Inductive Logic Programming at Scale

David M. Cerna

July 7^{th} 2026

Inductive Logic Programming (ILP)

- **ILP?** symbolic machine learning.
- Introduced in the early 90s (Muggleton, 1991).
- **Goal:** Form an explanatory hypothesis using:
  1) Positive and negative **evidence**
  2) Provided **background knowledge**

Background Knowledge (BK)

$\texttt{mom}(a, b).$ $\texttt{dad}(g, b).$
$\texttt{mom}(a, c).$ $\texttt{dad}(g, c).$
$\texttt{mom}(b, d).$ $\texttt{dad}(f, d).$
$\texttt{mom}(e, f).$ $\texttt{dad}(c, h).$

Evidence

$\texttt{grandparent}(a, d)^{+}.$
$\texttt{grandparent}(g, d)^{+}.$
$\texttt{grandparent}(a, h)^{+}.$
$\texttt{grandparent}(g, h)^{+}.$
$\texttt{grandparent}(a, e)^{-}.$

- $\texttt{mom}(a, b)$ denotes a is the mom of b.

Grandparent Example

- From *mom/2* and *dad/2* we learn a **logic program** for

  X is a grandparent of Y

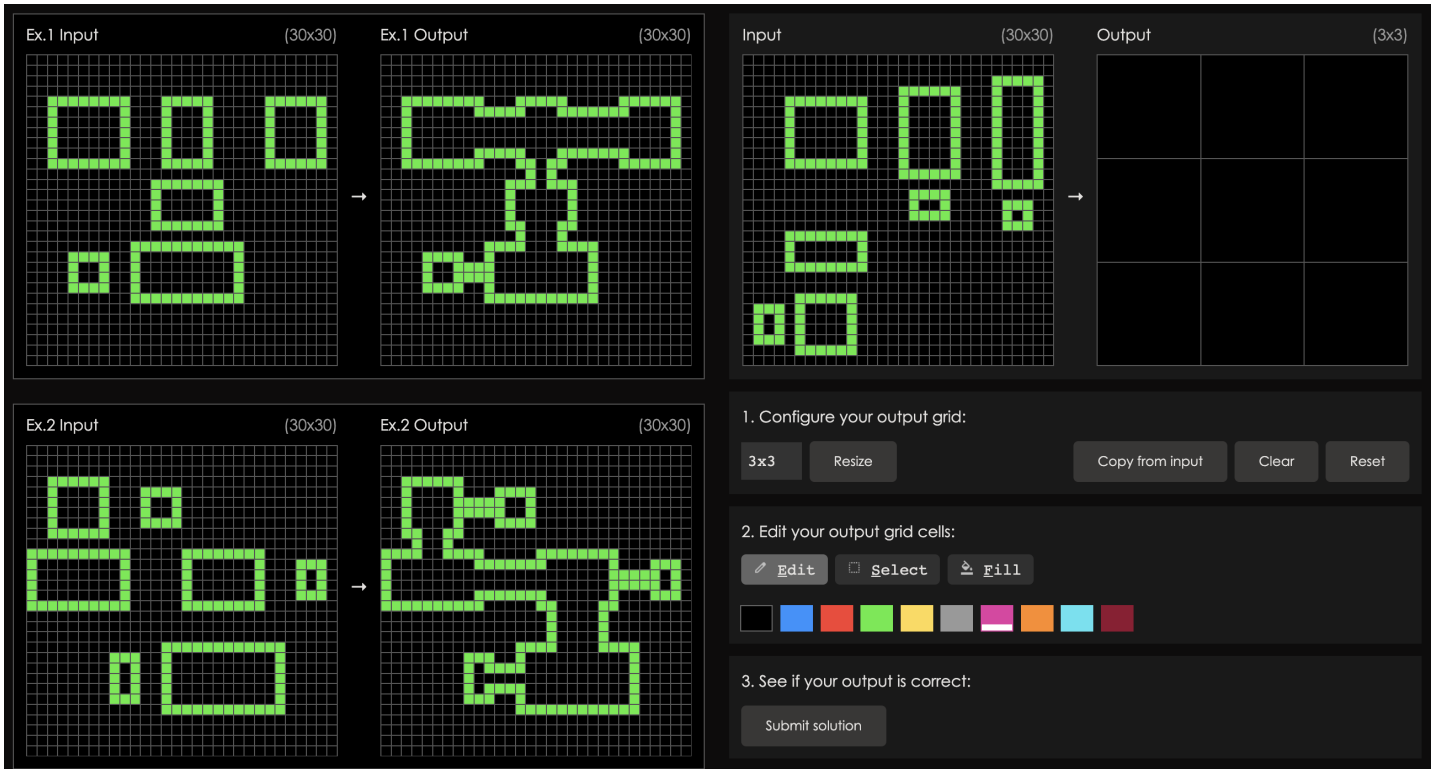
- Solutions include the following:

