Degrees of Laziness in Grounding

Effects of Lazy-Grounding Strategies on ASP Solving (Extended Abstract*)

Richard Taupe[1,2](✉), Antonius Weinzierl[3], and Gerhard Friedrich[2]

[1] Siemens AG Österreich
richard.taupe@siemens.com
[2] Alpen-Adria-Universität Klagenfurt
gerhard.friedrich@aau.at
[3] Insitute of Logic and Computation, Vienna University of Technology, Austria
weinzierl@kr.tuwien.ac.at

Abstract. The traditional ground-and-solve approach to Answer Set Programming (ASP) suffers from the grounding bottleneck. Lazy grounding is an alternative approach that interleaves grounding with solving and thus uses space more efficiently. This limited view on the search space poses unique challenges, however, and can have adverse effects on solving performance. We present a novel characterization of degrees of laziness in ASP grounding, i.e., of compromises between lazily grounding as little as possible and the traditional full grounding upfront. Our contributions are the introduction of a range of novel lazy grounding strategies, a formal account on their relationships and their correctness, and an investigation of their effects on solving performance. Experiments show that our approach performs significantly better than state-of-the-art lazy grounding in many cases.

1 Introduction

Answer Set Programming (ASP) is a declarative knowledge representation formalism whose successful application in science and industry is rooted in efficient solvers. Such solvers usually apply the *ground-and-solve* approach, i.e., they first instantiate a given non-ground program and then apply efficient solving techniques to find answer sets of the variable-free (i.e., ground) program.

This approach suffers from the *grounding bottleneck* since in many practical and industrial applications the ground program is too large to fit in memory. Such problem instances cannot be grounded by modern grounders such as GRINGO [3] or I-DLV [1] in acceptable time and/or space.

Lazy-grounding ASP [2, 4, 5, 7] successfully avoids the grounding bottleneck by interleaving grounding and solving, but suffers from substandard search performance. In order to improve solving performance, we explore various lazy-grounding strategies to find compromises between full upfront grounding and largely blind search heuristics.

* This is an extended abstract of research already published in [6].

2 Lazy-Grounding Strategies

In the lazy-grounding ASP system Alpha [7], a ground rule is currently only fed to the solver if its positive body is fully satisfied. This is a very restrictive grounding strategy in order to save space and avoid the grounding bottleneck. As experience shows, this *maximally strict* grounding strategy employed by Alpha results in non-optimal search performance, because state-of-the-art search procedures only operate on grounded parts of the problem. With maximally strict lazy-grounding these search procedures (e.g., branching heuristics) are left mostly blind when large parts of the given problem instance are not yet grounded.

We thus investigate more permissive lazy-grounding strategies that lie between the maximally strict one and the full upfront grounding (the *maximally permissive* grounding strategy). The more permissive a grounding strategy, the less restrictions it poses on ground rules returned by the grounder. Thus, ground rules are produced earlier and in higher quantity.

Our main contribution is the introduction and formal characterization of various classes of grounding strategies ("degrees of laziness"), like k-unassigned grounding strategies (which basically produce ground rules in which at most k positive body atoms are still unassigned) and accumulator-based ones (which use ground atoms encountered in other search paths to trigger additional grounding), that allow compromises between lazily grounding as little as possible and the traditional grounding upfront. Experimental results show a clear improvement over existing lazy-grounding strategies and that permissive grounding of constraints usually improves solving performance, while the performance improvements from other grounding strategies depend on the problem to be solved.

Acknowledgments. This work was 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 was supported by EU ECSEL Joint Undertaking under grant agreement no. 737459 (project Productive4.0).

References

1. Calimeri, F., Fuscà, D., Perri, S., Zangari, J.: I-DLV: the new intelligent grounder of DLV. Intelligenza Artificiale **11**(1), 5–20 (2017)
2. Dao-Tran, M., Eiter, T., Fink, M., Weidinger, G., Weinzierl, A.: OMiGA: An open minded grounding on-the-fly answer set solver. In: JELIA. LNCS, vol. 7519, pp. 480–483. Springer (2012)
3. Gebser, M., Kaminski, R., König, A., Schaub, T.: Advances in *gringo* series 3. In: LPNMR. LNCS, vol. 6645, pp. 345–351. Springer (2011)
4. Lefèvre, C., Béatrix, C., Stéphan, I., Garcia, L.: ASPeRiX, a first-order forward chaining approach for answer set computing. TPLP **17**(3), 266–310 (2017)
5. Palù, A.D., Dovier, A., Pontelli, E., Rossi, G.: GASP: answer set programming with lazy grounding. Fundam. Inform. **96**(3), 297–322 (2009)
6. Taupe, R., Weinzierl, A., Friedrich, G.: Degrees of laziness in grounding. In: LPNMR. LNCS, vol. 11481, pp. 298–311. Springer (2019)
7. Weinzierl, A.: Blending lazy-grounding and CDNL search for answer-set solving. In: LPNMR. LNCS, vol. 10377, pp. 191–204. Springer (2017)