Learning logic programs by discovering higher-order abstractions

Céline Hocquette, Sebastijan Dumančić, and Andrew Cropper

UNIVERSITY OF OXFORD

UK Research and Innovation

TU Delft
Delft University of Technology

Program Synthesis

Input	Output
ijcai	IJCAI
program	PROGRAM

Program Synthesis

Input	Output
ijcai	IJCAI
program	PROGRAM

First-order program

```
f(Input,Output) ←
   empty(Input),empty(Output).
f(Input,Output) ←
    head(Input,Head1), tail(Input,Tail1),
    head(Output,Head2), tail(Output,Tail2),
    uppercase(Head1,Head2), f(Tail1,Tail2).
```

Program Synthesis

Input	Output
ijcai	IJCAI
program	PROGRAM

First-order program

```
f(Input,Output) ←
   empty(Input),empty(Output).
f(Input,Output) ←
    head(Input,Head1), tail(Input,Tail1),
    head(Output,Head2), tail(Output,Tail2),
    uppercase(Head1,Head2), f(Tail1,Tail2).
```

Second-order program

```
f(Input,Output) ←
   map(uppercase,Input,Output).
```

Challenge

Automatically discover abstractions such as *map*, *fold*, *filter*, ...

Our approach

We introduce an approach that automatically discovers useful higher-order abstractions.

Our approach

We implement this approach as a constraint optimisation problem, where we compress a logic program.

1. abstract stage: we build higher-order abstractions
2. compress stage: we search for a subset of the abstractions that compresses the program

Our results

We can rediscover usual abstractions such as *map*, *filter*, and *fold*.

Our results

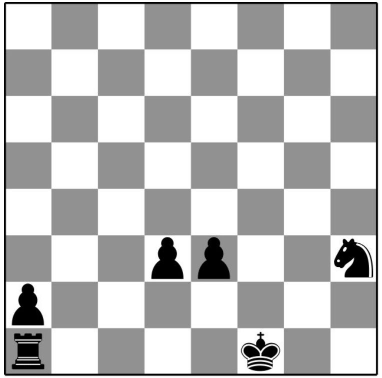
Discovering abstractions can:

- improve predictive accuracies by **27%**
- reduce learning times by **47%**

Our results

Discovered abstractions can be transferred to different domains

Input	Output
ijcai	IJCAI
program	PROGRAM



Thank you!

Poster 2677