$$\begin{array}{l} \texttt{gp}(X,Y)\texttt{:-mom}(X,C),\texttt{mom}(C,Y) \\ \texttt{gp}(X,Y)\texttt{:-mom}(X,C),\texttt{dad}(C,Y) \\ \texttt{gp}(X,Y)\texttt{:-dad}(X,C),\texttt{mom}(C,Y) \\ \texttt{gp}(X,Y)\texttt{:-dad}(X,C),\texttt{dad}(C,Y) \end{array} \quad \Big| \quad \begin{array}{l} \texttt{gp}(X,Y)\texttt{:-}\mathbf{p}(X,C),\mathbf{p}(C,Y) \\ \mathbf{p}(X,Y)\texttt{:-mom}(X,Y) \\ \mathbf{p}(X,Y)\texttt{:-dad}(X,Y) \end{array}$$

- **Goal:** Find **H** such that:

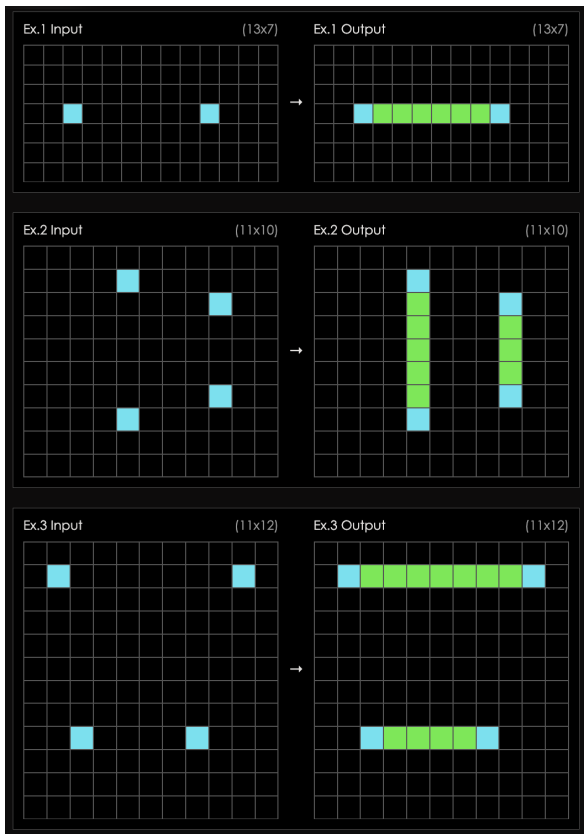
$$BK, \mathbf{H} \models E^{+} \qquad BK, \mathbf{H} \not\models E^{-}$$

Why ILP?



- https://arcprize.org/arc-agi/1
- Answer almost immediate to a human.
- Costly to solve via reasoning models.

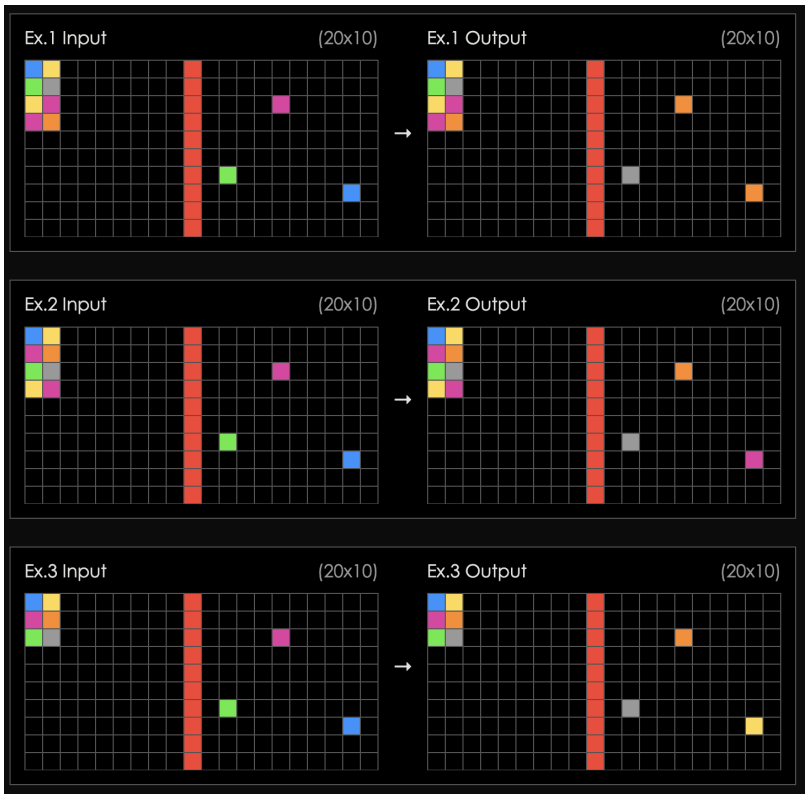
Via ILP

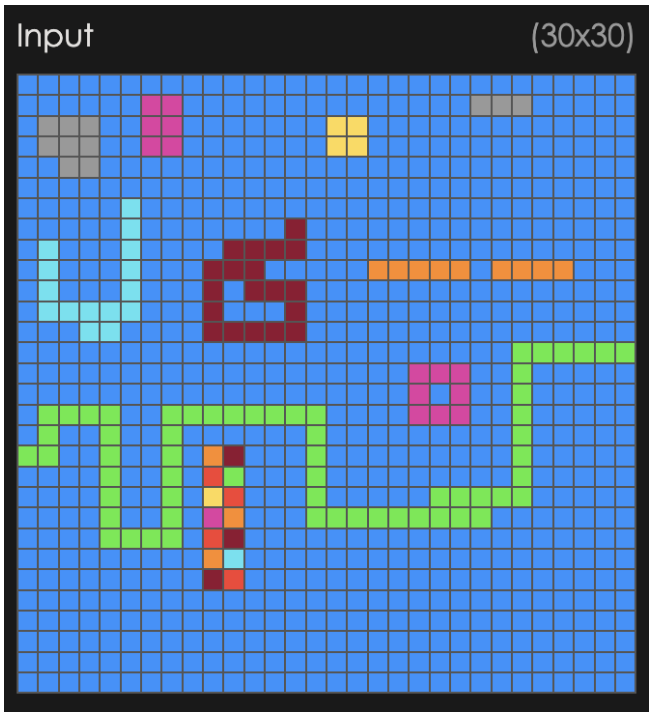


- A solution to this task can be learned using the ILP System **Popper** *(Cropper and Morel, 2021)*
- **22%** *on training:* Single core, generic BK, relational encoding.
- *"See Relational Decomposition for Program Synthesis"*, Hocquette and Cropper, 2025

```
out(X,Y,blue):- in(X,Y,blue).
out(X,Y,green):- in(X,Y1,blue), in(X,Y2,blue), Y1<Y<Y2.
out(X,Y,green):- in(X1,Y,blue), in(X2,Y,blue), X1<X<X2.
```

