Bypassing the ASP Bottleneck: Hybrid Grounding by Splitting and Rewriting

Alexander Beiser[1], Markus Hecher[2], Kaan Unalan[1], and Stefan Woltran[1]

[1] TU Wien, Vienna, Austria `<firstname>.<lastname>@tuwien.ac.at`
[2] Massachusetts Institute of Technology, MA, United States `hecher@mit.edu`

1 Introduction

Efficient Answer Set Programming (ASP) [13, 14, 17, 12] systems [4, 9] led to the success of ASP. Their *grounding* [16], instantiation of variables by their domain, strategy leads to the *grounding bottleneck* [11, 18]. It prohibits the grounding of large practical instances. This is no surprise, as it is well known that the evaluation of a non-ground normal ASP program is *NEXPTIME*-complete [6]. But, by bounding the maximum predicate arity, this can be reduced to Σ_2^{P} completeness [8]. Many approaches to circumvent this problem have been proposed, such as program compilation [5, 7], lazy-grounding [11, 19], ASP-modulo theory [9, 15], or body-decoupled grounding (BDG) [2]. However, no single approach works well in all tasks.

In this extended abstract, we present *Hybrid Grounding*, a short version of our paper presented at IJCAI24 [1]. Hybrid Grounding combines the best of the worlds of traditional grounding and BDG, by enabling the free (manual) *splitting* of an ASP program into a part grounded by traditional, and a part where grounding is accomplished by *rewriting* rules along the ideas of BDG.

2 Background

Traditional grounding approaches use bottom-up grounding combined with semi-naive grounding [10]. In contrast, BDG leverages complexity theoretical results of ASP. BDG is the reduction for the answer set existence problems of *(tight) normal* programs with bounded predicate arities, to disjunctive ground programs, which are both Σ_2^{P} complete [8][3]. This reduction procedure consists of the following steps: (i) