Learning higher-order logic programs through abstraction and invention

Andrew Cropper and Stephen Muggleton

Input	Output
ebhtuvim	dagstuhl
uvsjoh	turing
qsphsbnnjoh	?

Input	Output
ebhtuvim	dagstuhl
uvsjoh	turing
qsphsbnnjoh	programming

First-order

```
f(A,B):-
  empty(A),
  empty(B).
f(A,B):-
  head(A,Ha),
  head(B,Hb),
  char_code(Ha,Ca),
  succ(Cb,Ca),
  char_code(Hb,Cb),
  tail(A,Ta),
  tail(B,Tb),
  f(Ta,Tb).
```

First-order	**Higher-order**

First-order

```
f(A,B):-
  empty(A),
  empty(B).
f(A,B):-
  head(A,Ha),
  head(B,Hb),
  char_code(Ha,Ca),
  succ(Cb,Ca),
  char_code(Hb,Cb),
  tail(A,Ta),
  tail(B,Tb),
  f(Ta,Tb).
```

Higher-order

```
f(A,B):-
  map(A,B,f1).
f1(A,B):-
  char_code(A,Ca),
  succ(Cb,Ca),
  char_code(B,Cb).
```

Meta-interpretive learning [MLJ15]

Input

E^+ = positive examples
E^- = negative examples
B_c = compiled BK
M = higher-order metarules

Output

Program **H** consistent with **E+** and **E-**

Search space: $\mathbf{O((B_cM)^N)}$

Metarules [ILP14]

Name	**Metarule**
identity	$P(A,B) \leftarrow Q(A,B)$
inverse	$P(A,B) \leftarrow Q(B,A)$
chain	$P(A,B) \leftarrow Q(A,C),R(C,B)$
curry	$P(A,B) \leftarrow Q(A,B,R)$

P,Q,R are **existentially** quantified **higher-order** variables

A,B,*C* are **universally** quantified **first-order** variables

Metagol [MLJ2015]

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

% delegates proof to Prolog
prove([Atom|Atoms],P1,P2):-
 call(Atom),
 prove(Atoms,P1,P2).

% uses metarule to build proof
prove([Atom|Atoms],P1,P2):-
 metarule(Name,Sub,(Atom:-Body)),
 prove(Body,[sub(Name,Subs)|P1],P3)
 prove(Atoms,P3,P2).
```

Metagol [MLJ2015]

```
f(A,B):-empty(A),empty(B).
f(A,B):-f1(A,B),f2(A,B).
f1(A,B):-head(A,Ha),f4(Ha,B).
f2(A,B):-tail(A,Ta),f3(Ta,B).
f3(Ta,B):-tail(B,Tb),f(Ta,Tb).
f4(Ha,B):-f5(Ha,Hb),head(B,Hb).
f5(Ha,Hb):-char_code(Ha,Ca),f6(Ca,Hb).
f6(Ca,Hb):-succ(Hb,Ca),char_code(Hb,Cb).
```

Metagol [MLJ2015]

```
f(A,B):-empty(A),empty(B).
f(A,B):-f1(A,B),f2(A,B).
f1(A,B):-head(A,Ha),f4(Ha,B).
f2(A,B):-tail(A,Ta),f3(Ta,B).
f3(Ta,B):-tail(B,Tb),f(Ta,Tb).
f4(Ha,B):-f5(Ha,Hb),head(B,Hb).
f5(Ha,Hb):-char_code(Ha,Ca),f6(Ca,Hb).
f6(Ca,Hb):-succ(Hb,Ca),char_code(Hb,Cb).
```

Search space: $O((B_c{}^3M)^N)$

Meta-interpretive learning [IJCAI16]

Input

E^+ = positive examples

E^- = negative examples

B_c = compiled BK

B_i = interpreted BK

M = higher-order metarules

Output

Program **H** consistent with **E+** and **E-**

Interpreted BK [IJCAI16]

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

([fold,[],Acc,Acc,F]:-[]).
([fold,[A|As],B,Acc1,F]:-
  [[F,Acc1,A,Acc2],[fold,As,B,Acc2,F]]).
```