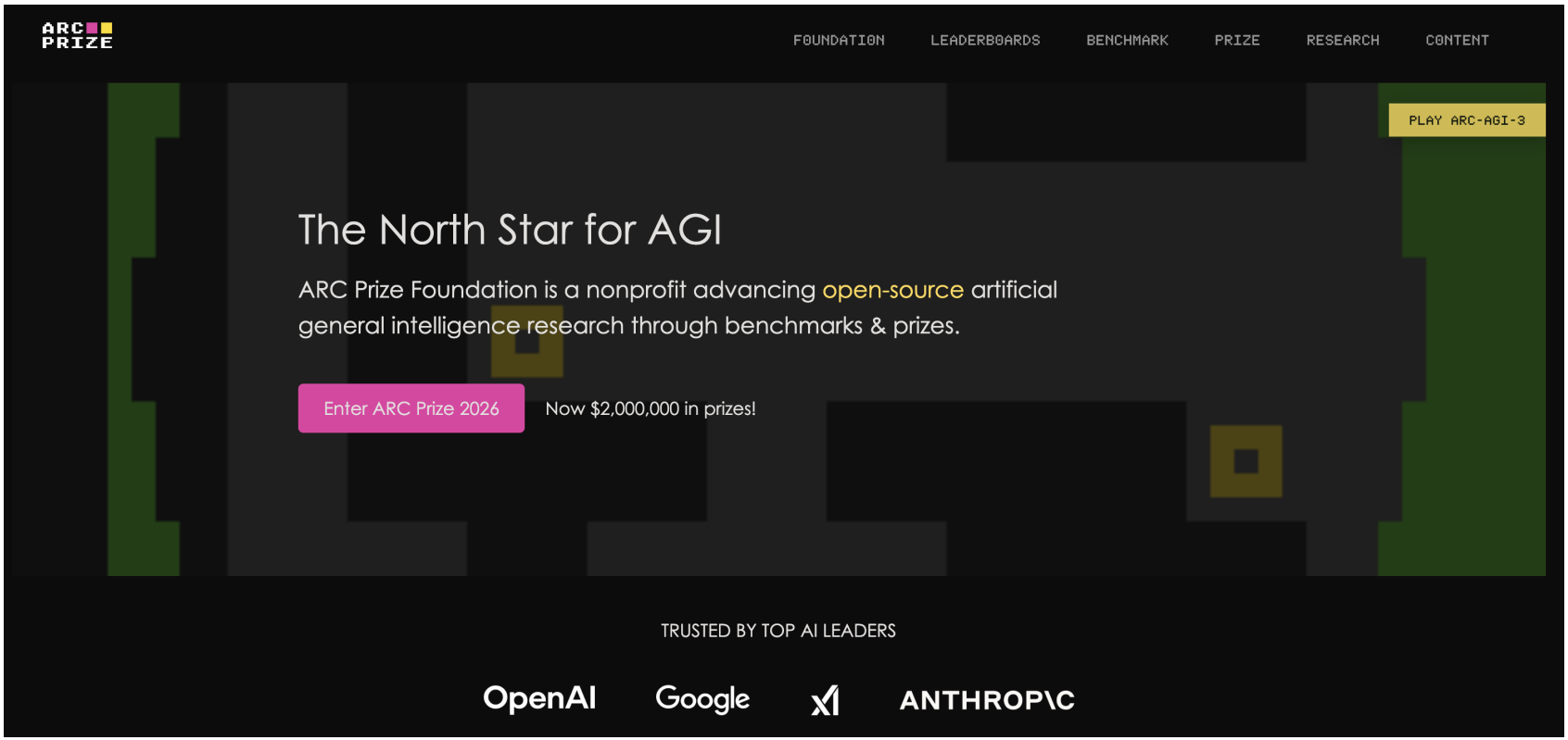
Scaling ILP: ARC-AGI-1, 2, 3, Beyond





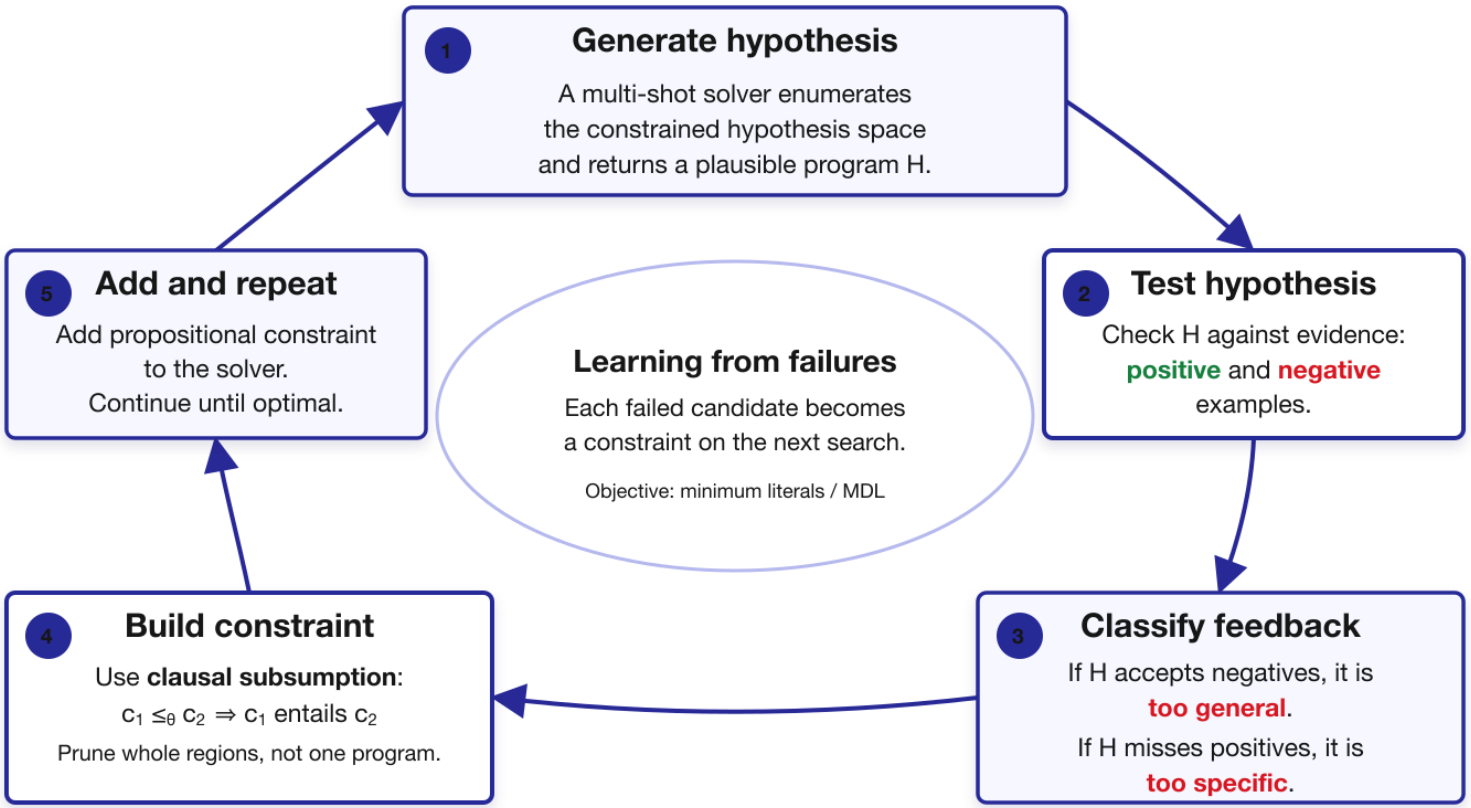
- **Training** on the left, **Test** on the right.
- https://arcprize.org/arc-agi/3

Arc Prize



- Arc challenge currently running.
- Participate: arcprize.org/

Popper: Learning from Failures



- Finds optimal solutions (Min literal count, MDL)
- **Issue:** Full of redundancy. Efficient search is needed.

New Constraints: Pointless rules

- Literals in the body of rules may imply each other.

$$r_1: \quad h(A) \text{ :- } odd(A), int(A).$$
$$r_2: \quad h(A) \text{ :- } odd(A).$$

- Observe, $odd(A) \Rightarrow int(A)$.
- Thus, keeping r_1 in the search space is pointless.

Definition (**Reducible**)

Let r be a rule, B be BK, $l \in body(r)$ be r-captured, and $B \models (body(r) \setminus \{l\}) \rightarrow l$. Then r is *reducible*.

Indiscriminate rules

- No implications present in the following rule:

$$\begin{aligned} r_1: &\quad h(A) \text{ :- } odd(A), lt(A, 10). \\ r_2: &\quad h(A) \text{ :- } odd(A). \\ & E^- = \{h(1), h(2), h(3)\} \end{aligned}$$

- Observe, $lt(A, 10)$ accepts all of E^-, i.e. r_1 and r_2 accept the same members of E^-.

Definition (**Indiscriminate**)

Let r be a rule, B be BK, E^- be negative examples with the same predicate symbol as the head of r, $l \in body(r)$ be captured, and for all $e \in E^-$, $B \models (body(r) \setminus \{l\})\theta_e^r \rightarrow l\theta_e^r$. Then r is *indiscriminate*.

* where θ_e^r is a substitution with domain $vars(head(r))$ such that $head(r)\theta_e^r = e$.

