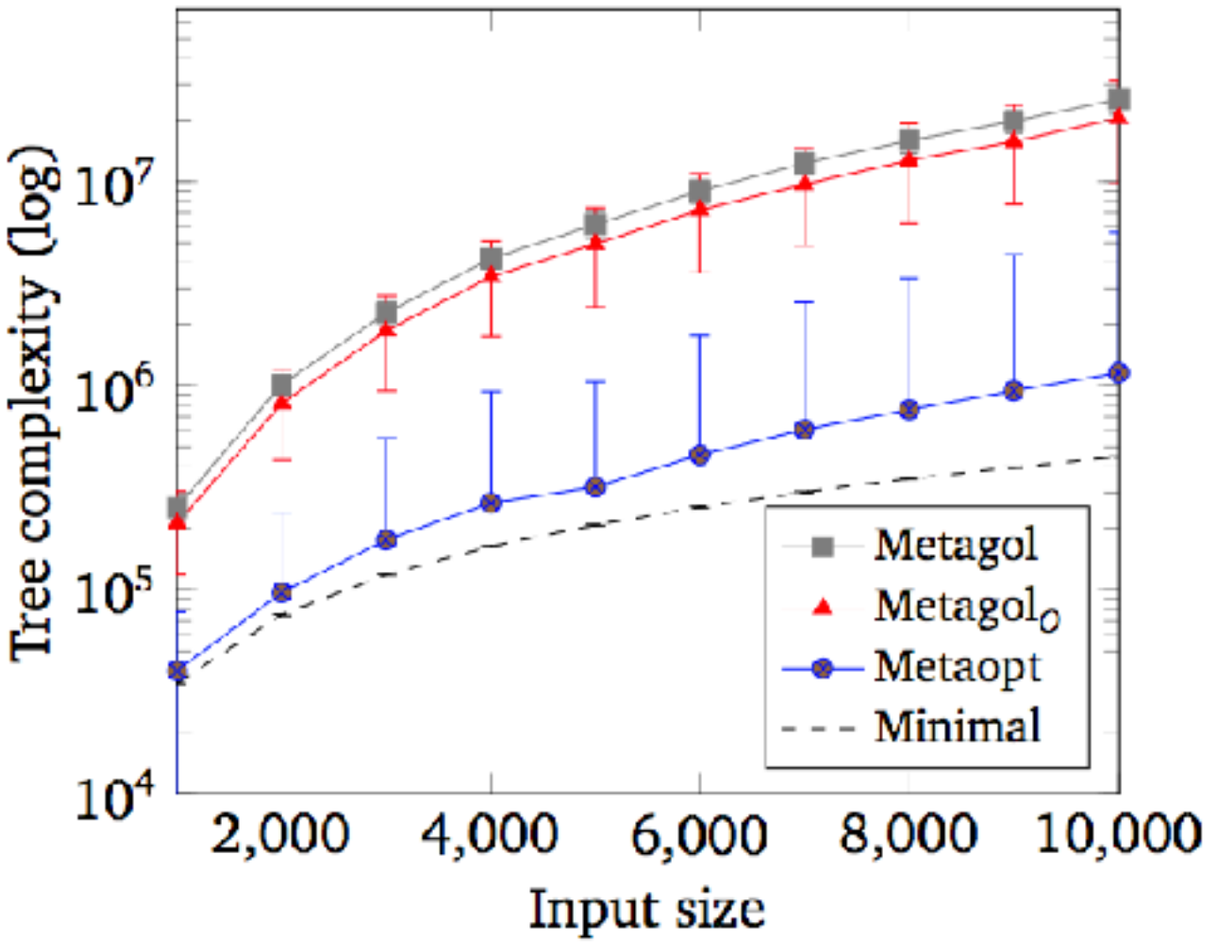
Metagol

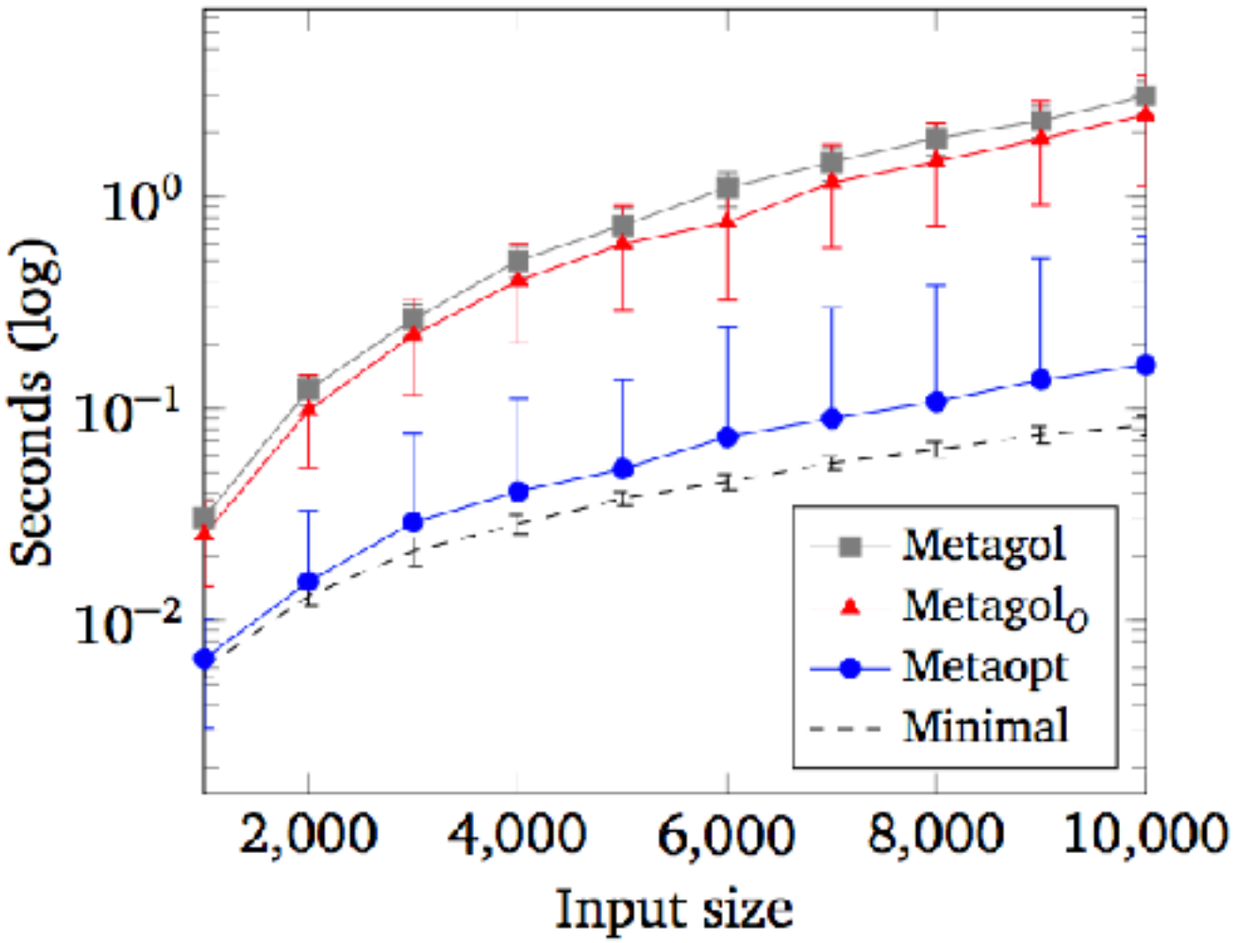
```
f(A,B):-head(A,C),f1(C,B).
f(A,B):-tail(A,C),f(C,B).
f1(A,B):-tail(A,C),member(B,C).
```

