Body-Decoupled Grounding via Solving: A Novel Approach on the ASP Bottleneck (Extended Abstract)[*]

Viktor Besin[1] , **Markus Hecher**[2] , **Stefan Woltran**[1]
[1] TU Wien, Austria
[2] Massachusetts Institute of Technology, United States
{vbesin, woltran}@dbai.tuwien.ac.at, hecher@mit.edu

Abstract

Answer-Set Programming (ASP) has seen tremendous progress over the last two decades and is nowadays successfully applied in many real-world domains. However, for certain problems, the well-known ASP grounding bottleneck still causes severe problems. This becomes virulent when grounding of rules, where the variables have to be replaced by constants, leads to a ground program that is too huge to be processed by the ASP solver. In this work, we tackle this problem by a novel method that decouples non-ground atoms in rules in order to delegate the evaluation of rule bodies to the solving process. Our procedure translates a non-ground normal program into a ground disjunctive program that is exponential only in the maximum predicate arity, and thus polynomial if this arity is bounded by a constant. We demonstrate the feasibility of this new method experimentally by comparing it to standard ASP technology in terms of grounding size, grounding time and runtime.

1 Body-Decoupled Grounding

Motivated by the ASP *grounding bottleneck* (Cuteri et al. 2020; Tsamoura, Gutiérrez-Basulto, and Kimmig 2020), the problem of traditional grounding systems resulting in exponentially large programs when instantiating non-ground rules (even for programs with bounded predicate arities), we briefly tease the concept of *body-decoupled* grounding. The idea of this approach is to reduce the grounding size and grounding time by decoupling dependencies between different predicates of rule bodies. The potential of this is motivated by the following example.

Example 1. *Assume the following non-ground program* Π *that decides in (1) for each edge (e) of a given graph, whether to pick it (p) or not ($\bar{p}$). Then, in (2) it is ensured that the choice of edges does not form triangles.*

$$p(A,B) \vee \bar{p}(A,B) \leftarrow e(A,B) \qquad (1)$$

$$\leftarrow p(X,Y), p(Y,Z), p(X,Z), X \neq Y, Y \neq Z, X \neq Z. \qquad (2)$$

The typical grounding effort of (2) is in $\mathcal{O}(|\mathrm{dom}(\Pi)|^3)$. Our approach grounds body predicates of (2) individually, yielding groundings that are linear in the size of the ground atoms. In our example, this corresponds to $\mathcal{O}(|\mathrm{dom}(\Pi)|^2)$ due to arity 2.

[*]This is an extended abstract of a paper (Besin, Hecher, and Woltran 2022) that appeared at IJCAI'22.

	Ground	Non-Ground (bounded arity)
Tight/Normal Programs	NP-c	Σ_2^P-c
Disjunctive Programs	Σ_2^P-c $\leftarrow$	Σ_3^P-c

Table 1: Known complexity results for some of the program types.

Based on earlier complexity results for ground and non-ground logic programs (Bidoít and Froidevaux 1991; Marek and Truszczyński 1991; Eiter et al. 2007), we introduce a reduction-based translation from non-ground, tight (and normal) programs to ground, disjunctive programs (see arrow in Tab. 1), resulting in an alternative grounding procedure. Our encodings translate a non-ground rule by (i) guessing whether the head atom is part of the answer set, (ii) ensuring satisfiabilty of the rule and (iii) preventing unfoundedness of the guessed head atom. To lift this idea to normal programs, one can rely on encoding (iv) orderings. Since every step of the procedure instantiates at most one body predicate at a time, we intuitively deploy body-decoupling, which keeps the grounding size polynomial when assuming bounded predicate arity. Notably, our results imply that body-decoupled grounding blends-in well with existing approaches, enabling us to interleave different techniques.

2 Experimental Results

We implemented a software tool, called `newground`[1], realizing body-decoupled grounding via search as described above. The system `newground` is written in Python3 and uses, among others, the API of `clingo` 5.5 and its ability to efficiently parse logic programs via syntax trees. In our implementation, we opted for partial reducability, allowing users to select program parts that shall be reduced and those being (traditionally) grounded, thereby internally relying on `gringo`.

In order to evaluate `newground`, we design a series of benchmarks. Clearly, we cannot beat highly optimized grounders in all imaginable scenarios. Instead, we discuss potential use cases, where body-decoupled grounding is preferrable, since this approach can be incorporated into every grounder. We consider these (directed) graph *scenarios*: (S1) 3-coloring, (S2) reachable paths, (S3) cliques,

[1]The system (incl. supplemental material) is available at https://github.com/viktorbesin/newground.

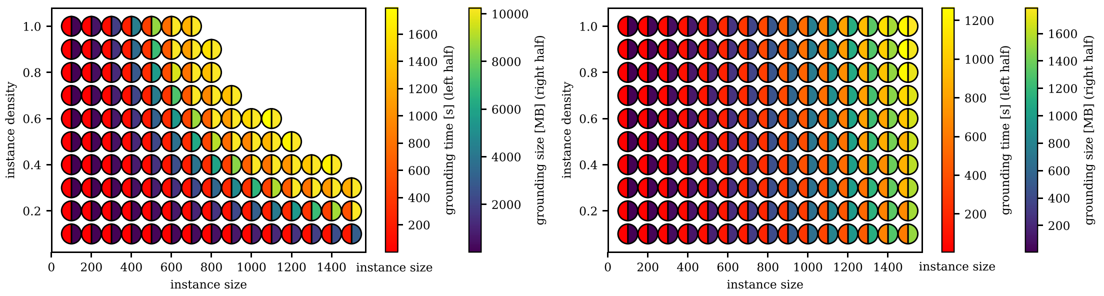

Figure 1: Grounding profile of S1 for `gringo` (left) and `newground` (right). The x-axis refers to the instance size; the y-axis indicates density. Circles mark instances grounded $< 1800s$; the left (right) half depicts grounding time (size), respectively. Mind the different color scales (e.g., 10000MB vs. 1600MB).