Preprocessing: Recall Reducible rules

- No implications present in the following rules:

$$
\begin{aligned}
BK &= \{edge(a,b), edge(b,c), edge(c,a)\} \\
r_1: &\quad h \text{ :- } edge(A,B), edge(B,C), edge(C,D), edge(D,E). \\
r_2: &\quad h \text{ :- } edge(A,B), edge(B,C), edge(C,A). \\
\theta &= \{D \mapsto A, E \mapsto B\}
\end{aligned}
$$

- Observe, that $r_1 \preceq_\theta r_2$ as $r_1\theta = r_2$, $|r_1| > |r_2|$, and $B \models r_1 \rightarrow r_2$ follows from $r_1 \preceq_\theta r_2$.
- Furthermore, $B \models r_2 \rightarrow r_1$ follows from *edge/2* being a bijective mapping from $\{a, b, c\}$ to itself.
- That is **"specializations"** of r_2 by adding an edge literal cannot be optimal.
- Such rules are referred to as Recall Reducible.

Beyond Definite Clause Subsumption

Definition (**Recall reducible**)

Let B be BK, r_1 be a rule, $r_1 \preceq_\theta r_2$, $|r_1| > |r_2|$, and $B \models r_1 \leftrightarrow r_2$. Then r_1 is recall reducible.

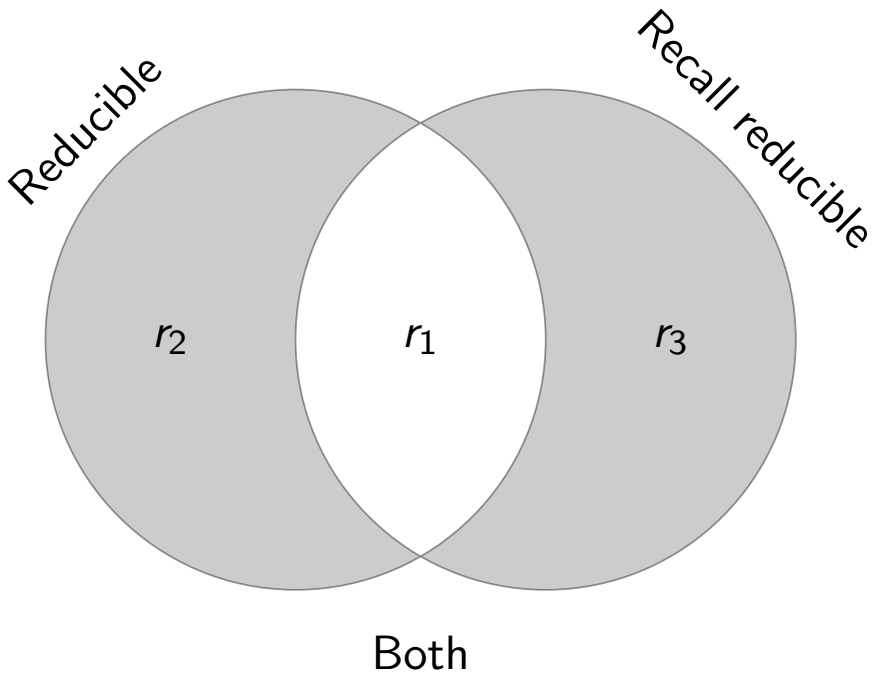
Proposition (**Recall specialisation**)

Let B be BK, r_1 be recall reducible, and $r_1 \subseteq r_2$. Then r_2 is recall reducible.

Proof.

We show that recall reducibility is equivalent to recognizing pigeonhole arguments in the definitions of the BK. □