Metaopt

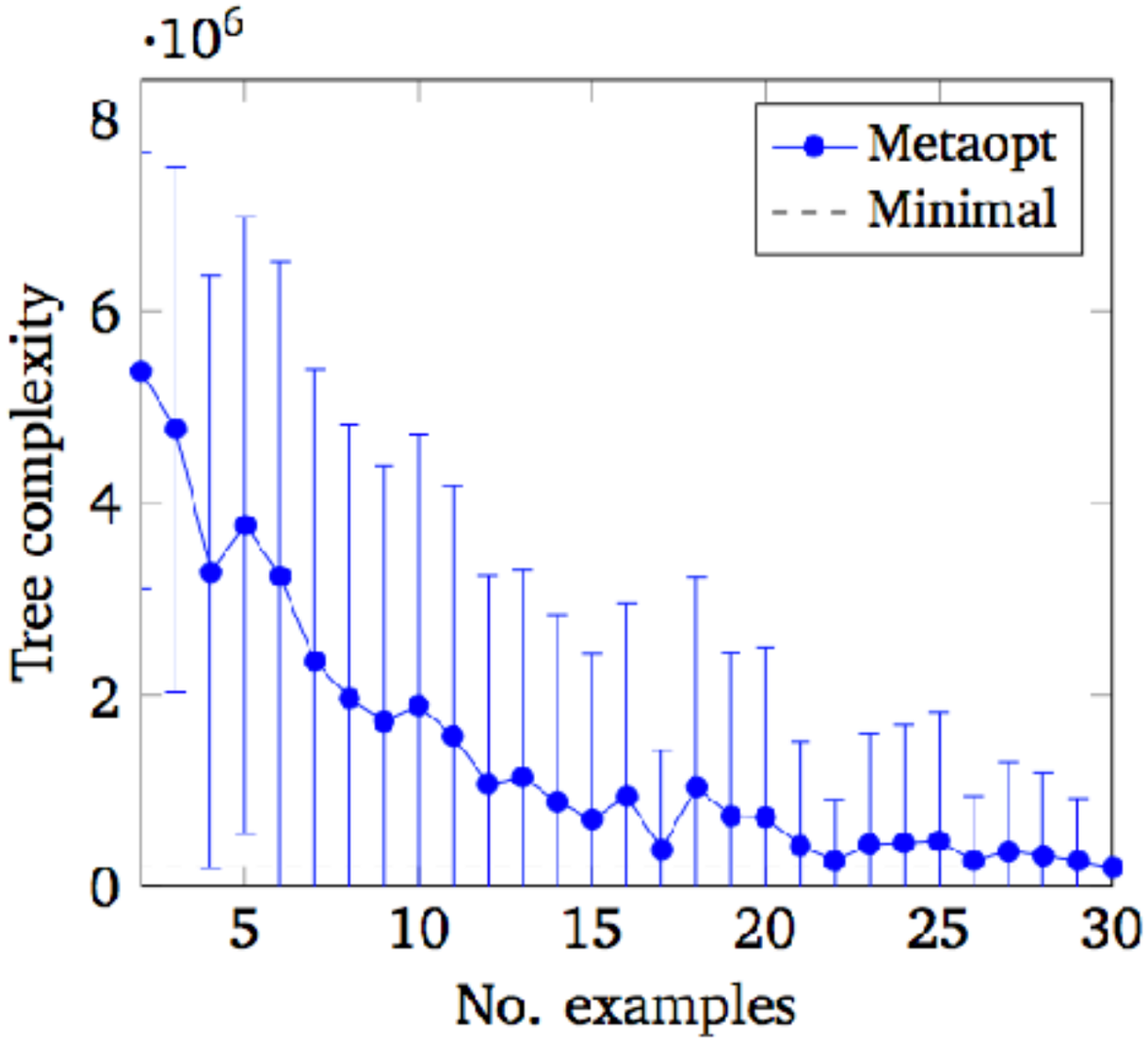
```
f(A,B):-msort(A,C),f1(C,B).
f1(A,B):-head(A,C),f2(C,B).
f1(A,B):-tail(A,C),f1(C,B).
f2(A,B):-tail(A,C),head(C,B).
```

