Learning higher-order logic programs

Andrew Cropper, Rolf Morel, and Stephen Muggleton

input	**output**
ecv	cat
fqi	dog
iqqug	**?**

input	**output**
ecv	cat
fqi	dog
iqqug	**goose**

```
f(A,B):-
    empty(A),
    empty(B).
f(A,B):-
    head(A,C),
    char_to_int(C,D),
    prec(D,E),
    int_to_char(E,F),
    head(B,F),
    tail(A,G),
    tail(B,H),
    f(G,H).
```

```
f(A,B):-
    empty(A),
    empty(B).
f(A,B):-
    head(A,C),
    f1(C,F),
    head(B,F),
    tail(A,G),
    tail(B,H),
    f(G,H).
```

```
f1(A,B):-
    char_to_int(A,C),
    prec(C,D),
    int_to_char(D,B).
```

Idea

Learn higher-order programs

```
f(A,B):-
    map(A,B,f1).
f1(A,B):-
    char_to_int(A,C),
    prec(C,D),
    int_to_char(D,B).
```

```
map([],[],_F).
map([A|As],[B|Bs],F):-
    call(F,A,B),
    map(As,Bs,F).
```

Why?

Increase branching but reduce depth

How?

Extend Metagol

```
learn(Pos,Neg,Prog):-
    prove(Pos,[],Prog),
    \+ prove(Neg,Prog,Prog).
```

```
prove([],Prog,Prog).
prove([Atom|Atoms],Prog1,Prog2):-
    prove_aux(Atom,Prog1,Prog3),
    prove(Atoms,Prog3,Prog2).
```

```
prove_aux(Atom,Prog,Prog):-
   call(Atom).
```

$$\mathbf{P}(A,B) \leftarrow \mathbf{Q}(A,C), \mathbf{R}(C,B)$$

$$\mathbf{P}(A,B) \leftarrow \mathbf{Q}(A,C), \mathbf{R}(C,B)$$

```
metarule(
  chain, % name
  [P,Q,R], % subs
  [P,A,B], % head
  [[Q,A,C],[R,C,B]] % body
).
```

```
prove_aux(Atom,Prog1,Prog2):-
    metarule(Name,Subs,Atom,Body),
    bind(Subs),
    Prog3 = [sub(Name,Subs)|Prog1],
    prove(Body,Prog3,Prog2).
```

% background knowledge
succ/2
int_to_char/2
map/3

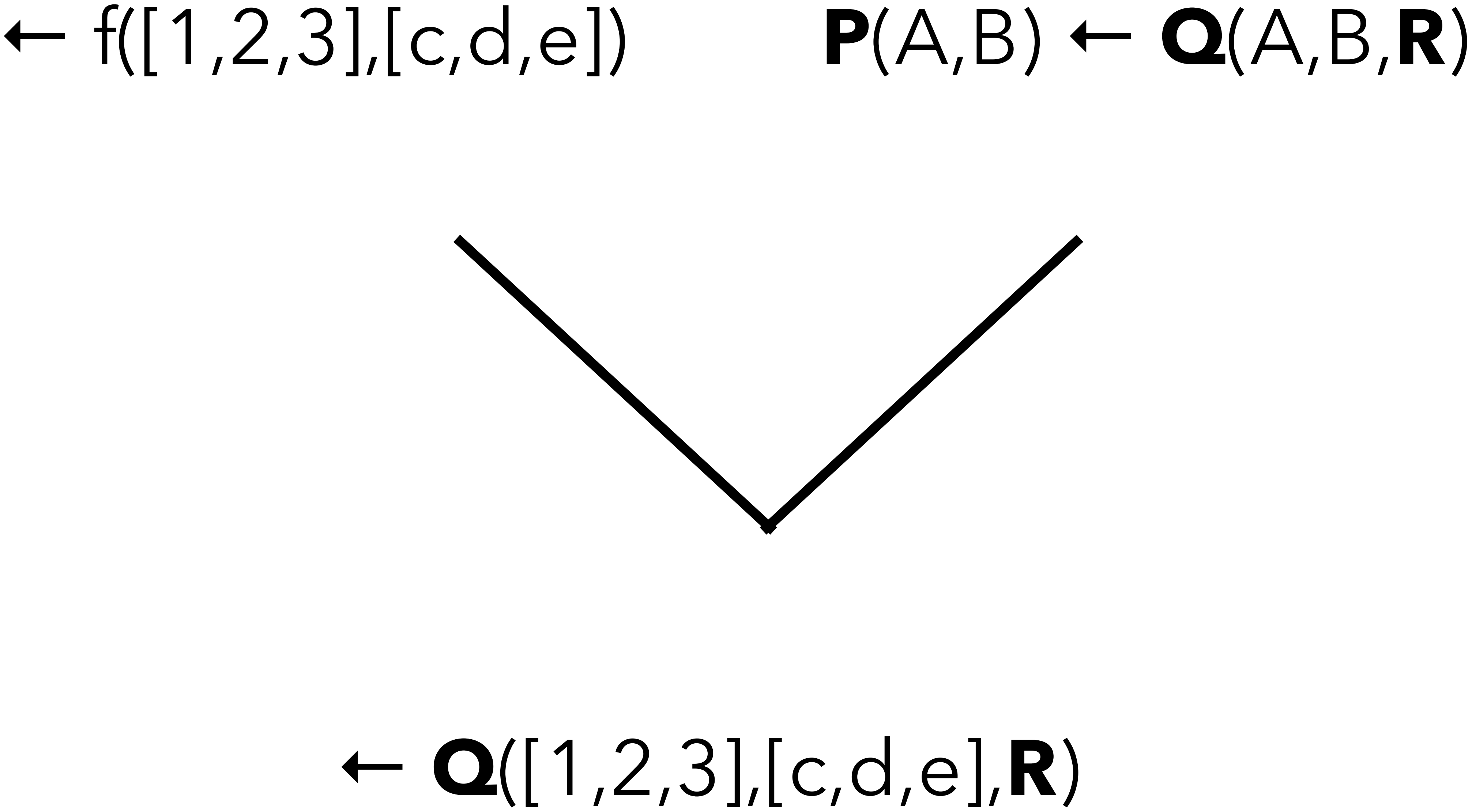
% positive example
f([1,2,3],[c,d,e])

