Constraint programming for inductive logic programming

Andrew Cropper, Céline Hocquette

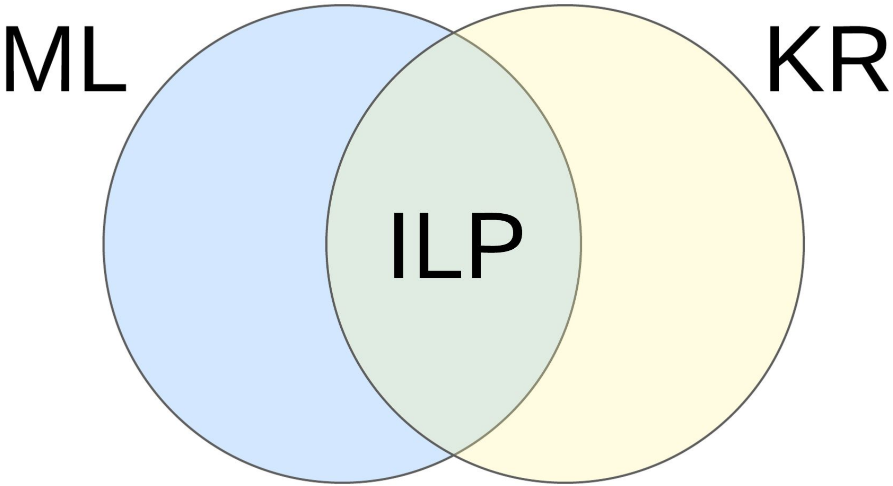
University of Oxford

UK Research and Innovation

UNIVERSITY OF OXFORD

Inductive Logic Programming (ILP)

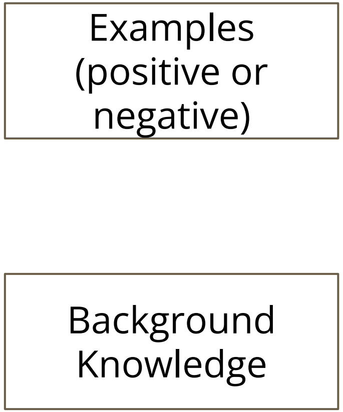
Inductive Logic Programming



Inductive Logic Programming

Examples
(positive or
negative)

