Can predicate invention compensate for incomplete background knowledge?

Andrew Cropper and Stephen H. Muggleton

Imperial College London

Incomplete background knowledge

- missing values
- **missing predicates**

Necessary predicate invention

```
% background knowledge
parent(amy,amelia) ←
parent(gavin,amelia) ←

% examples
father(gavin,amelia) ←
← father(amy,amelia)

% hypothesis
```

Necessary predicate invention

```
% background knowledge
parent(amy,amelia) ←
parent(gavin,amelia) ←

% examples
father(gavin,amelia) ←
← father(amy,amelia)

% hypothesis
father(X,Y) ← parent(X,Y), p1(X).
p1(gavin) ←
```

Necessary predicate invention

```
% background knowledge
parent(ann,amy) ←
parent(john,amy) ←
parent(amy,amelia) ←
parent(amy,bob) ←

mother(ann, amy) ←
father(john, amy) ←
mother(amy, amelia) ←
mother(amy, bob) ←

% examples
grandparent(ann, amelia) ←
grandparent(ann, bob) ←
grandparent(john, amelia) ←
grandparent(john, bob) ←

% hypothesis
```

Useful predicate invention

```
% background knowledge
parent(ann,amy) ←
parent(john,amy) ←
parent(amy,amelia) ←
parent(amy,bob) ←

mother(ann, amy) ←
father(john, amy) ←
mother(amy, amelia) ←
mother(amy, bob) ←

% examples
grandparent(ann, amelia) ←
grandparent(ann, bob) ←
grandparent(john, amelia) ←
grandparent(john, bob) ←

% hypothesis
grandparent(X,Y) ← parent(X,Z), parent(Z,Y)
```

Useful predicate invention

```
% background knowledge
mother(ann, amy) ←
father(john, amy) ←
mother(amy, amelia) ←
mother(amy, bob) ←

% examples
grandparent(ann, amelia) ←
grandparent(ann, bob) ←
grandparent(john, amelia) ←
grandparent(john, bob) ←

% hypothesis
```

Useful predicate invention

```
% background knowledge
mother(ann, amy) ←
father(john, amy) ←
mother(amy, amelia) ←
mother(amy, bob) ←
```

```
% examples
grandparent(ann, amelia) ←
grandparent(ann, bob) ←
grandparent(john, amelia) ←
grandparent(john, bob) ←
```

```
% hypothesis
grandparent(X,Y) ← mother(X,Z), mother(Z,Y)
grandparent(X,Y) ← mother(X,Z), father(Z,Y)
grandparent(X,Y) ← father(X,Z), father(Z,Y)
grandparent(X,Y) ← father(X,Z), mother(Z,Y)
```

Useful predicate invention

```
% background knowledge
mother(ann, amy) ←
father(john, amy) ←
mother(amy, amelia) ←
mother(amy, bob) ←
```

```
% examples
grandparent(ann, amelia) ←
grandparent(ann, bob) ←
grandparent(john, amelia) ←
grandparent(john, bob) ←
```

```
% hypothesis
grandparent(X,Y) ← p1(X,Z), p1(Z,Y)
p1(X,Y) ← mother(X,Y)
p1(X,Y) ← father(X,Y)
```

Meta-interpretive learning

Prolog meta-interpreter

```
prove(true).

prove((Atom,Atoms)):-
 prove(Atom),
 prove(Atoms).

prove(Atom):-
 clause(Atom,Body),
 prove(Body).
```

MIL meta-interpreter

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

prove([Atom|Atoms],G1,G2):-
 call(Atom),
 prove(Atoms,G1,G2).

prove([Atom|Atoms],G1,G2):-
 metarule(Name,MetaSub,(Atom:-Body)),
 abduce(Name,MetaSub,G1,G3),
 prove(Body,G3,G4).
 prove(Atoms,G4,G2).
```

* S.H. Muggleton, D. Lin, and A. Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. Machine Learning, 100(1):49-73, 2015.

Learning great-great-grandparent (gggparent) relation in MIL ***without*** predicate invention