% metarules
P(A,B) ← **Q**(A,C),**R**(C,B)
P(A,B) ← **Q**(A,B,**R**)

← f([1,2,3],[c,d,e])

← f([1,2,3],[c,d,e])

P(A,B) ← **Q**(A,B,**R**)

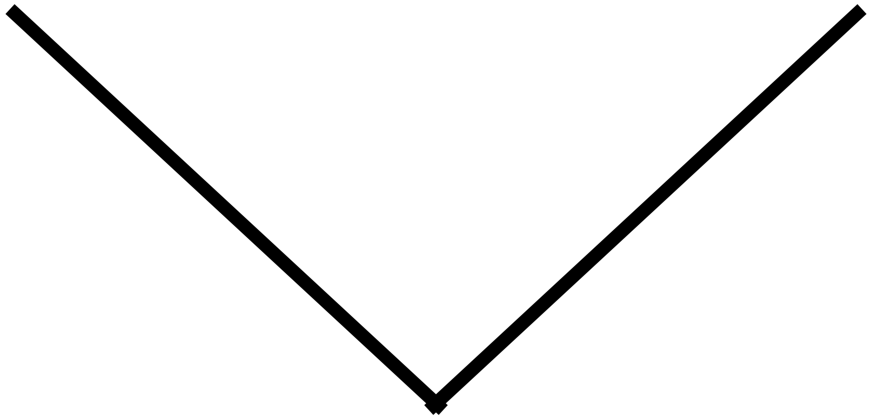


← f([1,2,3],[c,d,e])

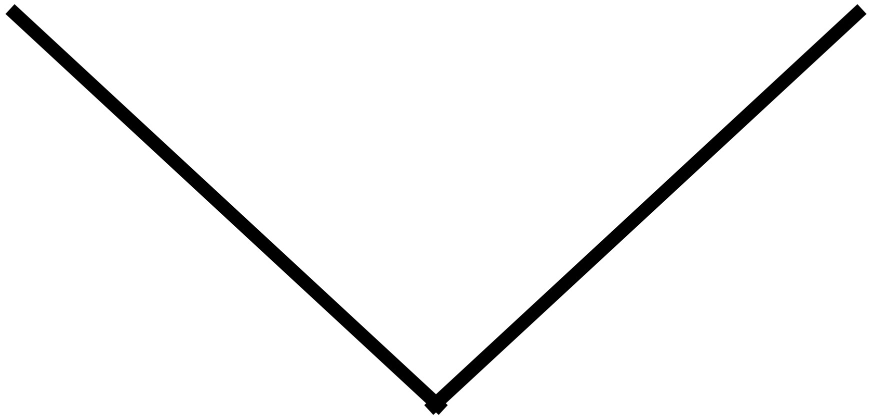
P(A,B) ← **Q**(A,B,**R**)