Find duplicate

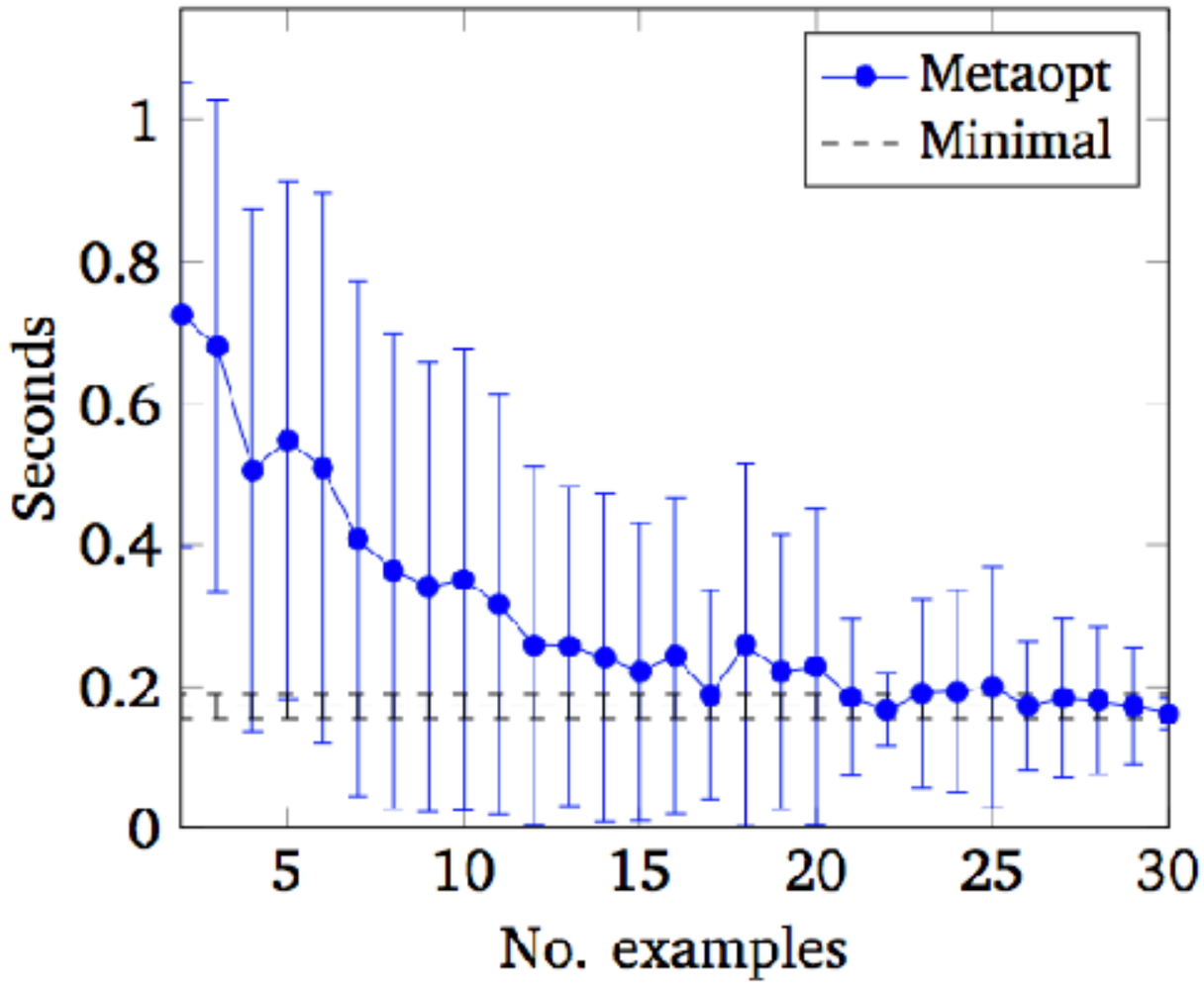




Find duplicate convergence



(a) Tree complexities



(b) Program runtimes

String transformations

Input	Output
My name is John.	John
My name is Bill.	Bill
My name is Josh.	Josh
My name is Albert.	Albert
My name is Richard.	Richard