```
gggparent(A,B):- father(A,C), father(C,D), father(D,B).
gggparent(A,B):- father(A,C), father(C,D), mother(D,B).
gggparent(A,B):- father(A,C), mother(C,D), father(D,B).
gggparent(A,B):- father(A,C), mother(C,D), mother(D,B).
gggparent(A,B):- mother(A,C), mother(C,D), mother(D,B).
gggparent(A,B):- mother(A,C), mother(C,D), father(D,B).
gggparent(A,B):- mother(A,C), father(C,D), mother(D,B).
gggparent(A,B):- mother(A,C), father(C,D), father(D,B).
```

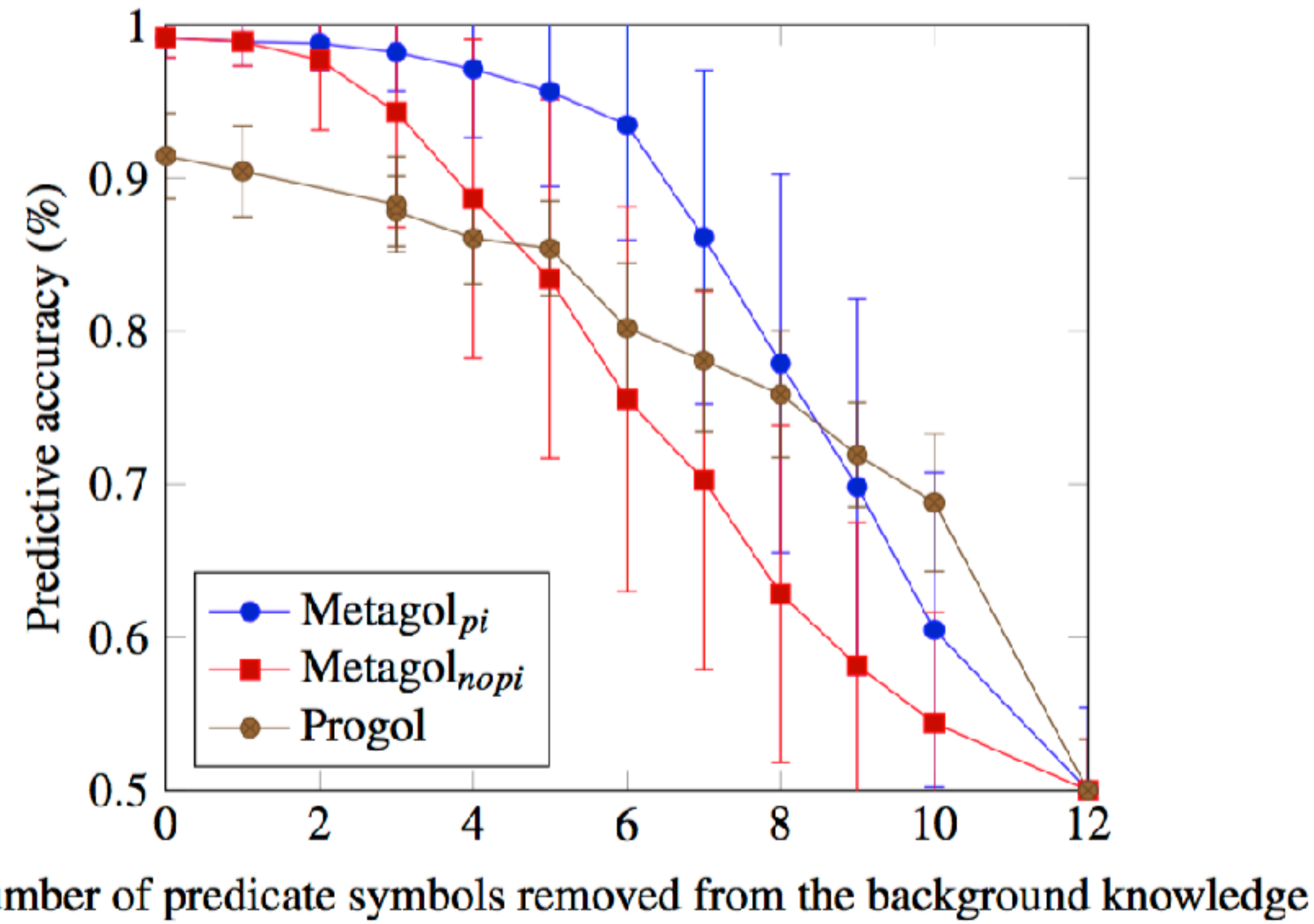
Learning great-great-grandparent (gggparent) relation in MIL ***with*** predicate invention

p2 is invented **grandparent** relation

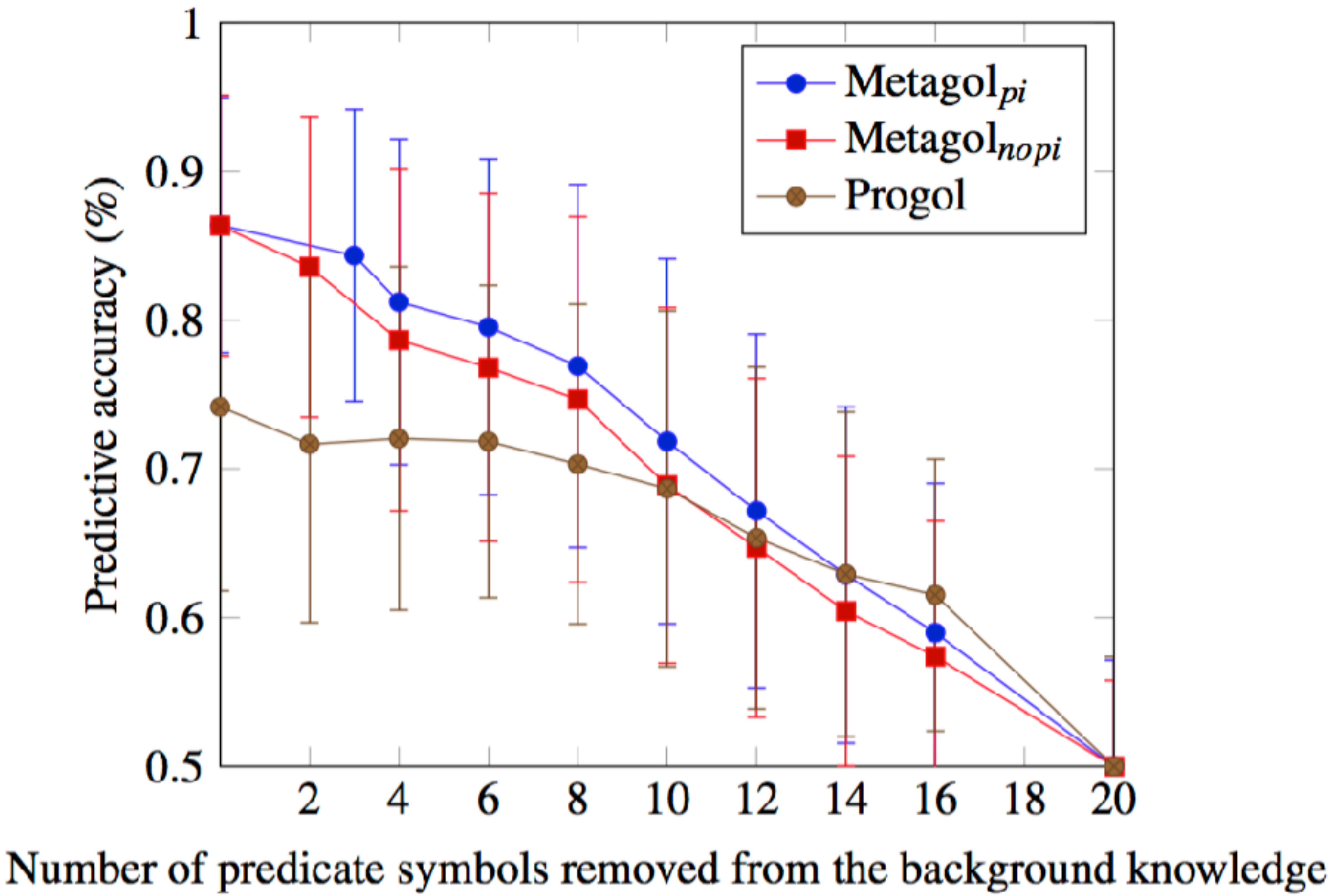
```
gggparent(A,B):- p2(A,C), p2(C,B).