← **Q**([1,2,3],[c,d,e],**R**)

← **Q**([1,2,3],[c,d,e],**R**)



← **Q**([1,2,3],[c,d,e],**R**)



% proof fails

map([1,2,3],[c,d,e],succ) ✗

map([1,2,3],[c,d,e],int_to_char) ✗

```
f(A,B):-f1(A,C),f3(C,B)
f1(A,B):-f2(A,C),f2(C,B).
f2(A,B):-map(A,B,succ).
f3(A,B):-map(A,B,int_to_char).
```

```
f(A,B):-
    map(A,C,succ).
    map(C,D,succ).
    map(D,B,int_to_char).
```

Higher-order definitions

```
ibk(
    [map,[],[],_F], % head
    [] % body
).
```

Higher-order definitions

```
ibk(
    [map,[A|As],[B|Bs],F], % head
    [[F,A,B],[map,As,Bs,F]] % body
).
```

Metagol$_{HO}$

```
prove_aux(Atom,Prog1,Prog2):-
    ibk(Atom,Body),
    prove(Body,Prog1,Prog2).
```

% background

succ/2, int_to_char/2

% ibk

map/3

% example

f([1,2,3],[c,d,e])

% metarule

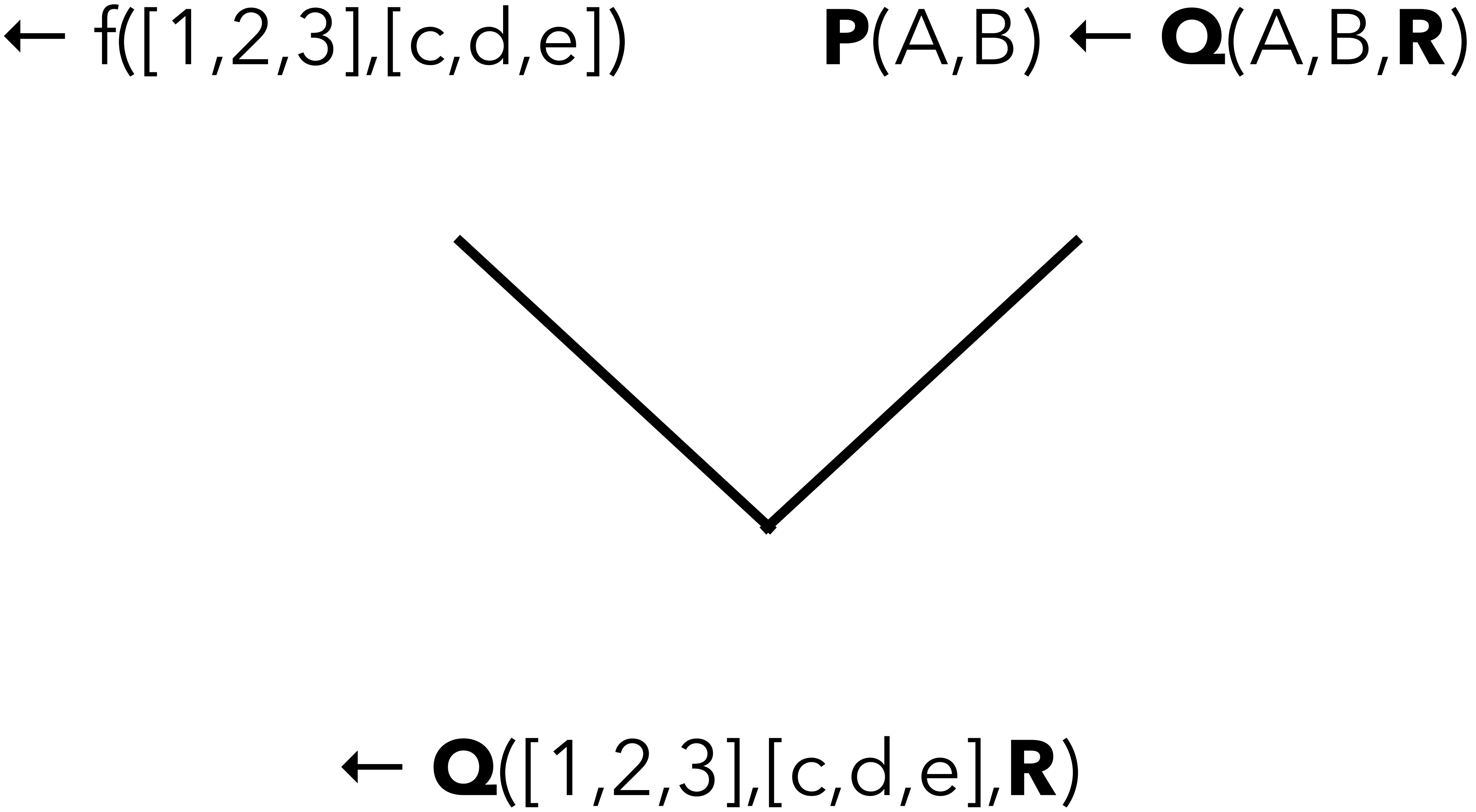
P(A,B) ← **Q**(A,C),**R**(C,B)