Comparision to Reducible/Indiscriminate



r_1 : h :- $leq(B, C), succ(A, B), succ(A, C).$

r_2 : h :- $nat(A), succ(A, B).$

r_3 : h :- $succ(A, B), succ(A, C), odd(B), prime(C).$

r_4 : h :- $leq(B, B), succ(A, B).$

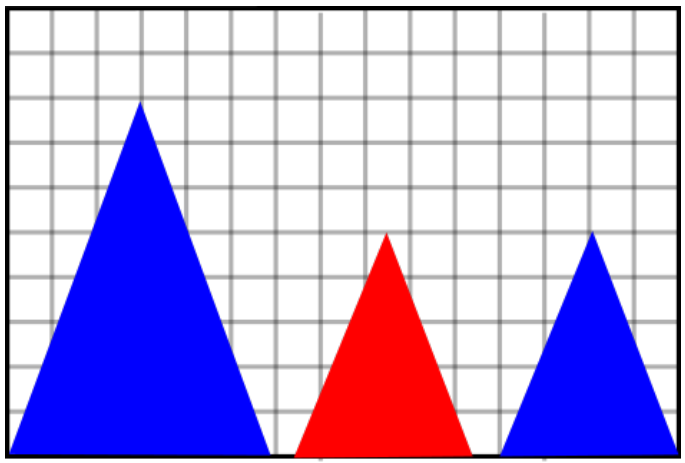
r_5 : h :- $succ(A, B), succ(A, C).$

r_6 : h :- $succ(A, B).$

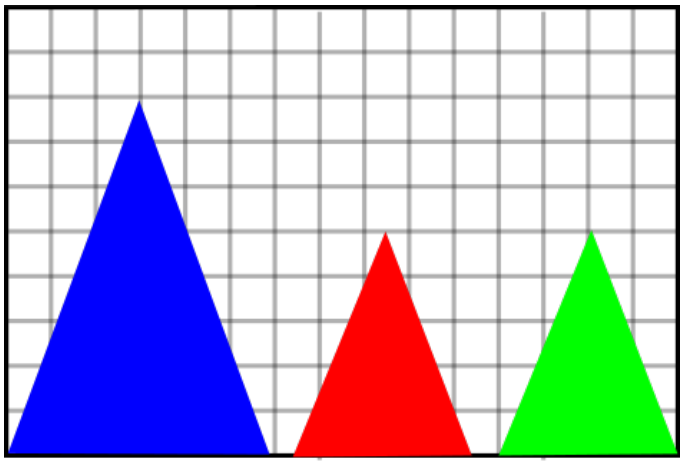
r_7 : h :- $succ(A, C), odd(C), prime(C).$

- All show exponential decrease in search on standard datasets.

Symmetries Breaking for ILP



E^+



E^-

$r_1 = zendo(A) \leftarrow piece(A,B),\ size(B,C),\ blue(B),\ small(C)$

$r_2 = zendo(A) \leftarrow piece(A,C),\ size(C,B),\ blue(C),\ small(B)$

- r_1 and r_2 are **variants**.
- How to quickly (and efficiently) recognize this?

Precise Problem

Definition (**Hypothesis space reduction problem**)

Given an ILP input $(E, B, \mathcal{H})$, the *hypothesis space reduction problem* is to find $\mathcal{H}' \subseteq \mathcal{H}$ such that if $\mathcal{H}$ contains an optimal hypothesis then there exists an optimal hypothesis $h \in \mathcal{H}'$.

- For example, removing body variants.

Definition (**Body variant**)

A rule r' is a *body-variant* of a rule r if there exists a bijective renaming σ from $body\_vars(r)$ to $body\_vars(r')$ such that $r\sigma = r'$.

Proposition (**Body-variant hardness**)

The body-variant problem is GI-hard.

- When can we solve the variant problem efficiently?

Ordering Variable Arguments

Definition (**Witnessed**)

Let r be a rule, v be a variable, $l_1 \in body(r)$ such that $v \in skipped(l_1)$, and $l_2 \in body(r)$ such that $v \in vars(l_2)$ and $l_2 <_{lex} l_1$. Then we say that l_1 is *v-witnessed* in r.

- In $p(A, C, E)$ the variables B and D are skipped.
- In r_1, $p(A, E)$ is not C-witnessed.
- In r_2, $p(B, D)$ is C-witnessed and $p(C, E)$ is E-witnessed.

$$r_1 : \quad h(A, B) \text{ :- } p(A, E), p(B, C), p(C, D).$$
$$r_2 : \quad h(A, B) \text{ :- } p(A, C), p(B, D), p(C, E).$$

Definition (**Safe variable**)

Let r be a rule and $v \in body\_vars(r)$ such that for all $l \in body(r)$, where $v \in skipped(l)$, l is v-witnessed in r. Then v is *safe*.