p2(A,B):- p1(A,C), p1(C,B).
p1(A,B):- father(A,B).
p1(A,B):- mother(A,B).
```

p1 is invented **parent** relation

Experiments - Hinton's kinship

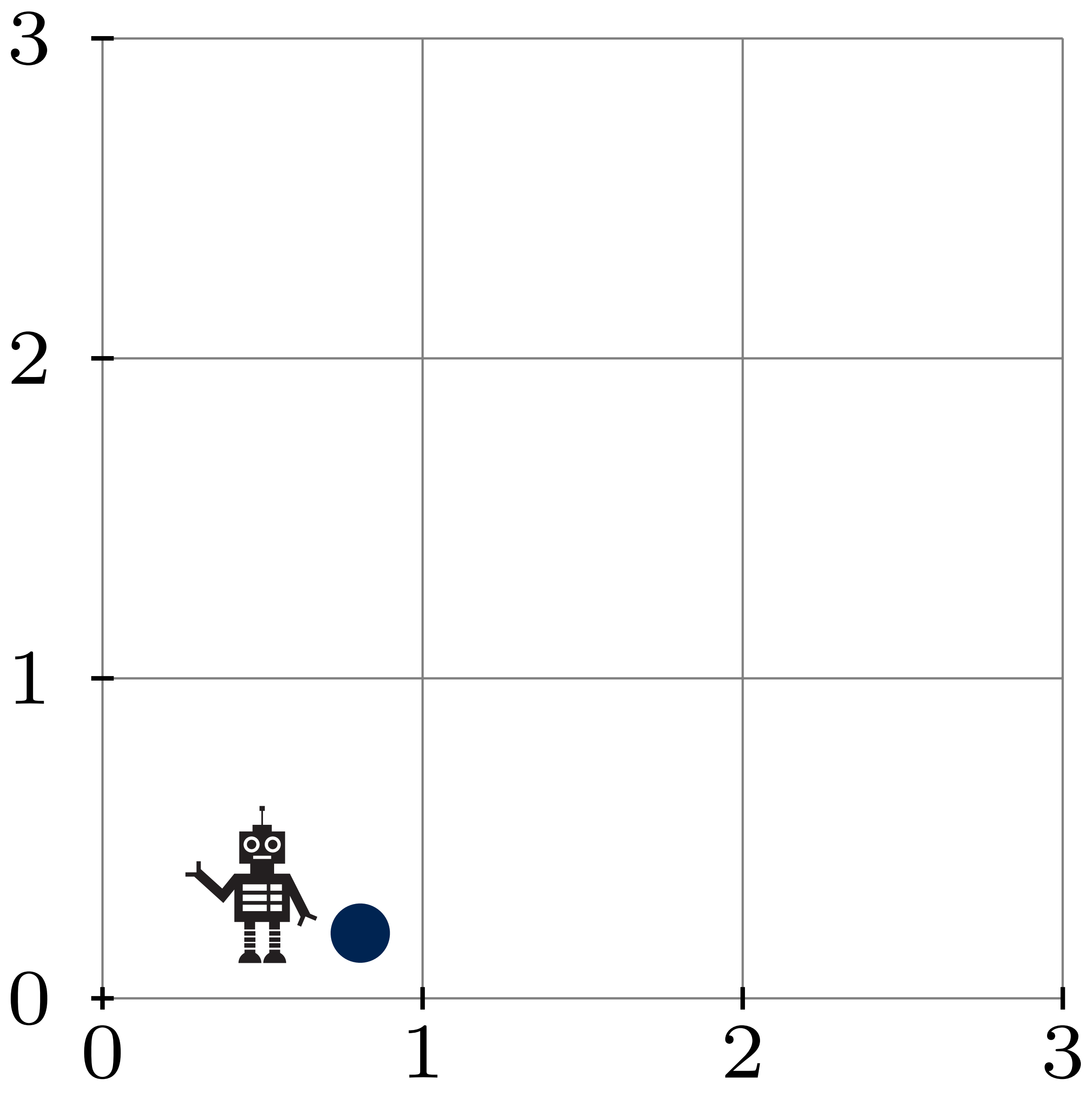


Experiments - custom kinship dataset

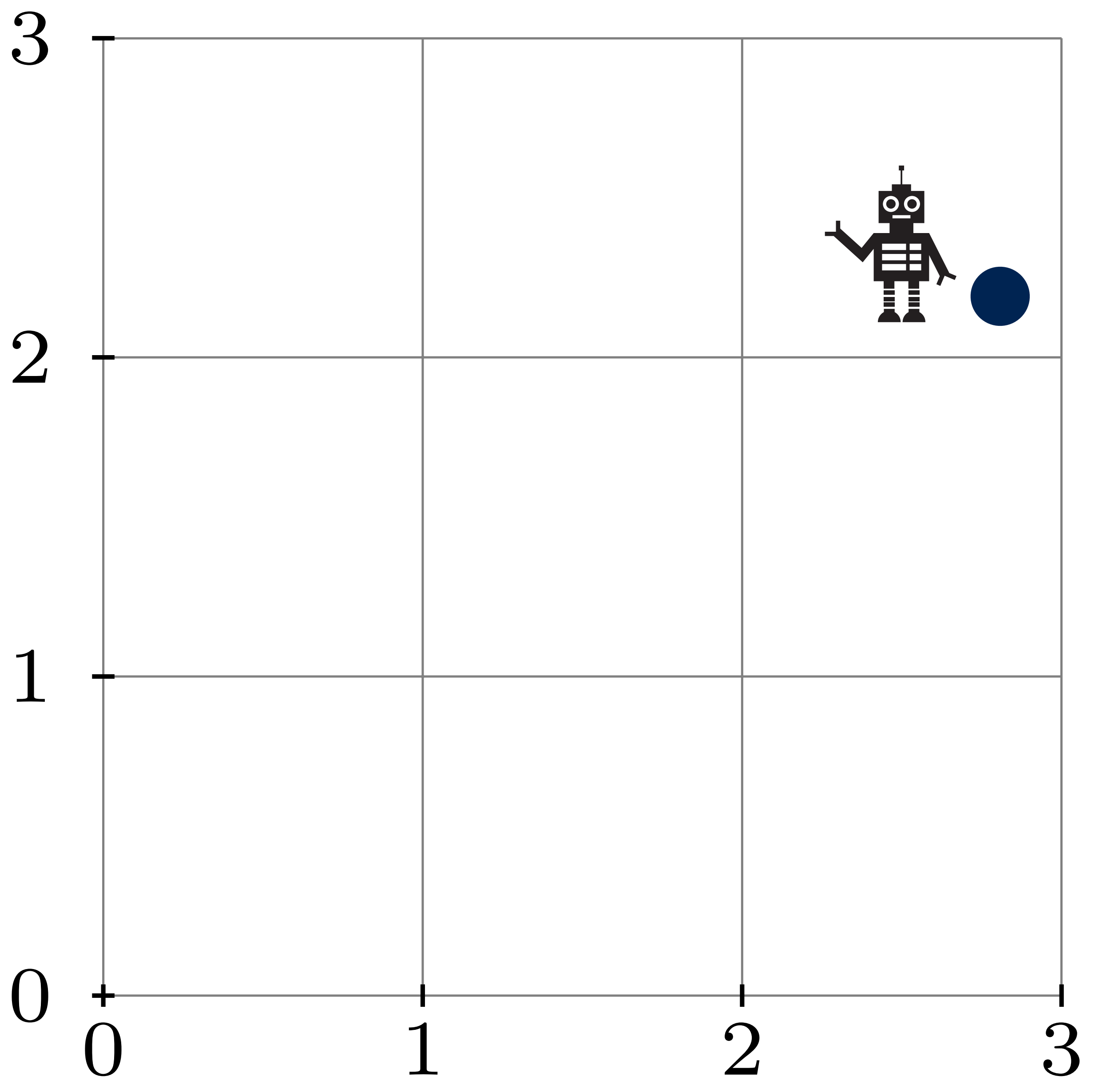