P(A,B) ← **Q**(A,B,**R**)

← f([1,2,3],[c,d,e])

$\leftarrow \mathrm{f}([1,2,3],[c,d,e])$

$\mathbf{P}(A,B) \leftarrow \mathbf{Q}(A,B,\mathbf{R})$



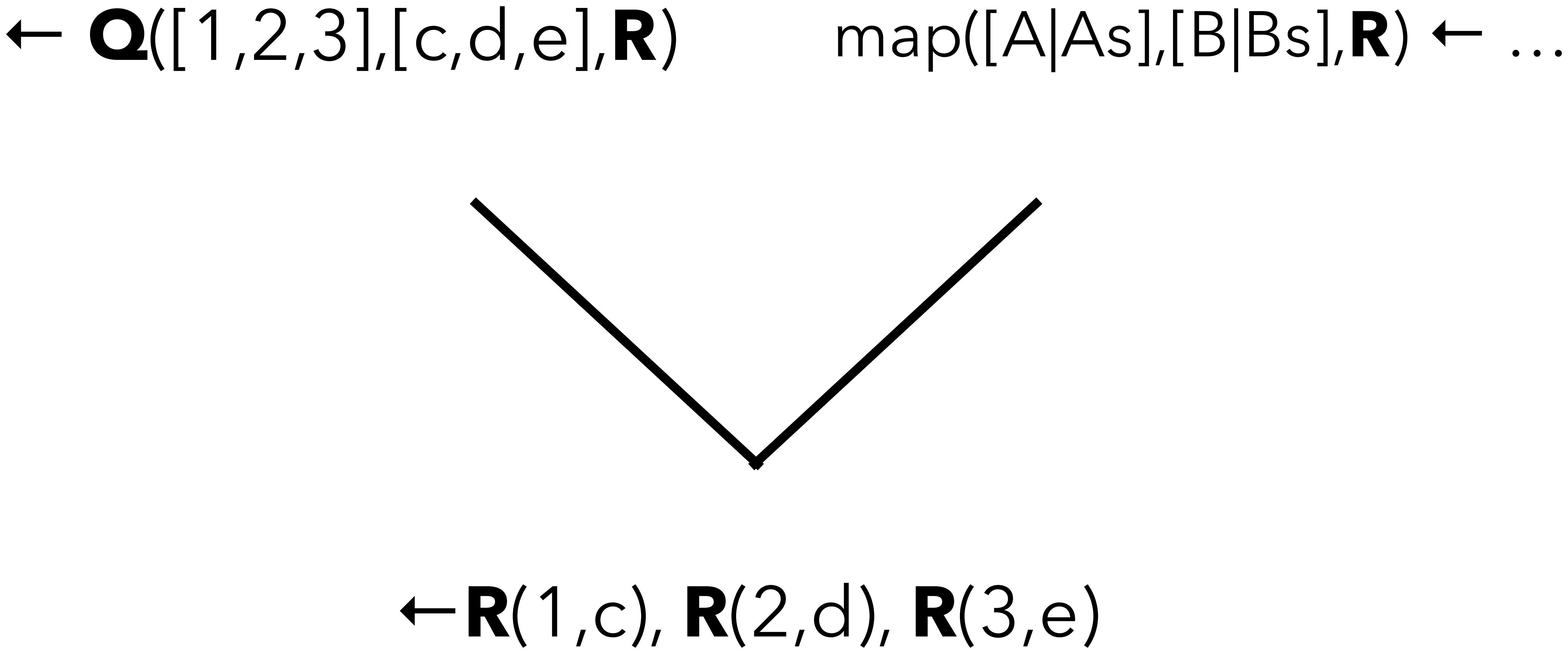
← f([1,2,3],[c,d,e])

P(A,B) ← **Q**(A,B,**R**)

← **Q**([1,2,3],[c,d,e],**R**)

← **Q**([1,2,3],[c,d,e],**R**)

← **Q**([1,2,3],[c,d,e],**R**)   map([A|As],[B|Bs],**R**) ← …

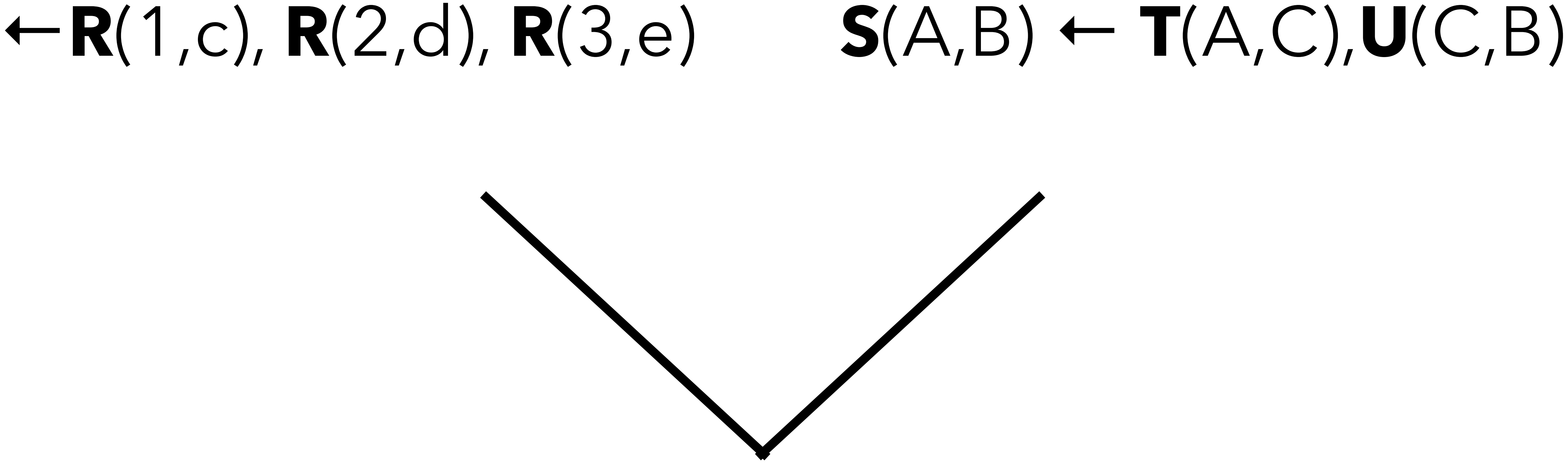


← **Q**([1,2,3],[c,d,e],**R**)