String transformations

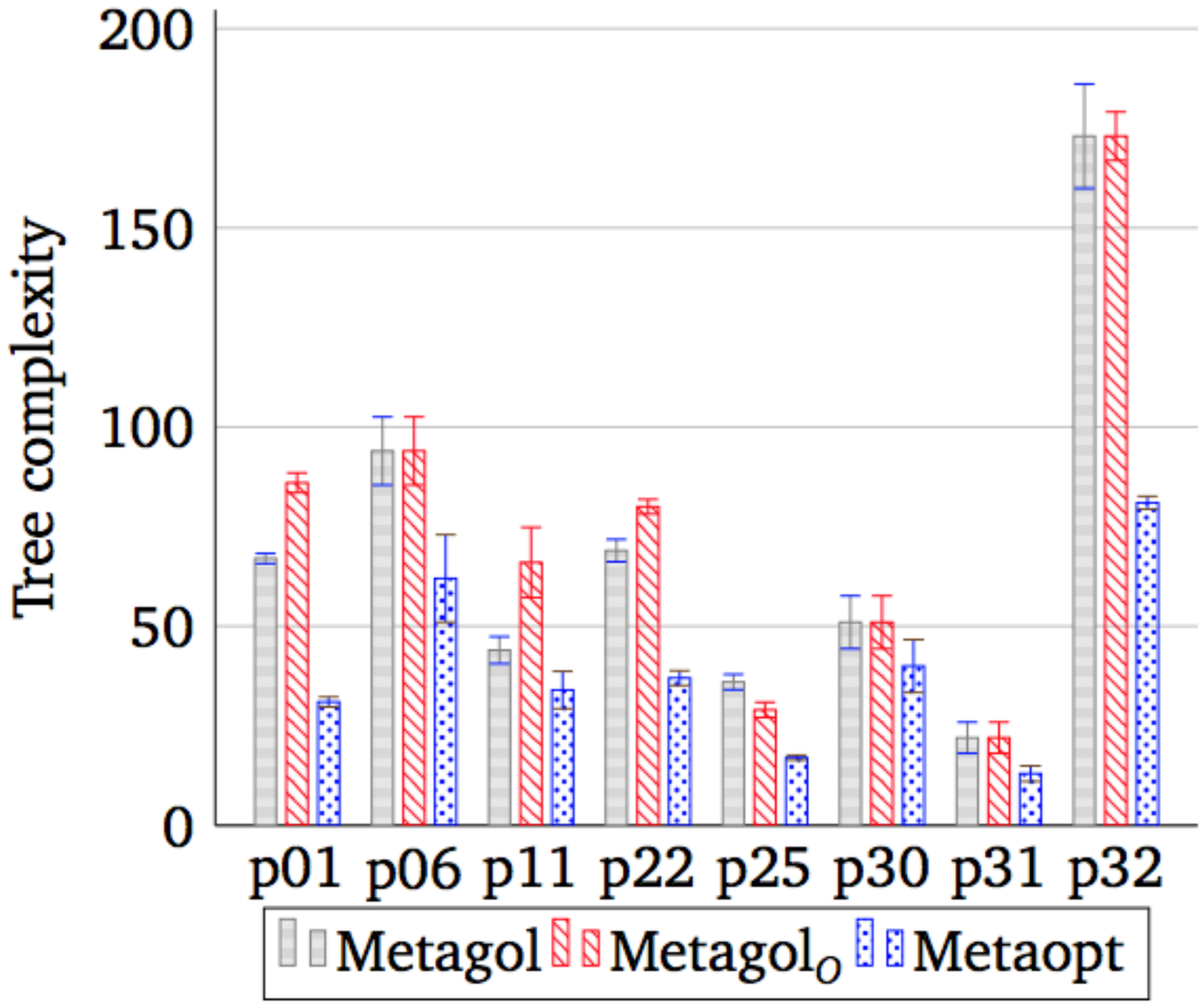
Metagol

```
f(A,B):-tail(A,C),f1(C,B).
f1(A,B):-dropLast(A,C),f2(C,B).
f2(A,B):-dropWhile(A,B,not_uppercase).
```

Metaopt

```
f(A,B):-tail(A,C),f1(C,B).
f1(A,B):-f2(A,C),dropLast(C,B).
f2(A,B):-f3(A,C),f3(C,B).
f3(A,B):-tail(A,C),f4(C,B).
f4(A,B):-f5(A,C),f5(C,B).
```

String transformations



Future work

Learn something novel