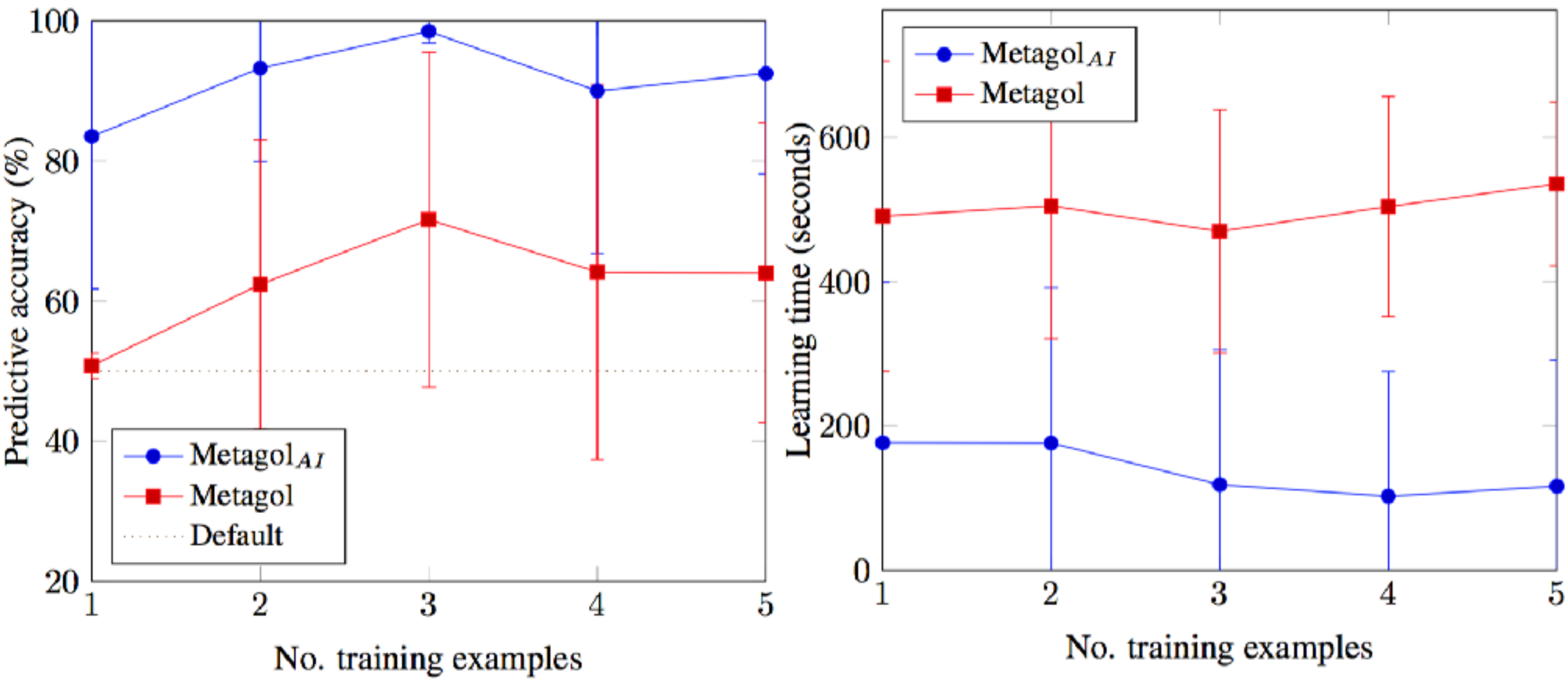
Metagol [IJCAI16]

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

prove([Atom|Atoms],P1,P2):-
  call(Atom),
  prove(Atoms,P1,P2).

prove_aux(Atom,P1,P2):-
   interpreted((Atom:-Body)),
   prove(Body,P1,P3),
  prove(Atoms,P3,P2).

prove([Atom|Atoms],P1,P2):-
  metarule(Name,Sub,(Atom:-Body)),
  prove(Body,[sub(Name,Subs)|P1],P3)
  prove(Atoms,P3,P2).
```

Metagol [IJCAI16]