map([A|As],[B|Bs],**R**) ← …

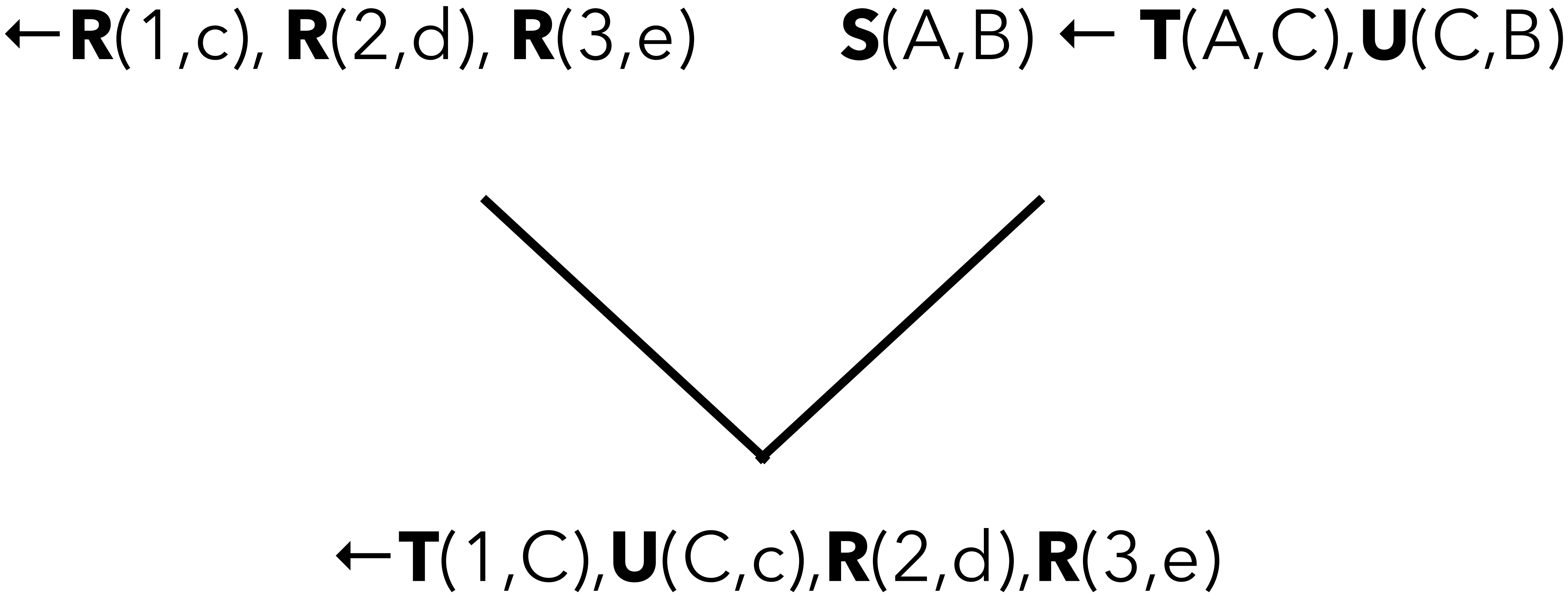
←**R**(1,c), **R**(2,d), **R**(3,e)

$\leftarrow \mathbf{R}(1,c), \mathbf{R}(2,d), \mathbf{R}(3,e)$



←**R**(1,c), **R**(2,d), **R**(3,e)

S(A,B) ← **T**(A,C),**U**(C,B)



←**R**(1,c), **R**(2,d), **R**(3,e)

S(A,B) ← **T**(A,C),**U**(C,B)

←**T**(1,C),**U**(C,c),**R**(2,d),**R**(3,e)

```
f(A,B):-map(A,B,f1).
f1(A,B):-succ(A,C),f2(C,B).
f2(A,B):-succ(A,C),int_to_char(C,B).
```

```
f(A,B):-
    map(A,B,f1).
f1(A,B):-
    succ(A,C),
    succ(A,D),
    int_to_char(D,B).
```

input	**output**
ecv	cat
fqi	dog
iqqug	**?**

Metagol

```
f(A,B):-f1(A,B),f5(A,B).
f1(A,B):-head(A,C),f2(C,B).
f2(A,B):-head(B,C),f3(A,C).
f3(A,B):-char_to_int(A,C),f4(C,B).
f4(A,B):-prec(A,C),int_to_char(C,B),
f5(A,B):-tail(A,C),f6(C,B).
f6(A,B):-tail(B,C),f(A,C).
```

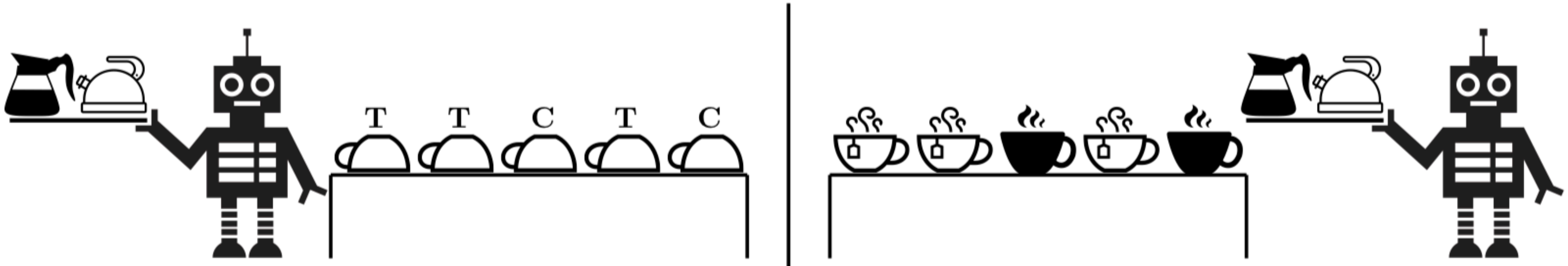
Metagol$_{HO}$

```
f(A,B):-map(A,B,f1).
f1(A,B):-char_to_int(A,C),f2(C,B).
f2(A,B):-prec(A,C),int_to_char(C,B).
```

Does it help in practice?

