Conflict Generalisation in ASP
Learning Correct and Effective Non-Ground Constraints
(Extended Abstract⋆)

Richard Taupe1,2, Antonius Weinzierl3, and Gerhard Friedrich2

1 Siemens AG Österreich
   richard.taupe@siemens.com
2 Alpen-Adria-Universität Klagenfurt
   gerhard.friedrich@aau.at
3 TU Wien (Vienna University of Technology), Austria
   weinzierl@kr.tuwien.ac.at

Conflict-Driven Nogood Learning (CDNL) [6, 10, 14] is a major success factor for
high-performance state-of-the-art ASP systems. When a conflict occurs, new proposi-
tional nogoods are learned that prevent the same conflict from re-occurring, which
improves search performance. We present an extension of CDNL that learns non-ground
constraints. The idea is that whole parts of the search tree can be pruned when these
learned constraints are added to the original program. We aim to derive non-ground
constraints from the problem encoding that are valid for all possible inputs and which
can be employed to speed up solving new instances.

CDNL and Explanation-Based Learning (EBL) [2, 7, 8, 12] are our starting point.
EBL is a well-known logic-based machine learning technique which learns first-order
rules that are entailed by the background knowledge (in our case, the problem encod-
ing). We combine CDNL with EBL to learn non-ground nogoods while solving prior
problem instances. Since the number of generalised nogoods can be overwhelming,
choosing those that will actually pay off is particularly challenging. Our basic idea is to
generalise those non-ground conflicts that occur most often, i.e., we generalise proposi-
tional nogoods learned from frequently violated nogoods. The underlying assumption is
that nogoods learned from frequent conflicts will also be able to prevent many conflicts.

Previous work by Lühne et al. [4, 11] used different means to generalise learned
nogoods for future re-use: In their approach, propositional constraints are extracted
while solving, then generalized by minimization and abstraction, and finally validated
by means of proof techniques.

We use a realistic hardware configuration example (the House Reconfiguration Prob-
lem (HRP) [3, 13]) and a graph colouring problem for demonstration and experimen-
tation purposes. In simplified terms, the HRP is about assigning things to cabinets and
cabinets to rooms s.t. every room contains things belonging to only one person. This is
expressed by the following rule and constraint:

\[
\text{personTOroom}(P, R) \leftarrow \text{personTOthing}(P, T), \text{cabinetTOthing}(C, T), \text{roomTOcabinet}(R, C).
\]

⋆ This is an extended abstract of research already published in [15].
Employing our method, the following redundant constraint can be learned when the above constraint is violated:

\[
\text{cabinetTOthing}(C, T_1), \text{personTOthing}(P_1, T_1), \\
\text{cabinetTOthing}(C, T_2), \text{personTOthing}(P_2, T_2), P_1 < P_2.
\]

This makes the knowledge explicit that no two things belonging to different persons may be placed in the same cabinet.

Experimental results with ALPHA [9, 16], DLV 2.0 [1], and CLINGO [5] show that both lazy-grounding and ground-and-solve systems benefit from our approach. By adding learned constraints to the problem encodings, more instances can be solved and/or the problems can be solved faster. Learning itself has been implemented within ALPHA and requires low computational resources.

It seems natural to assume that many domains feature redundant constraints that may not be obvious to a human modeller. We therefore see our approach mainly as a tool to support the design of efficient answer-set programs.

Acknowledgments. This work has been conducted in the scope of the research project DynaCon (FFG-PNr.: 861263), which is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program “ICT of the Future” between 2017 and 2020, and in the scope of the research project Productive4.0, which is funded by EU-ECSEL under grant agreement no737459.

References


4 https://github.com/alpha-asp/Alpha


