# Lifting Symmetry Breaking Constraints with Inductive Logic Programming

**Alice Tarzariol, Martin Gebser, Konstantin Schekotihin**
Department of Artificial Intelligence and Cybersecurity, University of Klagenfurt, Austria
firstname.lastname@aau.at

ALPEN-ADRIA UNIVERSITÄT
KLAGENFURT | WIEN GRAZ

## Instance-specific Approaches

Instance-specific approaches automatically identify and remove symmetries of a problem instance through Symmetry Breaking Constraints (SBCs).

Limitations:

- The propositional constraints do not generalize.
- They are often hard to interpret and comprehend.
- The computation of symmetries burdens the solving phase.
- Many redundant constraints might by added to the input program, thus, degrading the solving performance.

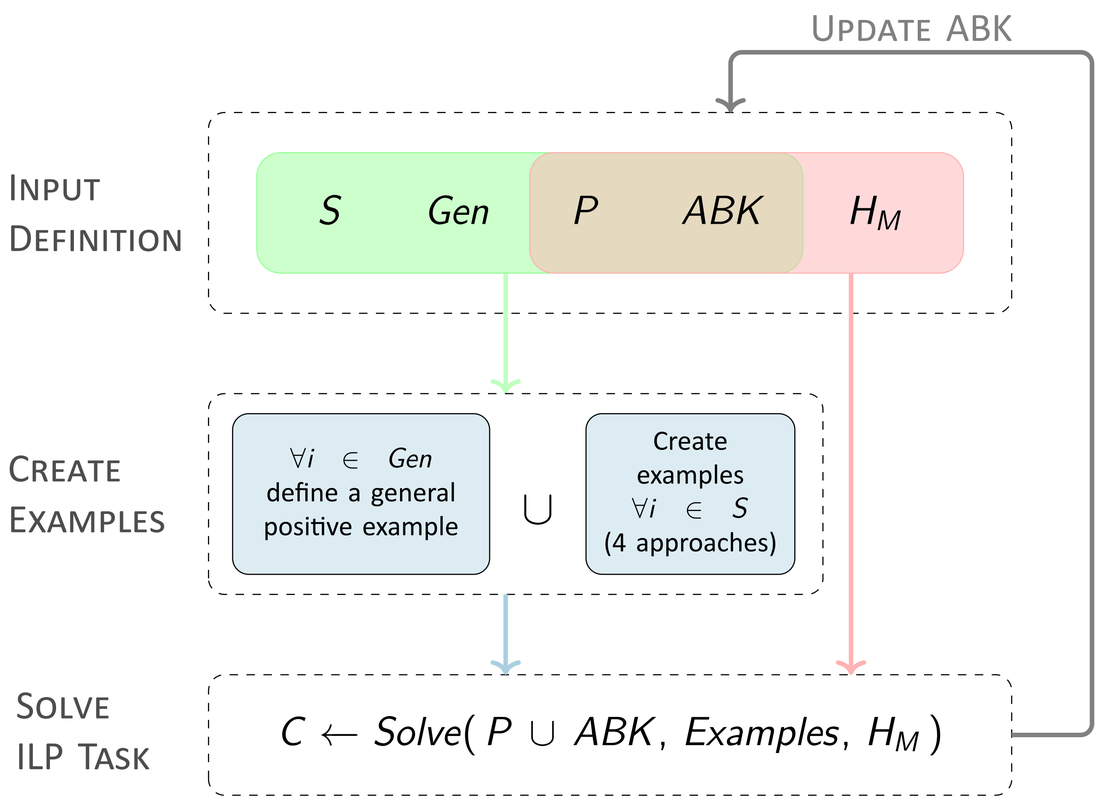
## Suggested Approach

To overcome the limitations of instance-specific methods, we suggest an approach to compute SBCs offline and lift them using Inductive Logic Programming (ILP) based on the Answer Set Programming (ASP) paradigm.

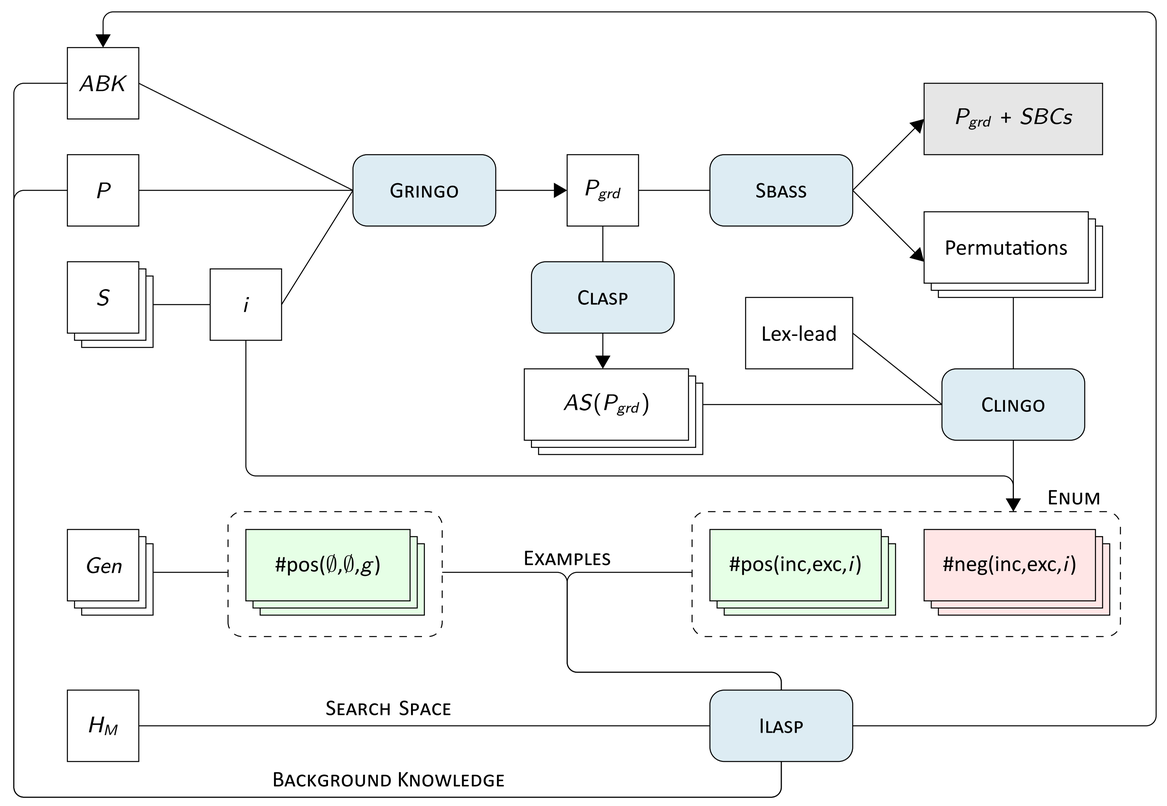
It consists of executing the following steps:

1) Define a set of small representative instances for a target ASP program.
2) Use SBASS (an instance-specific SBCs system) to identify their symmetries.
3) Use the symmetries to create examples for an ILP task.
4) Solve the ILP task with the system ILASP and learn first-order constraints.

## Framework




## Implementation




## Learning Approaches

Each instance $i \in S$ defines a set of positive and negative examples. We implemented four possibilities for defining them:

- **Enum** Enumerate all the answer sets computed from $i$ and use the lex-leader criterion with the permutations to label them as positive or negative examples.
- **Sat** Similar to *Enum*, but instead of a set of positive examples, it produces a single example with empty inclusions and exclusions.
- **Ord** An alternative atom ordering for the lex-leader schema, giving shorter first-order constraints for configuration problem encodings.
- **Full** Explore the orbit of symmetric interpretation to label the answer sets as positive or negative according to full symmetry breaking.

With all of them, we run ILASP incrementally by gradually extending the language bias.

## Solving Experiments

We tested our approach over three versions of the pigeon-hole problem and the house-configuration problem, setting a timeout of 900 seconds.
Here, we report the results for the house-configuration problem (left) and an extension of the pigeon-hole problem with color assignment (right).

| | ENUM | SAT | ORD | FULL | BASE | SBASS | CLASP$^\pi$ |
|---|---|---|---|---|---|---|---|
| p2-c6-t13 | 0.337 | 0.329 | 0.097 | **0.025** | 219.753 | 0.095 | 12.951 |
| p2-c80-t160 | **4.928** | 5.024 | 5.166 | 5.001 | 6.583 | TO | – |
| p3-c6-t13 | 0.342 | 0.424 | 0.093 | **0.031** | 254.065 | 0.242 | 73.041 |
| p3-c80-t160 | **13.682** | 14.110 | 15.174 | 14.315 | 20.724 | TO | – |
| p4-c6-t13 | 0.420 | 0.349 | 0.102 | **0.035** | 221.784 | 0.453 | 105.145 |
| p4-c80-t160 | 27.437 | **27.299** | 28.145 | 28.866 | 40.121 | TO | – |
| p5-c6-t13 | 0.418 | 0.397 | 0.101 | **0.042** | 236.961 | 0.890 | 405.461 |
| p5-c80-t160 | **48.057** | 49.645 | 48.598 | 49.382 | 68.167 | TO | – |
| p4-c7-t15 | 13.263 | 14.229 | 1.524 | **0.335** | TO | 0.729 | TO |
| p15-c15-t30 | 5.655 | **2.525** | 2.700 | 6.777 | 4.155 | TO | – |

| | ENUM | SAT | ORD | FULL | BASE | SBASS | CLASP$^\pi$ |
|---|---|---|---|---|---|---|---|
| c1-p12-h11 | 1.939 | **0.006** | 0.007 | 0.007 | 692.704 | 0.284 | 0.015 |
| c1-p52-h52 | 0.147 | **0.100** | 0.101 | **0.100** | 0.145 | 63.455 | 1.083 |
| c2-p12-h12 | 6.648 | 0.010 | 0.010 | **0.009** | TO | 0.092 | TO |
| c2-p52-h53 | 0.559 | **0.293** | 0.421 | 0.298 | TO | 93.870 | TO |
| c3-p12-h13 | 4.457 | 0.019 | 0.052 | **0.014** | TO | 0.228 | TO |
| c3-p52-h54 | 4.974 | 0.628 | 1.072 | **0.545** | 1.674 | 449.324 | TO |
| c4-p12-h14 | 4.195 | 0.075 | 3.026 | **0.022** | TO | 0.479 | TO |
| c4-p52-h55 | 3.131 | 3.849 | 2.033 | **0.959** | TO | 550.065 | TO |
| c5-p12-h15 | 5.673 | 0.251 | 138.861 | **0.035** | TO | 1.020 | TO |
| c5-p52-h56 | 18.336 | 11.704 | 581.164 | **1.561** | 5.930 | TO | – |

## Conclusions

In our experiments, the learned constraints:

- are general and easy to interpret, since they are first-order;
- show different characteristics w.r.t. the used example generation method;
- obtain better solving performance than the original program and the online application of instance-specific symmetry breaking.

## Future Works

- Tackle more advanced configuration problems.
- Address scalability issues.
- Evaluate and automate the generation of *Gen*, $S$, $H_M$, so that learned constraints evolve incrementally.