Q. Can learning higher-order programs improve performance?

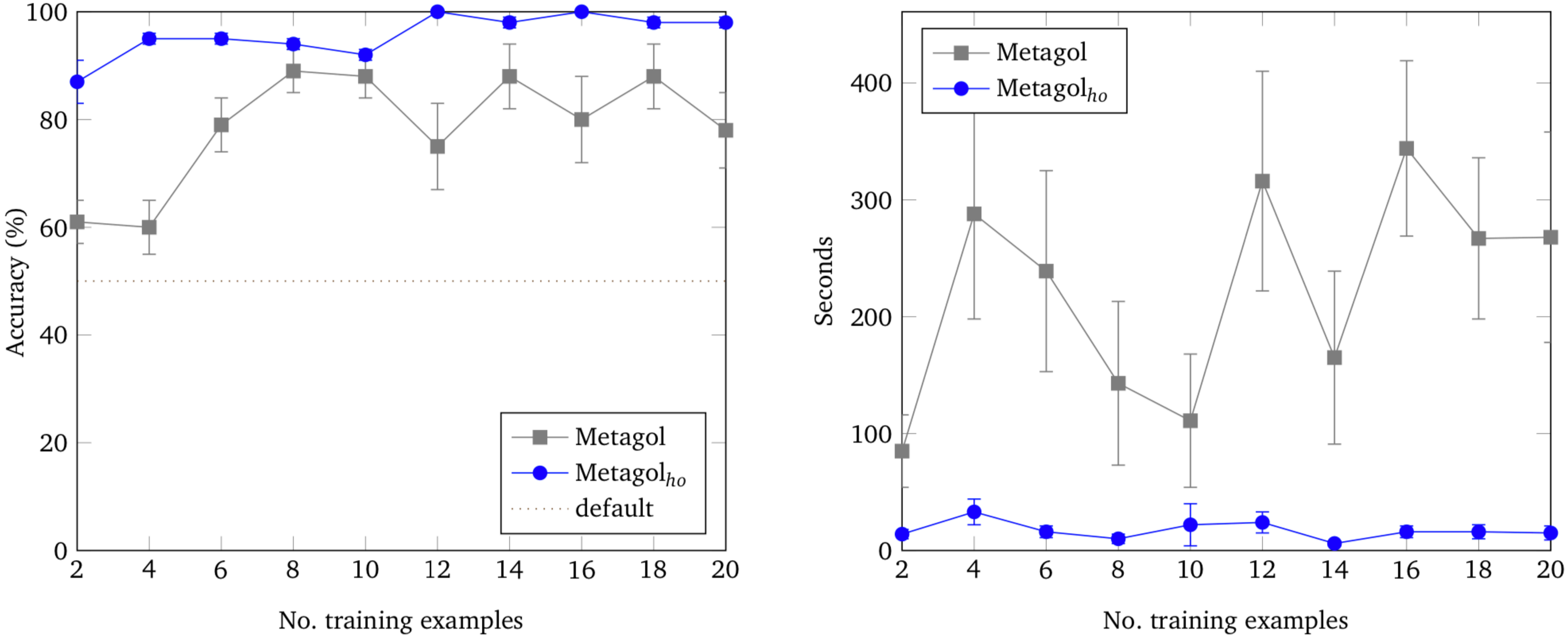
Robot waiter



(a) Initial state

(b) Final state

Robot waiter



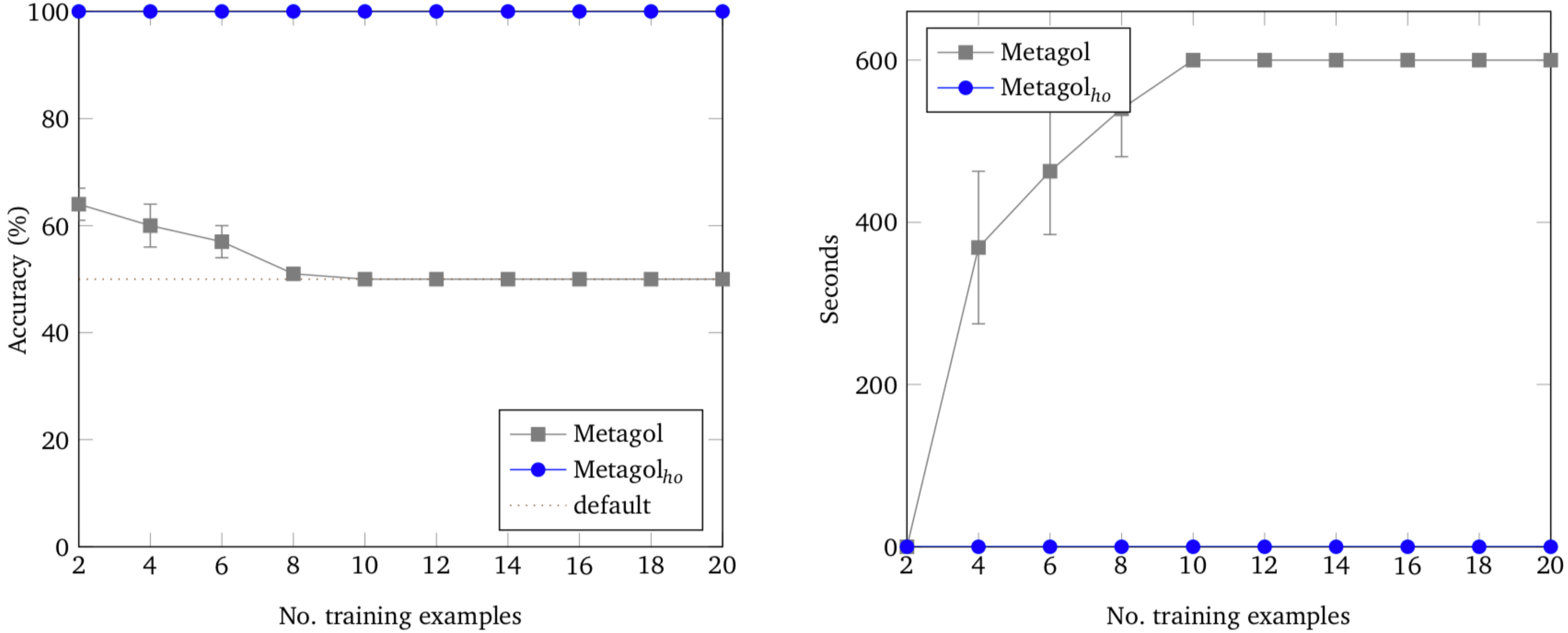
(a) Predictive accuracies

(b) Learning times

Droplasts

Input	Output
[alice,bob,charlie]	[alic,bo,charli]
[inductive,logic,programming]	[inductiv,logi,programmin]
[ferrara,orleans,london,kyoto]	[ferrar,orlean,londo,kyot]

Droplasts



(a) Predictive accuracies

(b) Learning times

```
f(A,B):-map(A,B,f1).
f1(A,B):-f2(A,C),f3(C,B).
f2(A,B):-f3(A,C),tail(C,B).
f3(A,B):-reduceback(A,B,concat).
```

```
f(A,B):-map(A,B,f1).
f1(A,B):-f2(A,C),tail(C,D),f2(D,B).
f2(A,B):-reduceback(A,B,concat).
```

Double droplasts

Input	Output
[alice,bob,charlie]	[alic,bo]
[inductive,logic,programming]	[inductiv,logi]
[ferrara,orleans,london,kyoto]	[ferrar,orlean,londo]

```
f(A,B):-f1(A,C),f2(C,B).
f1(A,B):-map(A,B,f2).
f2(A,B):-f3(A,C),f4(C,B).
f3(A,B):-f4(A,C),tail(C,B).
f4(A,B):-reduceback(A,B,concat).
```

```
f(A,B):-map(A,C,f1),f1(C,B).
f1(A,B):-f2(A,C),tail(C,D),f2(D,B).
f2(A,B):-reduceback(A,B,concat).
```

Limitations

Inefficient search

Which metarules?

Which higher-order definitions?

Conclusion

Learning higher-order programs can help