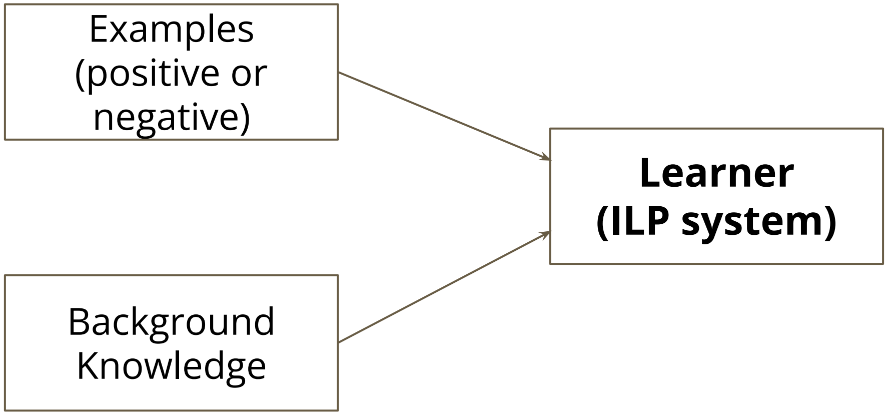
Inductive Logic Programming



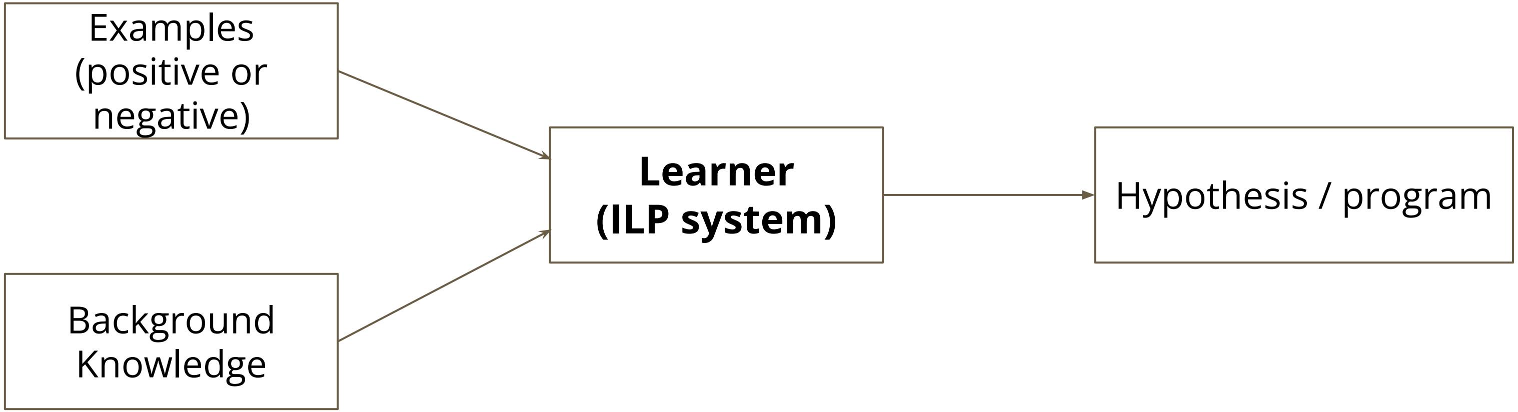
Examples (positive or negative)

Background Knowledge

Inductive Logic Programming



Inductive Logic Programming



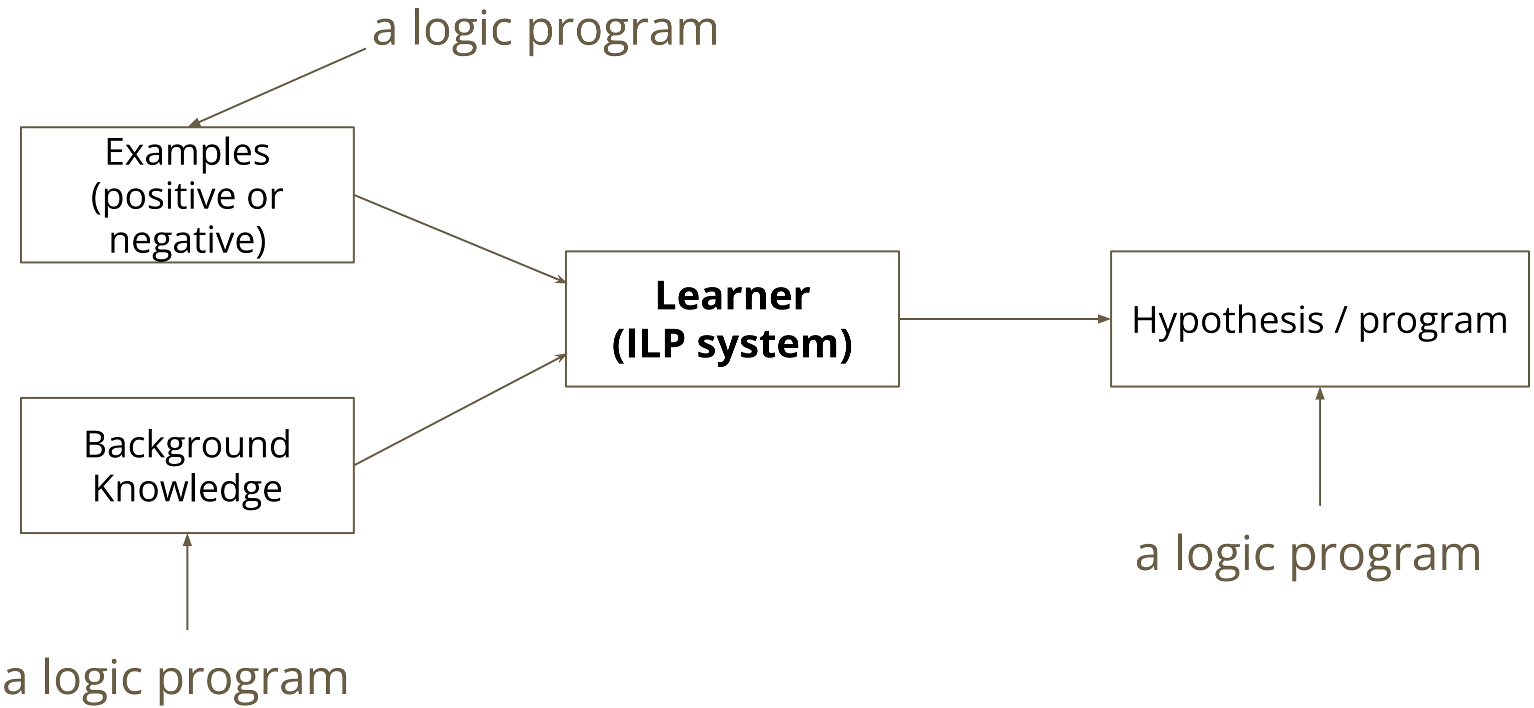
Examples (positive or negative)

