

Logic-based inductive synthesis of efficient programs

Andrew Cropper

Imperial College London

String transformations

| Input | Output |
|--|---------------|
| Computer Aided Verification | CAV |
| Principles Of Programming Languages | POPL |
| International Conference on Functional Programming | ??? |
| International Joint Conference Artificial Intelligence | ??? |
| Neural Information Processing Systems | ??? |

```
% python
def f(input):
    output=[]
    for x in string:
        if x.isupper():
            output.append(x)
    return output
```

```
% prolog
f([], []).
f([X|Input], [X|Output]):-
    is_uppercase(X),
    f(Input, Output).
f([X|Input], Output):-
    not_uppercase(X),
    f(Input, Output).
```

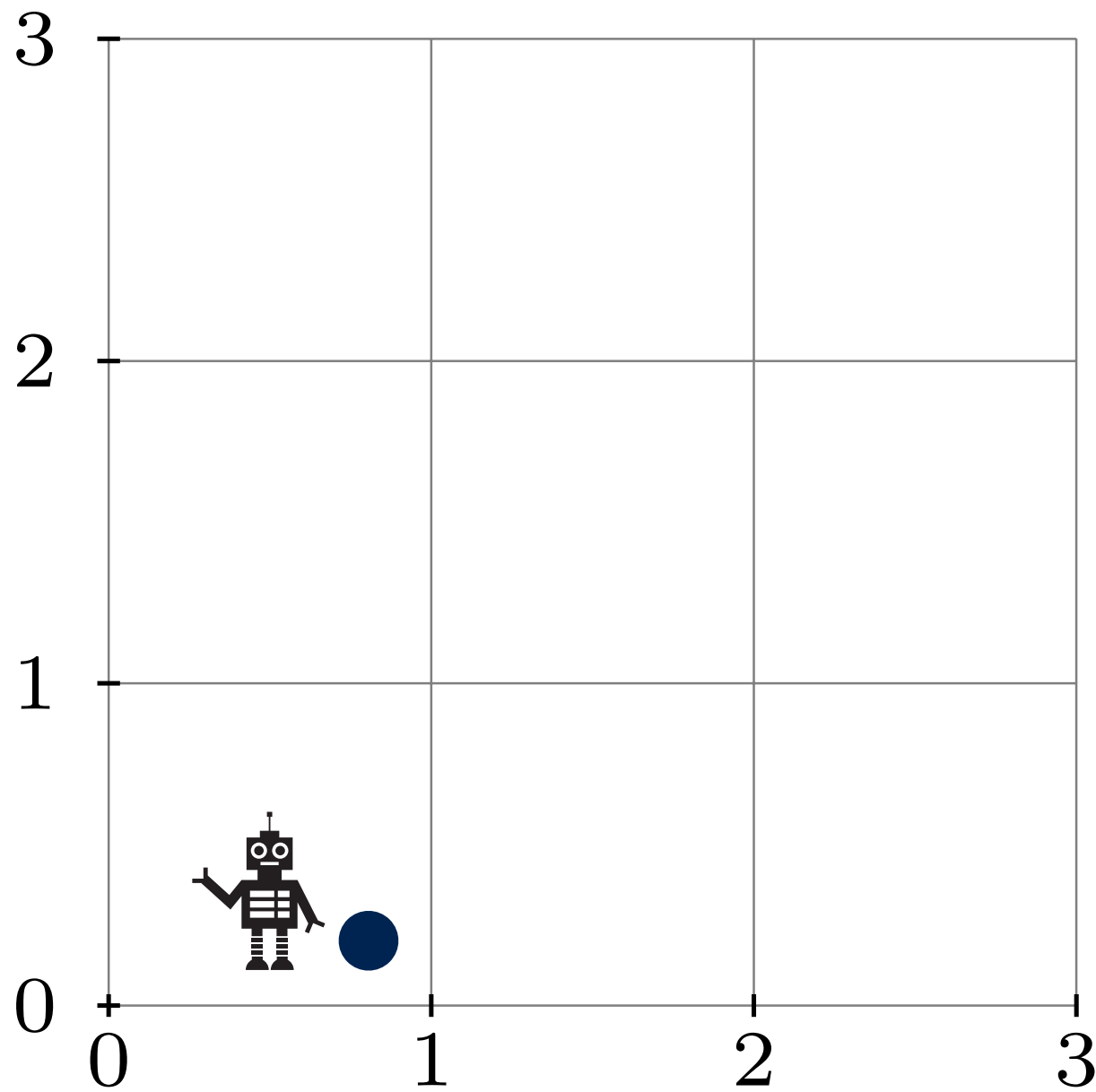
Sorting

| 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] | ??? |

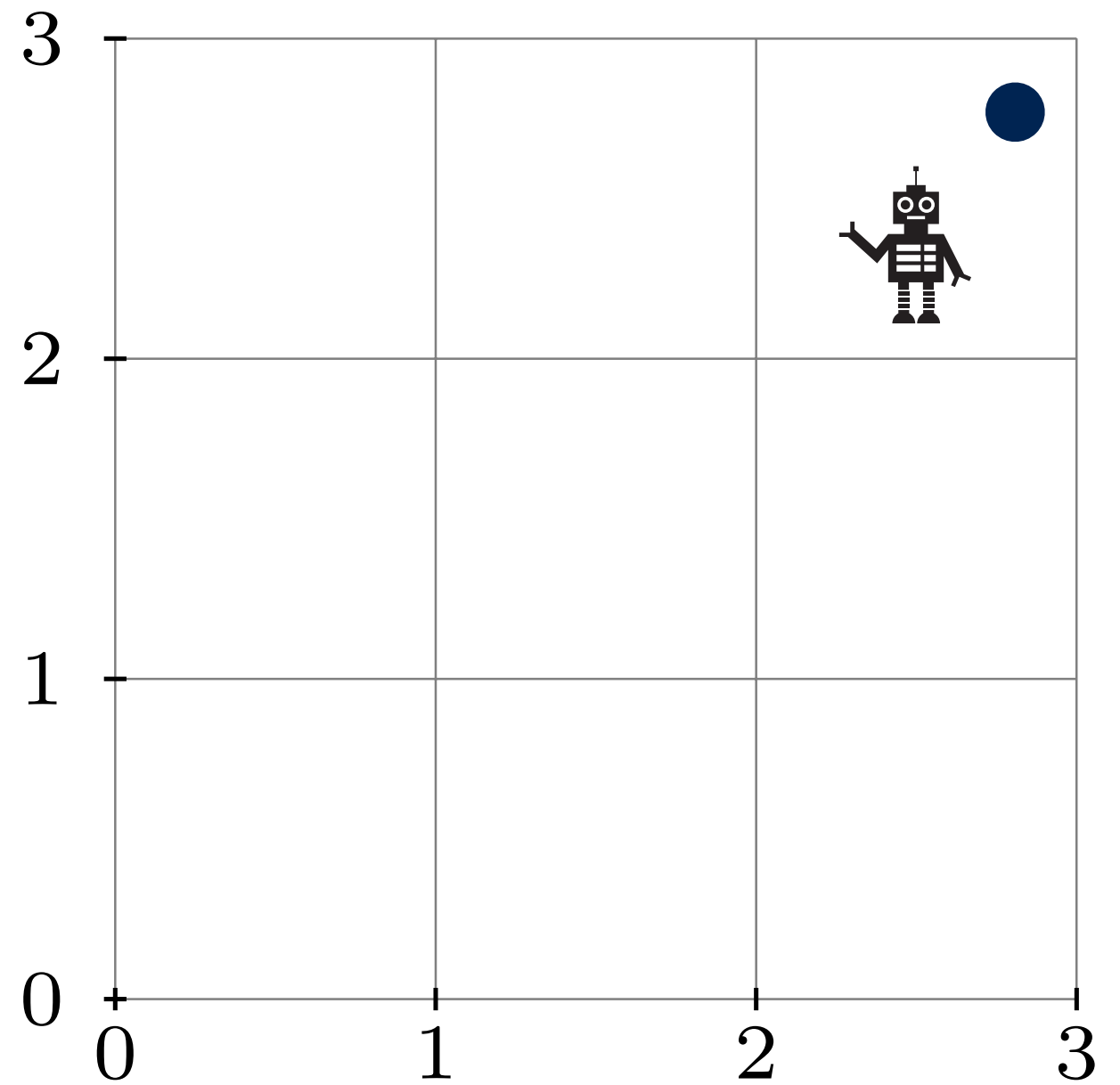
Learning efficient robot strategies

[IJCAI15]

Initial state



Final state

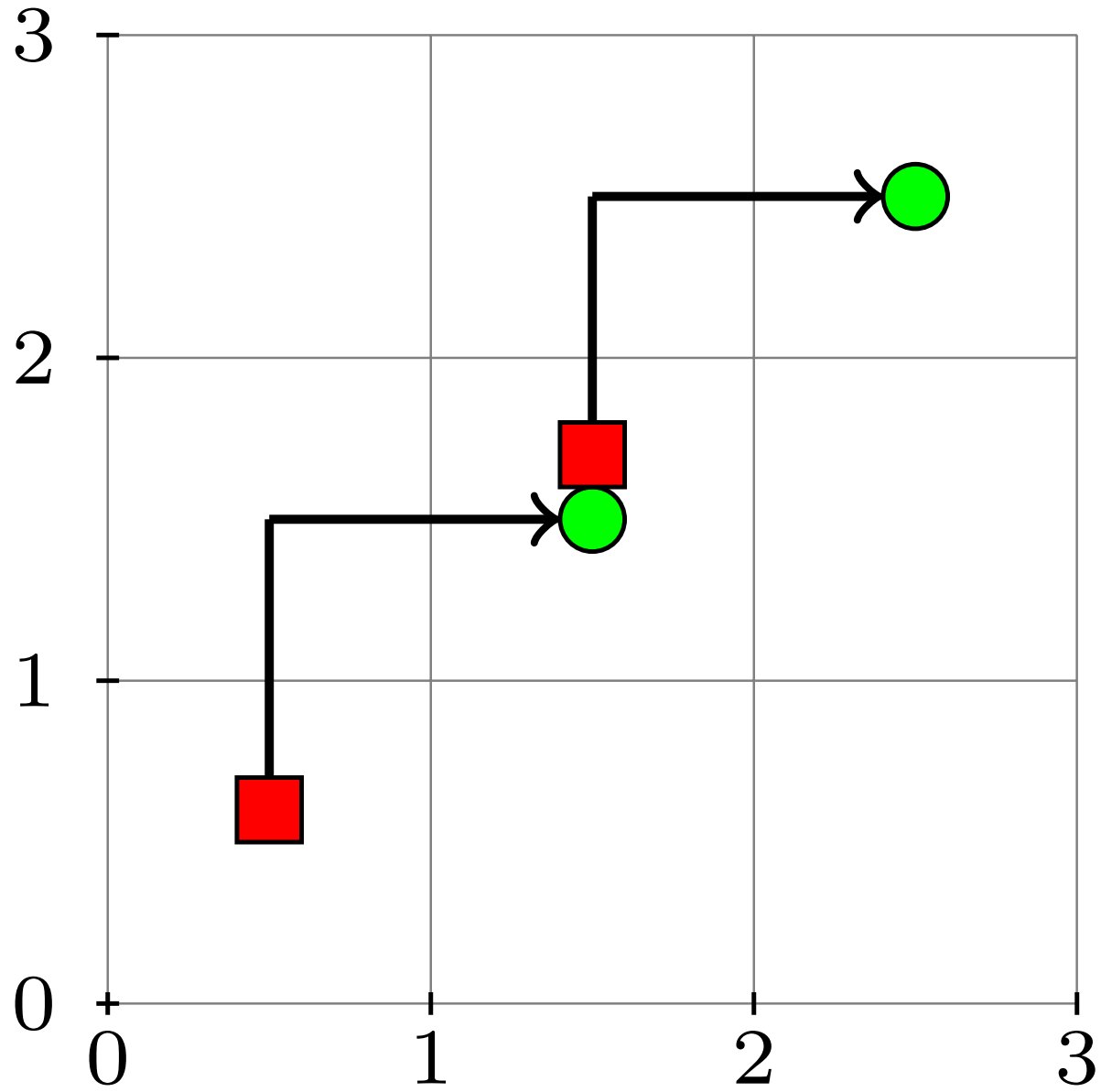


move(X, Y) :- p3(X, Z), p3(Z, Y).
p3(X, Y) :- p2(X, Z), drop(Z, Y).
p2(X, Y) :- grab(X, Z), p1(Z, Y).
p1(X, Y) :- north(X, Z), east(Z, Y).

move(X, Y) :- p3(X, Z), drop(Z, Y).
p3(X, Y) :- grab(X, Z), p2(Z, Y).
p2(X, Y) :- p1(X, Z), p1(Z, Y).
p1(X, Y) :- north(X, Z), east(Z, Y).

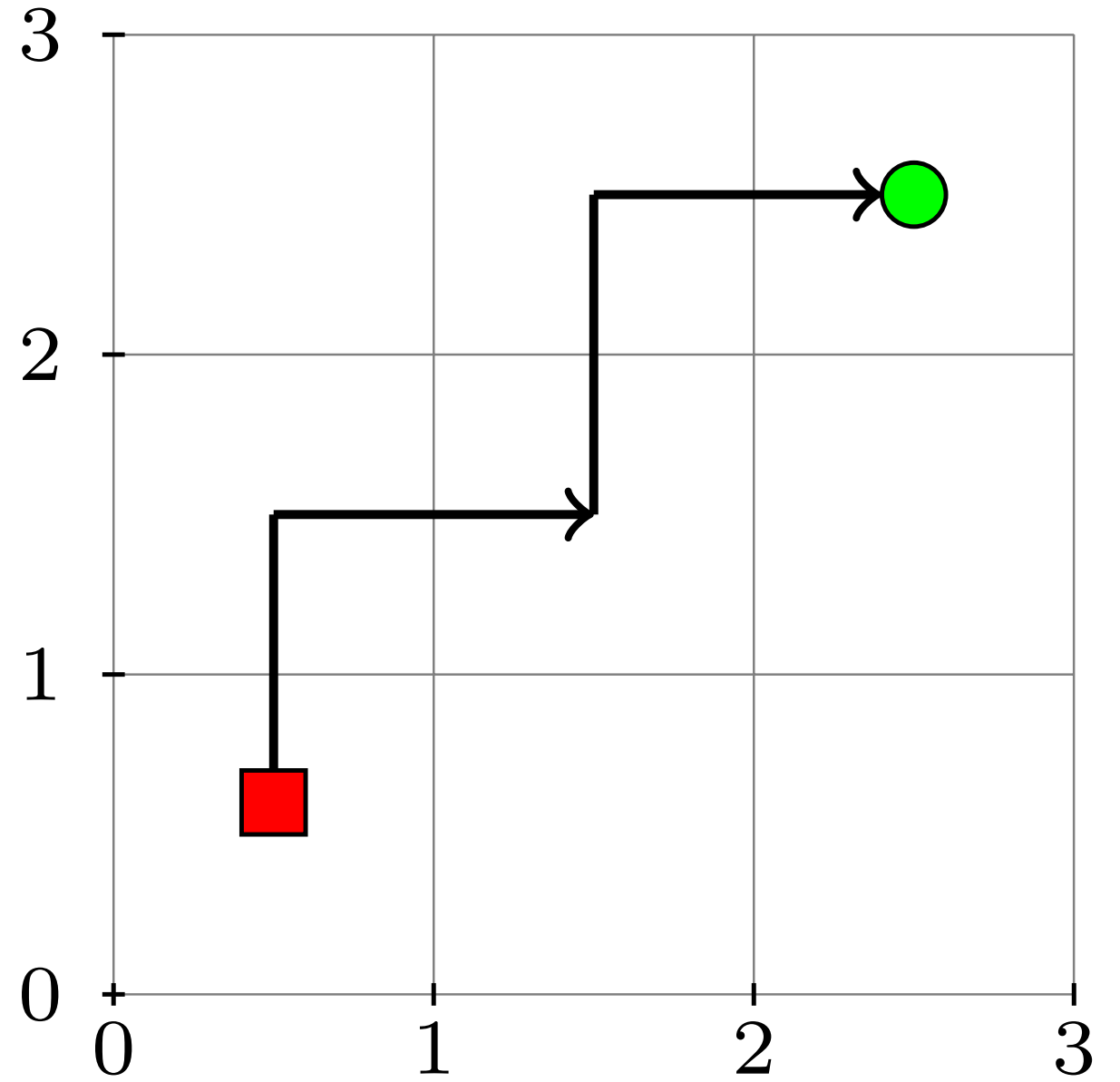
■ grab ● drop

Inefficient solution



resource complexity: 8

Efficient solution



resource complexity: 6

| | | | | |
|---------------|------|------|-------|------|
| Action | drop | grab | north | east |
| Cost | 1 | 1 | 1 | 1 |