Learning robot plans

Initial state



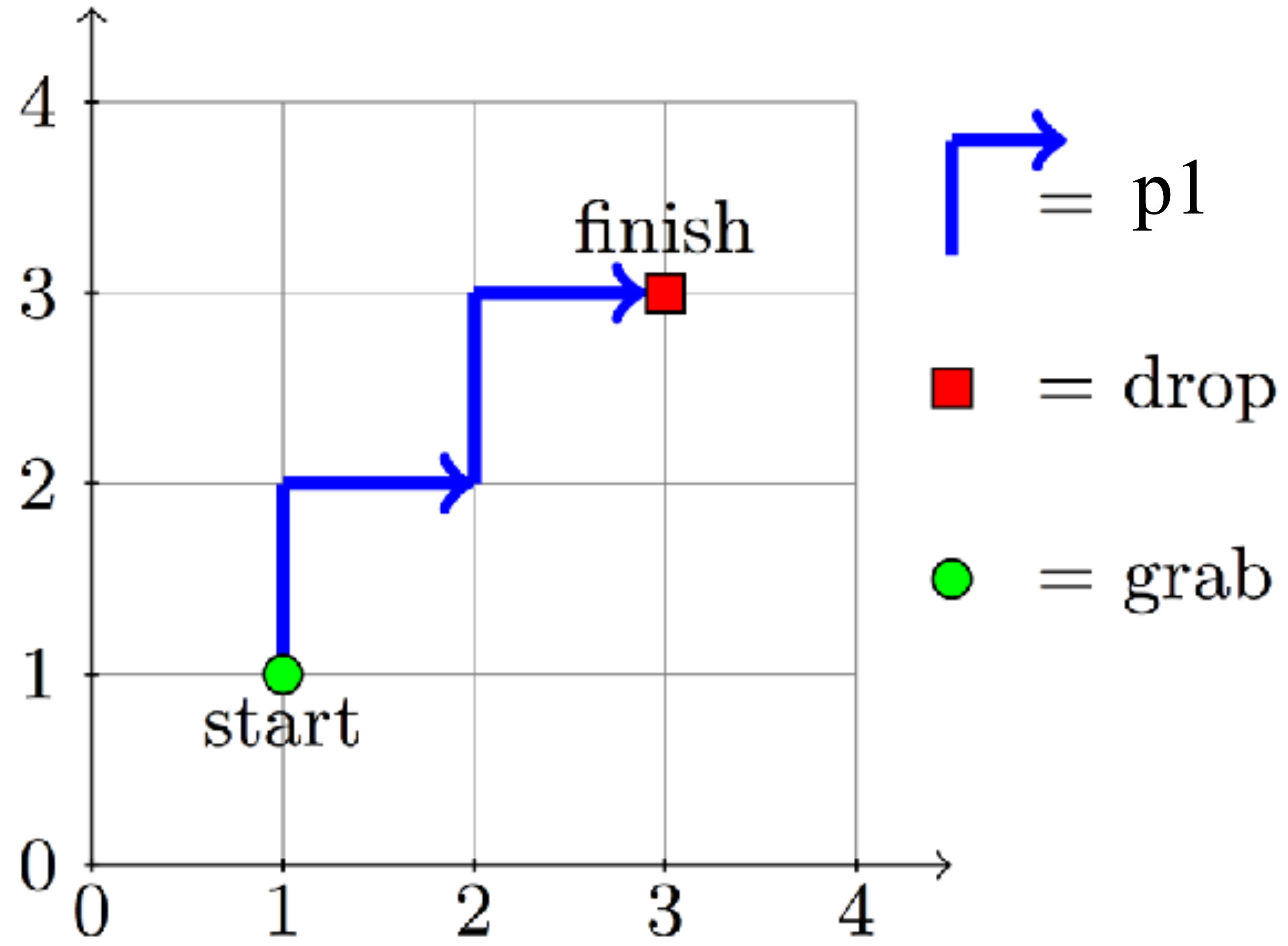
Final state



[pos(robot,1/1),pos(ball,1/1)]

[pos(robot,3/3),pos(ball,3/3)]

Plan learned with MIL

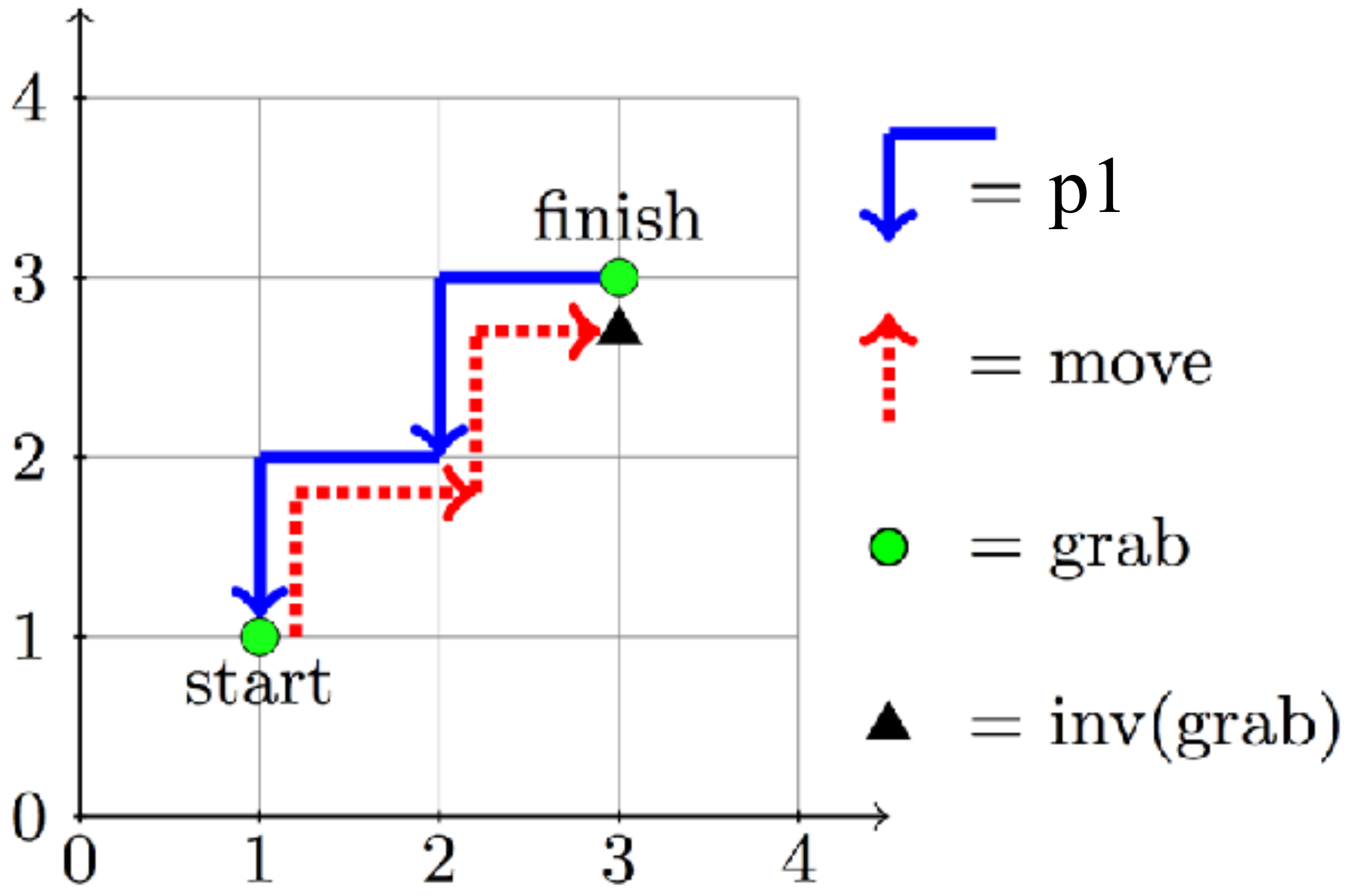