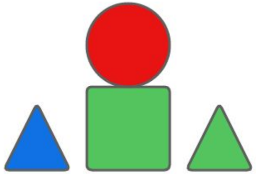
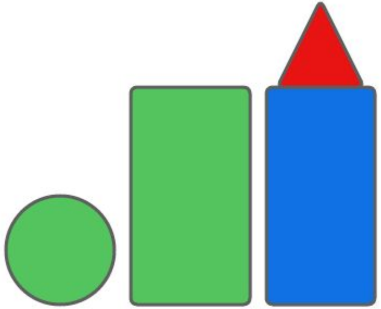
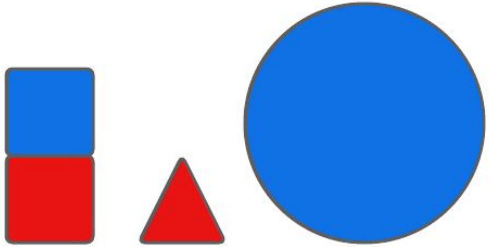
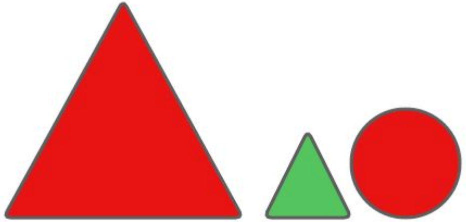
Background Knowledge

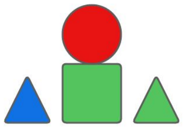
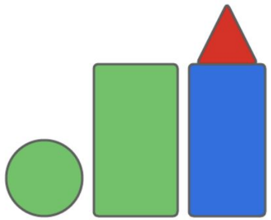
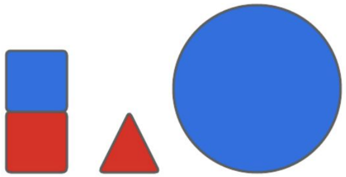
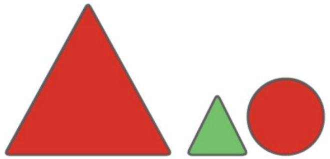
Learner (ILP system)

Hypothesis / program

Inductive Logic Programming

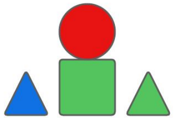


Positive examples	Negative examples
	
	

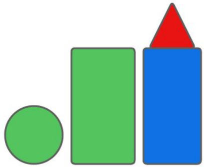
Positive examples	Negative examples
	
	

There must be a red piece in contact with a small piece

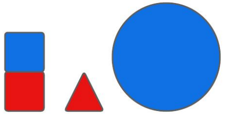
Positive examples	Negative examples
zendo(ex1). zendo(ex2).	zendo(ex3). zendo(ex4).