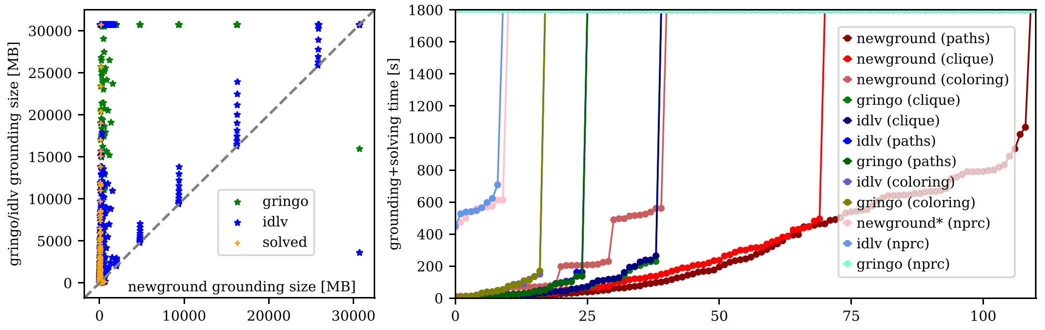

Figure 2: (Left): Scatter plot of grounding size over Scenarios S1–S4 of `newground` (x-axis) compared to both `gringo` (blue) and `idlv` (green) on the y-axis. Those instances that could be solved are highlighted in orange. (Right): Cactus plot of overall (grounding *and solving*) time over Scenarios S1-S4.

(S4) non-partition-removal colorings (Weinzierl, Taupe, and Friedrich 2020) and (S5) stable marriages (ASP comp. 2014), and compared to `gringo` and `idlv` while measuring grounding size, grounding time as well as overall time.

From our experiments we crafted grounding profiles for each tool. Figure 1 depicts the grounding profile of S1 for `gringo` (left) and `newground` (right), showing grounding times and sizes, depending on the instance size (x-axis) and density (y-axis). Interestingly, for `newground` grounding times and sizes for a fixed instance size (column) of Figure 1 (right) are quite similar, which is in contrast to Figure 1 (left), suggesting that compared to `gringo`, `newground` is not that sensitive to instance density.

In terms of pure grounding time, our experiments show that `newground` in fact outperforms in four of five scenarios. Interestingly, while doing so, `newground` also massively reduces the grounding size (cf., Figure 2 (left), almost all dots above diagonal), while keeping instances solvable where `gringo` and `idlv` output groundings beyond 30GB. For the overall (solving) performance we refer to Figure 2 (right). While `newground` performs best, we still see a clear difference between solving and grounding performance, which reveals that only a small amount of those grounded instances can then actually be solved by `clingo` within the remaining time. More plots and evaluation can be found in (Besin, Hecher, and Woltran 2022).

3 Conclusion

This work introduces a grounding-approach based on a reduction suggesting the body-decoupling of grounding-intense ASP rules. The reduction translates tight (normal) non-ground rules into disjunctive ground rules, thereby being exponential only in the maximum predicate arity. While our evaluation shows that body-decoupled grounding applied on crucial (tight) program parts reduces grounding size compared to state-of-the-art exact grounders, we are currently working on evaluating and tuning of an implementation for normal programs (Unalan 2022).

References

Besin, V.; Hecher, M.; and Woltran, S. 2022. Body-Decoupled Grounding via Solving: A Novel Approach on the ASP Bottleneck. In *IJCAI'22*, 2546–2552. ijcai.org.

Bidoít, N., and Froidevaux, C. 1991. Negation by default and unstratifiable logic programs. *Theoretical Computer Science* 78(1):85–112.

Cuteri, B.; Dodaro, C.; Ricca, F.; and Schüller, P. 2020. Overcoming the grounding bottleneck due to constraints in ASP solving: Constraints become propagators. In *IJCAI*, 1688–1694. ijcai.org.

Eiter, T.; Faber, W.; Fink, M.; and Woltran, S. 2007. Complexity results for answer set programming with bounded predicate arities and implications. *Annals of Mathematics and Artificial Intelligence* 51(2-4):123–165.

Marek, W., and Truszczyński, M. 1991. Autoepistemic logic. *Journal of the ACM* 38(3):588–619.

Tsamoura, E.; Gutiérrez-Basulto, V.; and Kimmig, A. 2020. Beyond the grounding bottleneck: Datalog techniques for inference in probabilistic logic programs. In *AAAI'20*, 10284–10291. AAAI.

Unalan, K. 2022. Body-Decoupled Grounding in Normal Answer Set Programs. Bachelor's Thesis, TU Wien, Austria.

Weinzierl, A.; Taupe, R.; and Friedrich, G. 2020. Advancing lazy-grounding ASP solving techniques - restarts, phase saving, heuristics, and more. *Theory Pract. Log. Program.* 20(5):609–624.