Learning big logical rules by joining small rules

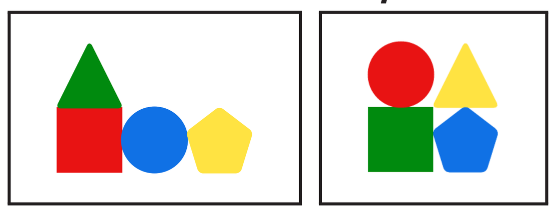
Céline Hocquette[1], Andreas Niskanen[2], Rolf Morel[1], Matti Järvisalo[2], Andrew Cropper[1]
[1]University of Oxford; [2]University of Helsinki
celine.hocquette@cs.ox.ac.uk

1 - Introduction

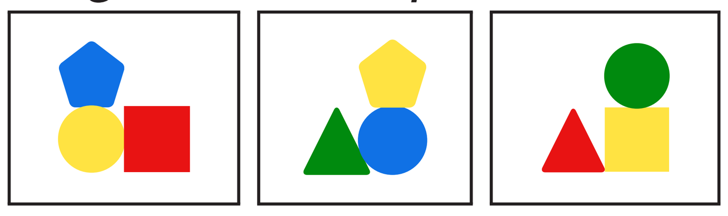
The goal of inductive logic programming (ILP) is to induce a program (a set of logical rules) that generalises training examples.

Problem: learning programs with big rules is difficult.

Example 1 (Zendo)
Positive examples:



Negative examples:



zendo(S) ← piece(S,B), blue(B),
piece(S,R), red(R),
piece(S,G), green(G)

We introduce an approach where we join small rules to learn big rules.

We first search for rules that entail at least one positive example, such as:

$zendo_1$(S) ← piece(S,B), blue(B)
$zendo_2$(S) ← piece(S,R), red(R)
$zendo_3$(S) ← piece(S,G), green(G)
$zendo_4$(S) ← piece(S,Y), yellow(Y)

We then search for subsets of these rules which entail at least one positive example and no negative examples:

zendo(S) ← $zendo_1$(S), $zendo_2$(S), $zendo_3$(S)

Example 2 (List classification)

Positive	*Negative*
f([a,b,c,d])	*f([a,c,d,e])*
f([c,b,d,e])	*f([c,b])*
	f([d,b])

We first learn programs that entail at least one positive example:

{ f_1(List) ← head(List,a) }
{ f_2(List) ← head(List,c) }
{ f_3(List) ← tail(List,Tail),head(Tail,b) }
{ f_4(List) ← head(List,c)
f_4(List) ← tail(List,Tail), f_4(Tail) }
{ f_5(List) ← head(List,d)
f_5(List) ← tail(List,Tail), f_5(Tail) }

We then search for subsets of these rules which entail at least one positive example and no negative examples:

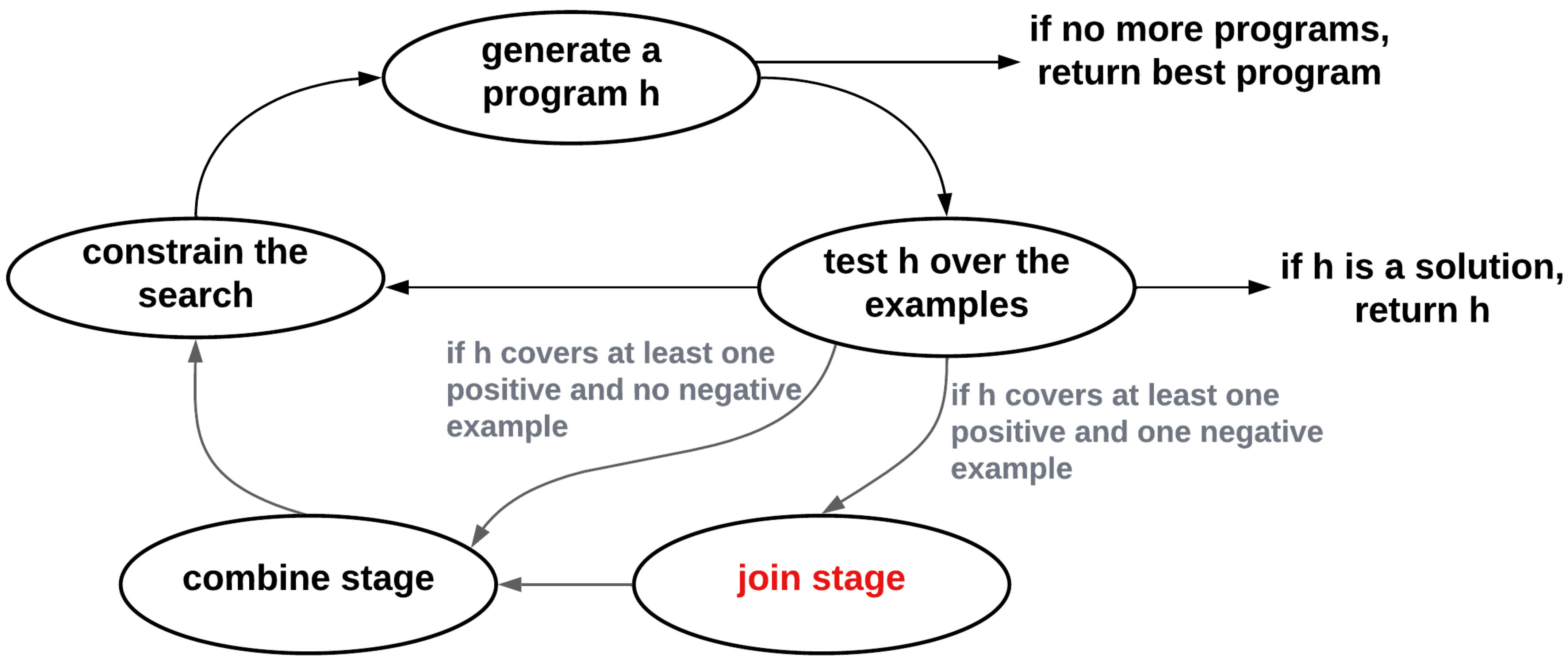
f(List) ← f_3(List), f_4(List), f_5(List)

3 - Theoretical Analysis

Theorem JOINER learns an optimal solution (a program with minimal size).