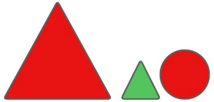
ex1



ex3



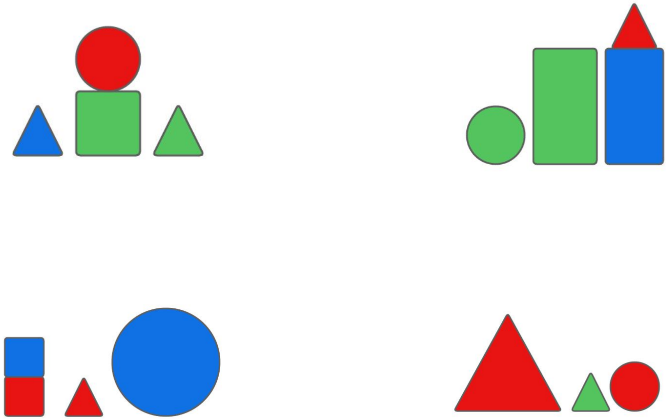
ex2



ex4

Background Knowledge

```
piece(ex1, p1).
piece(ex1, p2).
piece(ex1, p3).
piece(ex1, p4).
blue(p1).
triangle(p1).
size(p1, 2).
small(2).
red(p2).
round(p2).
triangle(p4).
contact(p2, p3).
on(p2, p3).
right(p4, p3).
left(p1, p2).
…
```



Hypothesis

```
zendo(Structure):-
    piece(Structure,Piece1),
    red(Piece1),
    contact(Piece1,Piece2),
    size(Piece2,Size),
    small(Size).
```

Why care about ILP?

Why care about ILP?

- Learn from small amount of data

Why care about ILP?

- Learn from small amount of data
- Learn explainable models

Why care about ILP?

- Learn from small amount of data
- Learn explainable models
- Learn from relational data

Why care about ILP?

- Learn from small amount of data
- Learn explainable models
- Learn from relational data
- ILP can be applied to many problems
  - robot scientist, biology, learning game strategies

In this presentation

Popper: an inductive logic programming system

Learning logic programs by combing programs, Andrew Cropper and Céline Hocquette, ECAI, 2023.
Learning MDL logic programs from noisy data, Céline Hocquette, Andreas Niskanen, Matti Järvisalo, and Andrew Cropper, AAAI, 2024.

Why care?

Why care?

- Popper formulates the ILP problem as a CP problem

Why care?

- Popper formulates the ILP problem as a CP problem

- Challenges for the CP community to address limitations

Why care?

- Popper formulates the ILP problem as a CP problem
- Challenges for the CP community to address limitations
- Benchmarks tasks for the CP community

Why care?

- Popper formulates the ILP problem as a CP problem

- Challenges for the CP community to address limitations

- Benchmarks tasks for the CP community

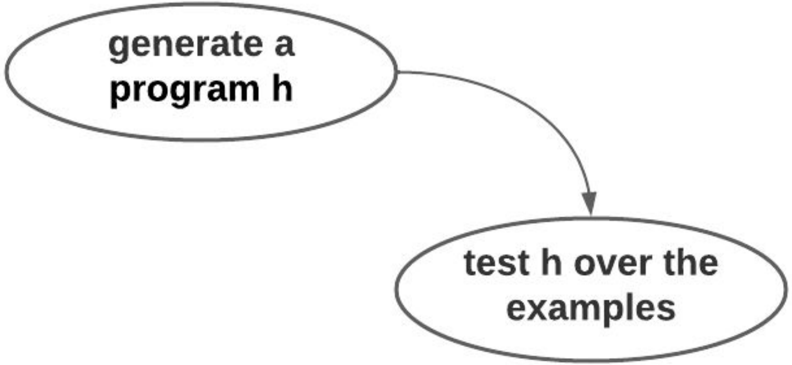
- Accessible way to bridge CP and ML

How does Popper work?