```
f(A,B):-map(A,B,f1).
f1(A,B):-char_code(A,Ca),f2(Ca,B).
f2(Ca,B):-succ(Cb,Ca),char_code(B,Cb).
```

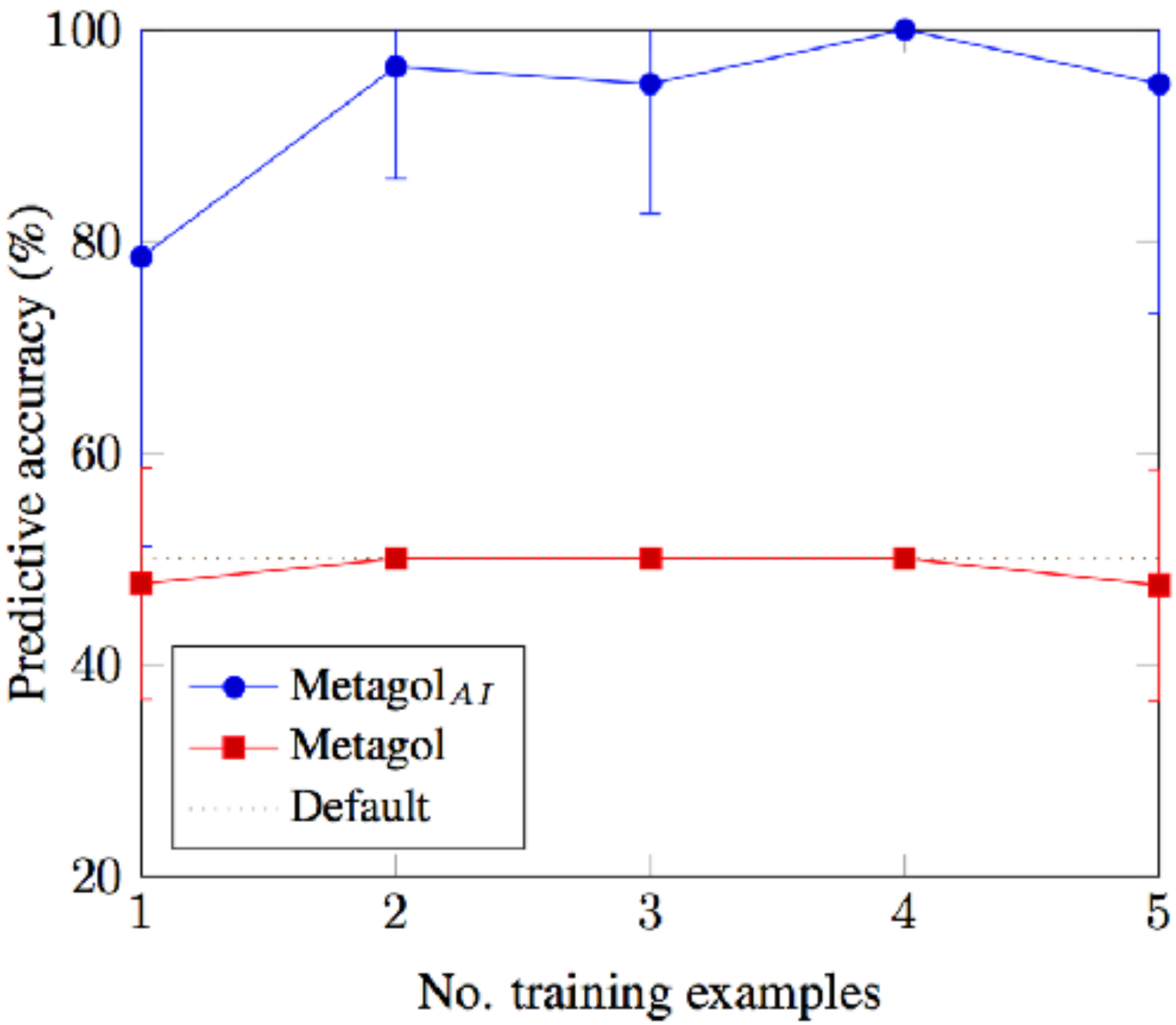
Robot waiter results



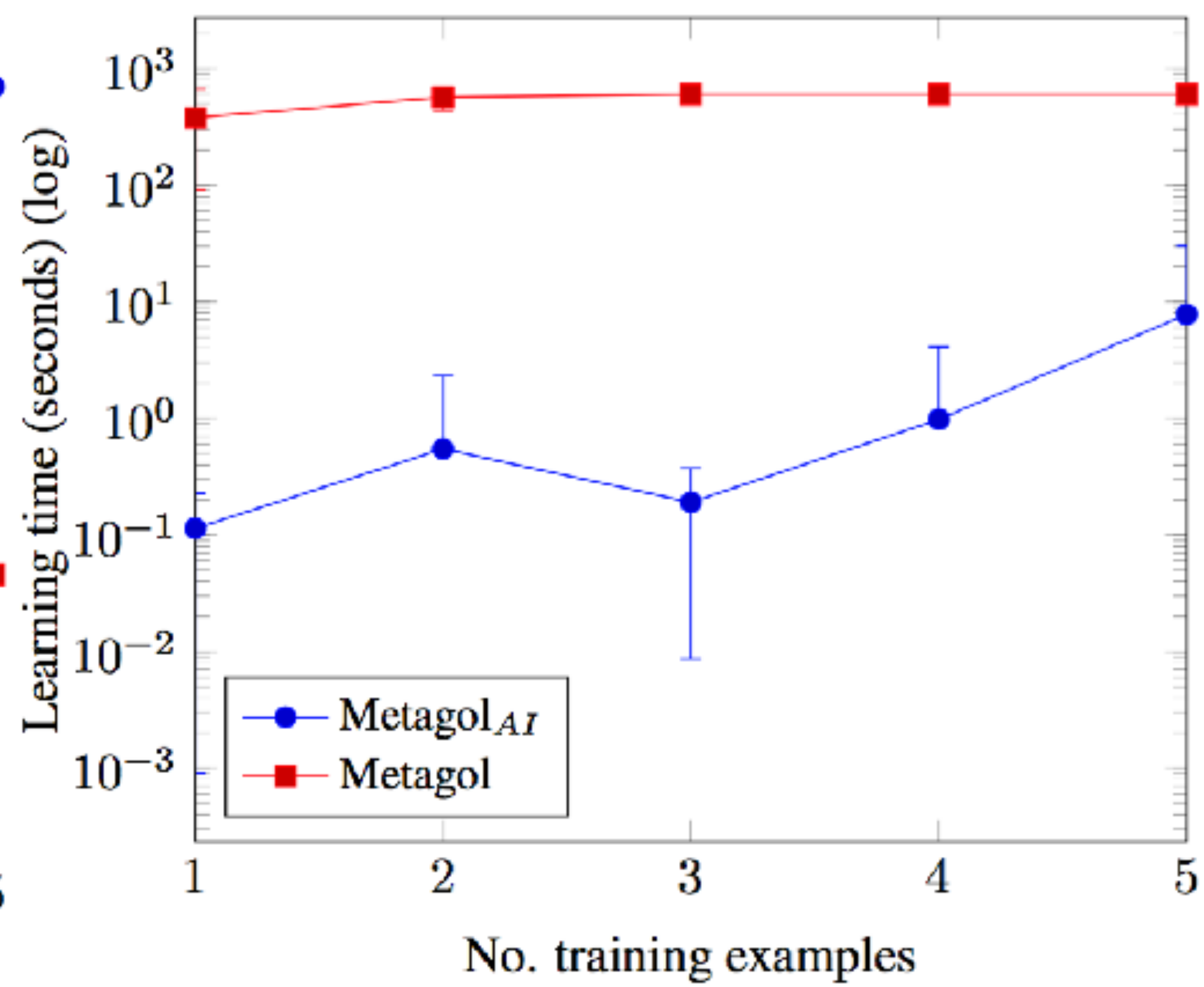
(a) Predictive accuracies

(b) Learning times

Chess results



(a) Predictive accuracies



(b) Learning times

Input	Output
[dagstuhl,2017]	[dagstuh,201]
[alice,bob,charlie]	[alic,bo,charli]
[1234,12,564]	[123,1,56]
[ab,abc,abcd,abcde]	[a,ab,abc,abcd]

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).

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

Input	Output
[dagstuhl,2017]	[dagstuh]
[alice,bob,charlie]	[alic,bo]
[1234,12,564]	[123,1]
[ab,abc,abcd,abcde]	[a,ab,abc]

```
f(A,B):-f4(A,C),f3(C,B).
f4(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).
```

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

Problems

- What metarules do we need?
- What higher-order abstractions do we need?
- Can we invent higher-order abstractions, such as map, fold, etc?

- S.H. Muggleton et al. Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. Machine Learning 100(1): 49-73 (2015).
- A. Cropper and S.H. Muggleton. Logical minimisation of meta-rules within meta-interpretive learning. ILP 2014.
- A. Cropper and S.H. Muggleton. Learning efficient logical robot strategies involving composable objects. IJCAI 2015.
- A. Cropper and S.H. Muggleton. Learning higher-order logic programs through abstraction and invention. IJCAI 2016.