Soundness modulo Variants

Proposition (**Soundness**)

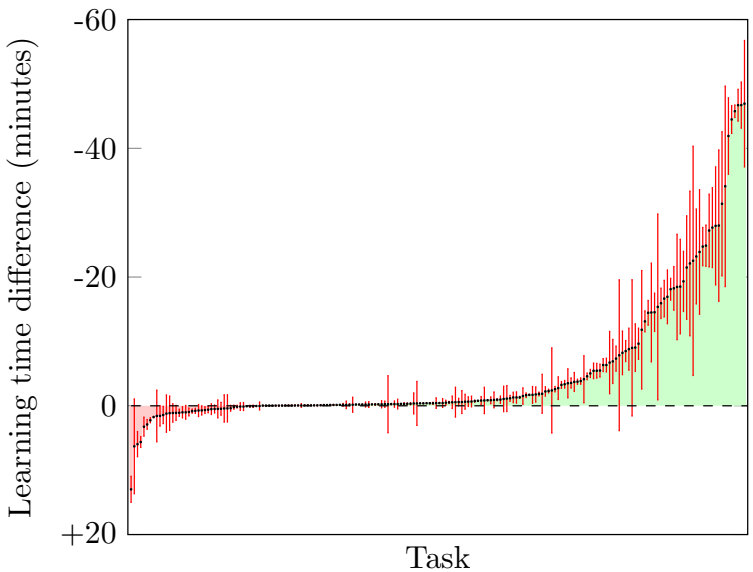
For every rule r there exists a rule r' such that r' is a body-variant of r and all variables in r' are safe.

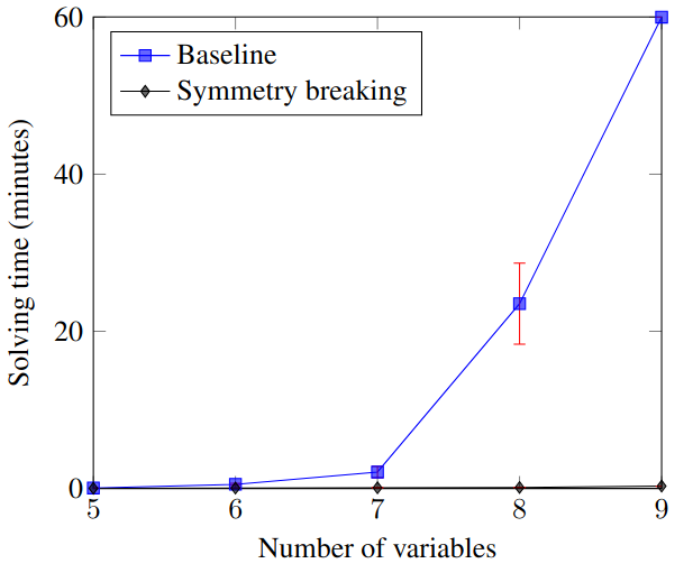
Proof.

Proof by induction. Selecting the smallest unsafe variable x and construct a substitution that, when applied to r, results in a rule where the smallest unsafe variable is larger than x. □

$$
\begin{aligned}
r_1: \quad & h(A,B) \text{ :- } p(A,E), p(B,C), p(\mathbf{C},D). \\
& \qquad \sigma_1 = \{E \mapsto C, C \mapsto F\}\{F \mapsto E, E \mapsto D\} \\
r_2: \quad & h(A,B) \text{ :- } p(A,C), p(B,E), p(E,\mathbf{D}). \\
& \qquad \sigma_2 = \{E \mapsto D, D \mapsto F\}\{F \mapsto E, E \mapsto D\} \\
r_3: \quad & h(A,B) \text{ :- } p(A,C), p(B,D), p(D,E).
\end{aligned}
$$

Experiments





- (right) Tested on a single hard tasks from *trains* dataset.

Future work

- **Simple:**
  - Integrate with statisitical learning
  - **Solve ARC!**
- Papers can be found on ArXiv:
  - *Efficient Rule Induction by Ignoring Pointless Rules* (A. Cropper, D. M. Cerna, **AAAI-26**)
  - *Honey, I Shrunk the Hypothesis Space (Through Logical Preprocessing)* (A. Cropper, F. Gouveia, D. M. Cerna, **Journal of AI Research**)
  - *Symmetry Breaking for Inductive Logic Programming* (A. Cropper, D. M. Cerna, M. Järvisalo, AAAI-26)