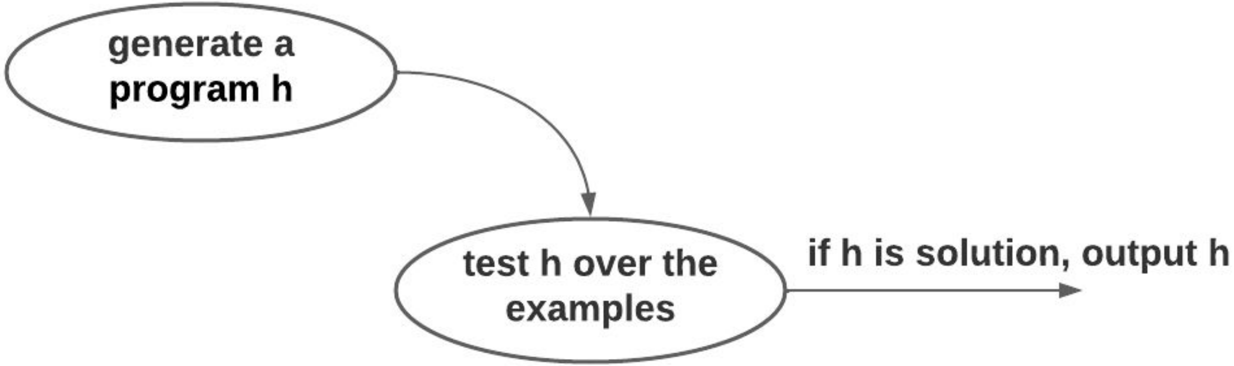
How does Popper work?

generate a program h

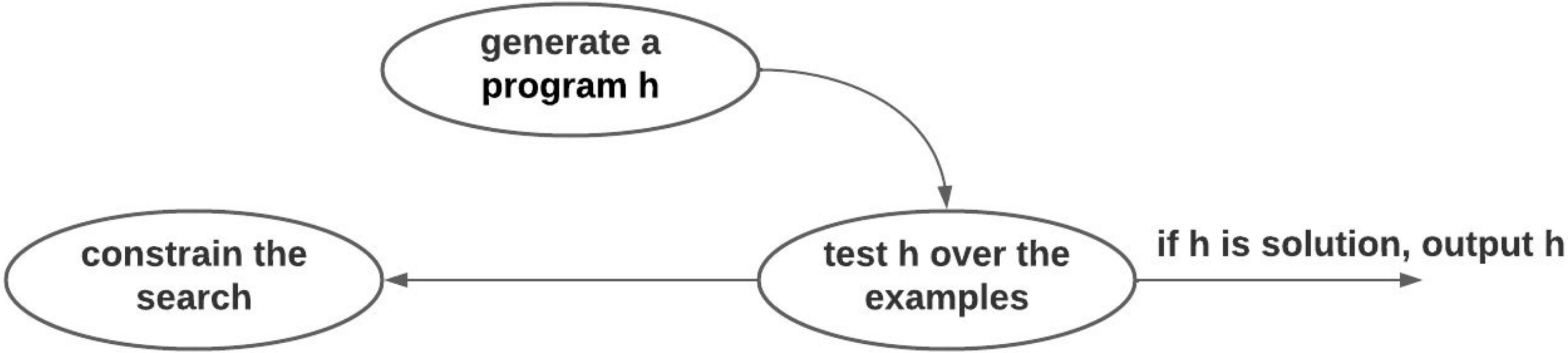
How does Popper work?



How does Popper work?



How does Popper work?