```
move(A,B):- p3(A,C),drop(C,B).
p3(A,B):- grab(A,C), p2(C,B).
p2(A,B):- p1(A,C),p1(C,B).
p1(A,B):- forward(A,C),right(C,B).
```

Robot moving a ball - missing actions

left/2
~~right/2~~
~~forwards/2~~
backwards/2
grab/2
~~drop/2~~

Plan learned with MIL



move(A,B):- grab(A,C), p4(C,B).
p4(A,B):- p3(B,C).
p3(A,B):- p2(A,C), p1(C,B).
p2(A,B):- grab(A,C), p1(C,B).
p1(A,B):- left(A,C), back(C,B).

Conclusions

- Predicate invention can compensate for incomplete background information
- Metagol (an MIL implementation) supports predicate invention
- Suggests motivation to purposely predicates to improve efficiency, analogous to dimensionality reduction

Future work

- Naming invented predicates
- Automate removal of redundant background predicates

Related work

Missing data (feature based ML)
- Ghahramani & Jordan (1995)
- Marlin (2006)

Incomplete background knowledge
- Srinivasan, et al.,(1995)
- Muggleton(2011)

Effect of missing predicates
- Liu and Zhong (1999)

Compensating for incomplete background knowledge
- Dzeroski (1993)

Dimensionality reduction
- Furnkranz (1997)

Thank you