Meta-interpretive learning (MIL)

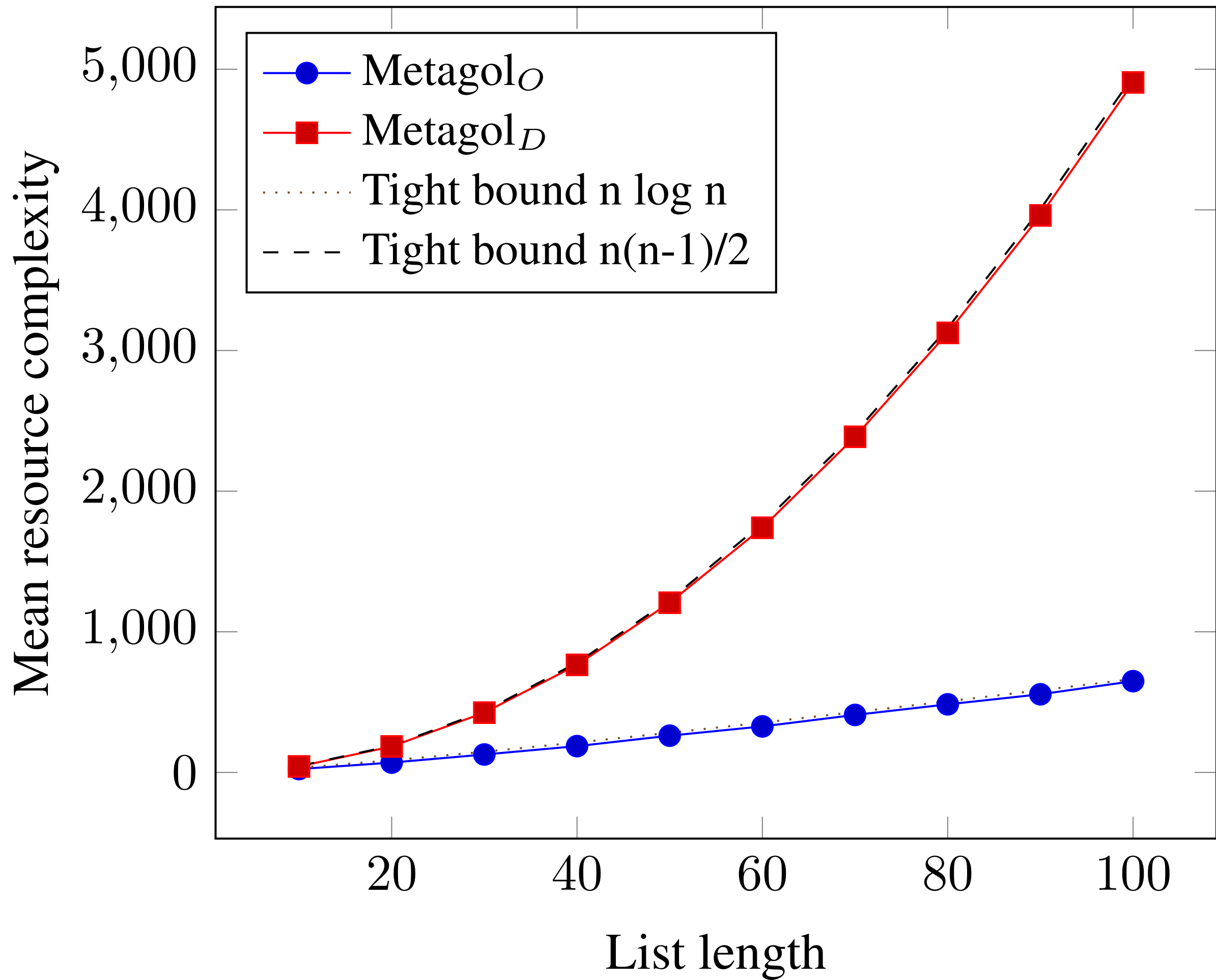
A form of inductive logic programming based on Prolog meta-interpreter which supports:

- predicate invention
- learning recursive programs
- learning efficient programs
- learning higher-order programs

<https://github.com/metagol>

Sorting experiment

| 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] | ??? |



Conclusions

- pruning the search space [IJCAI15]
- higher-order programs [IJCAI16]

Future work

- Demonstrate generality
- Use to discover novel algorithms

Thank you