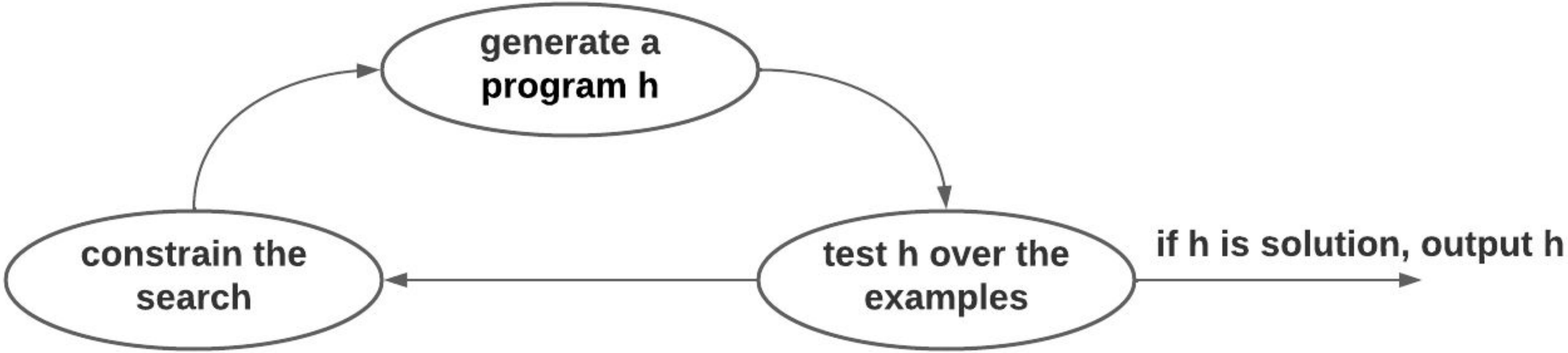
generate a program h

test h over the examples

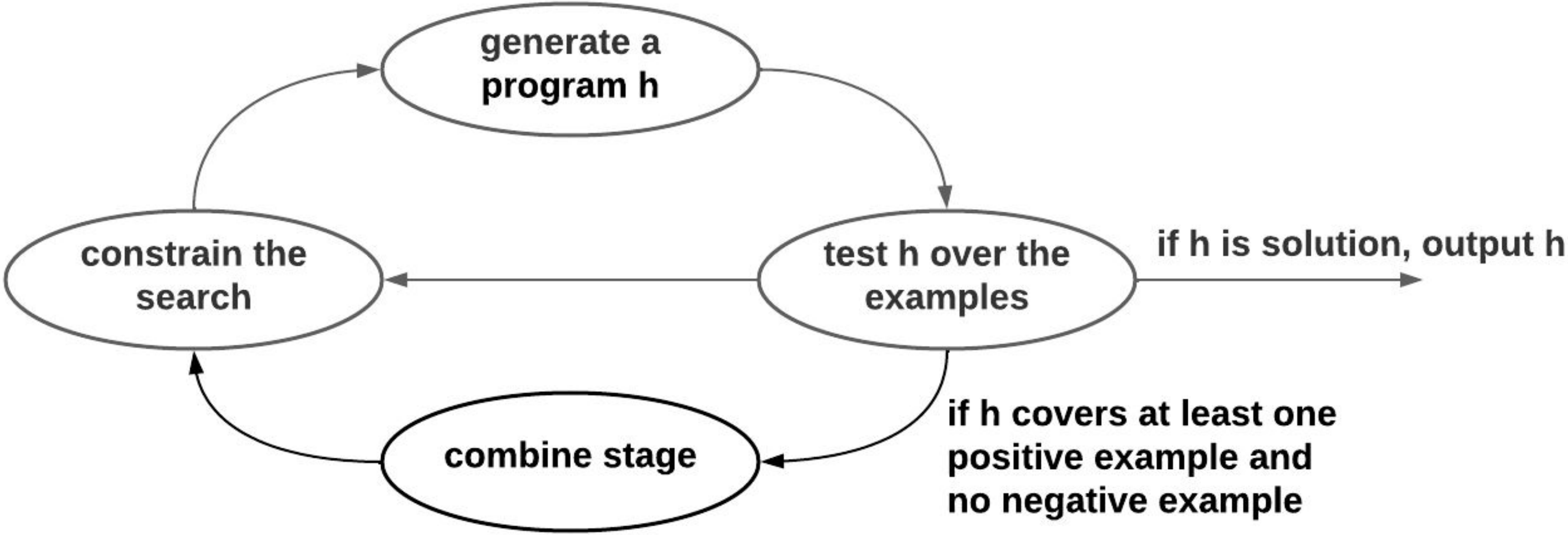
if h is solution, output h

constrain the search

How does Popper work?



How does Popper work?


generate a program h
test h over the examples
if h is solution, output h
if h covers at least one positive example and no negative example
combine stage
constrain the search

Learning programs by combining programs, Andrew Cropper and Céline Hocquette, *ECAI, 2023*.

