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

Viktor Besin1, Markus Hecher2, Stefan Woltran1
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 (¬p). Then, in (2) it is ensured that the choice of edges does not form triangles.

\[ p(A, B) \lor \bar{p}(A, B) \leftarrow e(A, B) \]  
\[ p(X, Y), p(Y, Z), p(X, Z), X \neq Y, Y \neq Z, X \neq Z. \]

The typical grounding effort of (2) is in \( O(|\text{dom}(\Pi)|^3) \). 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 \( O(|\text{dom}(\Pi)|^5) \) due to arity 2.

<table>
<thead>
<tr>
<th></th>
<th>Ground</th>
<th>Non-Ground (bounded arity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tight/Normal Programs</td>
<td>NP-c</td>
<td>( \Sigma_2^{NP} )-c</td>
</tr>
<tr>
<td>Disjunctive Programs</td>
<td>( \Sigma_2^{NP} )-c</td>
<td>( \Sigma_3^{NP} )-c</td>
</tr>
</tbody>
</table>

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

Based on earlier complexity results for ground and non-ground logic programs (Bidoit 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 satisfiability 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-decoupiling, 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 newground1, 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,

∗This is an extended abstract of a paper (Besin, Hecher, and Woltran 2022) that appeared at IJCAI’22.

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