2 - Our approach (JOINER)

Key idea: learn small programs independently and then try to find conjunctions of these programs which cover no negative examples.



We develop a Boolean satisfiability approach to find conjunctions in the join stage.

4 - Experiment

Q1 Can the join stage improve learning performance?
Q2 How well does JOINER scale with the size of rules?
Q3 How well does JOINER compare against other approaches?

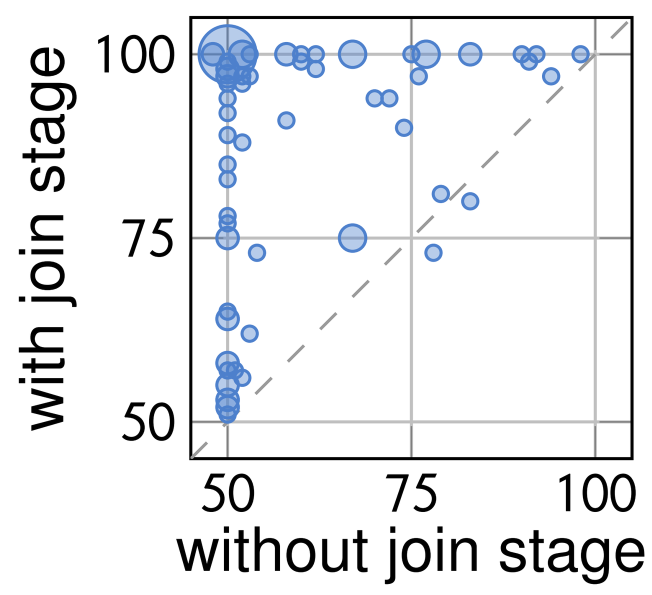


Fig. 1: Predictive accuracy (%) with and without join stage.

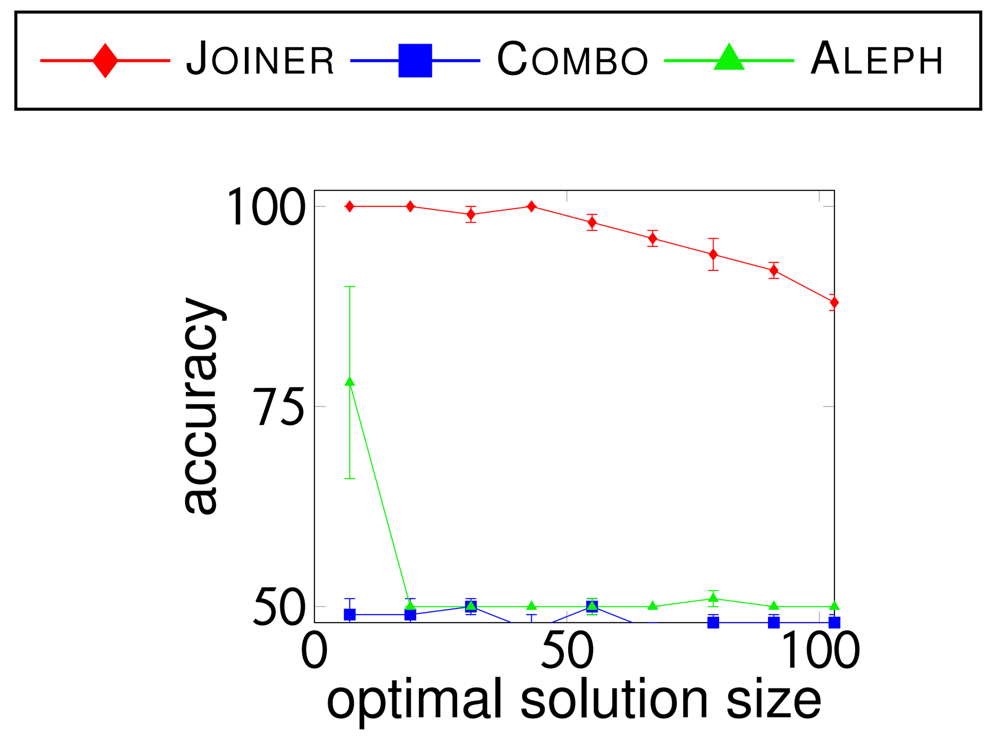


Fig. 2: Predictive accuracy versus the size of programs for *zendo*.

Task	ALEPH	COMBO	JOINER
iggp	78 ± 3	86 ± 2	**96 ± 1**
zendo	**100 ± 0**	86 ± 3	94 ± 2
pharma	50 ± 0	53 ± 2	**98 ± 1**
imdb	67 ± 6	**100 ± 0**	**100 ± 0**
string	50 ± 0	50 ± 0	**100 ± 0**
onedarc	51 ± 1	57 ± 2	**89 ± 1**

Table 1: Predictive accuracies (%).

Q1 The join stage can substantially improve predictive accuracies.
Q2 JOINER can learn rules with more than 100 literals.
Q3 JOINER can outperform existing approaches in terms of predictive accuracies.

5 - Conclusion and Limitation

- **An approach which learns big rules by joining small rules.**

Future work: noisy examples.

Article Code