Generate stage

Generate stage

Input:

- a set of literals L
- a set of constraints C

Generate stage

Input:

- a set of literals L
- a set of constraints C

Output: a set of literals L’⊂L such that:

- L’ is consistent with C
- L’ is minimal in size

Generate stage

Input:

```
{piece(A,B),red(B),blue(B),small(B),red(C),blue(C),small(C),red(D),
blue(D),small(D),contact(B,C),contact(C,B),contact(B,D),
contact(D,B),contact(C,D),contact(D,C)}
```

Generate stage

Input:

```
{piece(A,B),red(B),blue(B),small(B),red(C),blue(C),small(C),red(D),
blue(D),small(D),contact(B,C),contact(C,B),contact(B,D),
contact(D,B),contact(C,D),contact(D,C)}
```

Output:

```
{piece(A,B),red(B)}
```

Generate stage

Input:

```
{piece(A,B),red(B),blue(B),small(B),red(C),blue(C),small(C),red(D),
blue(D),small(D),contact(B,C),contact(C,B),contact(B,D),
contact(D,B),contact(C,D),contact(D,C)}
```

Output:

```
{piece(A,B),red(B)}
```

```
zendo(A) ← piece(A,B),red(B)
```

Generate stage

We currently use ASP

Generate stage

We currently use ASP

- easy for us

Generate stage

We currently use ASP

- easy for us
- easy to express recursive concepts (connectedness)

Generate stage

We currently use ASP

- easy for us
- easy to express recursive concepts (connectedness)
- incremental solving

Combine stage

Combine stage

Input:

- a set of programs P, with their size and coverage, such that for all p∈P:
    - p covers at least one positive example
    - p does not cover any negative example

Combine stage

Input:

- a set of programs P, with their size and coverage, such that for all p∈P:
    - p covers at least one positive example
    - p does not cover any negative example

Output: a set of programs P’⊂P such that:

- P’ covers as many positive examples as possible
- P’ is minimal in size

Combine stage

Input:

Program	Positive examples covered	Size
p1	{e1,e2,e3}	3
p2	{e9}	3
p3	{e1,e3,e5,e6,e7}	4
p4	{e2,e6,e7}	4
p5	{e2,e5,e8,e9}	5
p6	{e8,e9}	6

Combine stage

Input:

Program	Positive examples covered	Size
p1	{e1,e2,e3}	3
p2	{e9}	3
p3	{e1,e3,e5,e6,e7}	4
p4	{e2,e6,e7}	4
p5	{e2,e5,e8,e9}	5
p6	{e8,e9}	6

Output:
{p1,p3,p5} covers {e1,e2,e3,e5,e6,e7,e8,e9} and has size 12

Combine stage

We used ASP and switched to MaxSAT

Combine stage

We used ASP and switched to MaxSAT

We can support noise

Thursday 22nd, 2:00-3:15
Knowledge Representation

Learning MDL logic programs from noisy data, Céline Hocquette, Andreas Niskanen, Matti Järvisalo, and Andrew Cropper, AAAI, 2024.

Why not one big SAT/ASP problem?

Why not one big SAT/ASP problem?

- infinite domains, function symbols (lists), numerical reasoning

Why not one big SAT/ASP problem?

- Infinite domains, function symbols (lists), numerical reasoning

- The problem quickly becomes infeasible

Conclusion

- Popper, an ILP algorithm which uses CP

Limitations: generate stage

Limitations: generate stage

- The generate stage can be prohibitively slow and it prevents us to use Popper on some tasks

Limitations: generate stage

- The generate stage can be prohibitively slow and it prevents us to use Popper on some tasks

Can our ASP encoding be improved?

Limitations: generate stage

- The generate stage can be prohibitively slow and it prevents us to use Popper on some tasks

Can our ASP encoding be improved?
Would a different CP approach be more suitable?

Limitations: combine stage

- We use UWRMaxSAT.

Limitations: combine stage

- We use UWRMaxSAT. Can your solver / encoding do better?

Limitations: combine stage

- We use UWRMaxSAT. Can your solver / encoding do better?
    - Currently single-threaded

Limitations: combine stage

- We use UWRMaxSAT. Can your solver / encoding do better?
    - Currently single-threaded
    - Currently non-incremental

Limitations: combine stage

- We use UWRMaxSAT. Can your solver / encoding do better?
    - Currently single-threaded
    - Currently non-incremental
    - Struggles with weights

Benchmarks

We have hard and large (1gb+) instances if you want to try!

Thank you!

https://github.com/logic-and-learning-lab/Popper

celine.hocquette@cs.ox.ac.uk
andrew.cropper@cs.ox.ac.uk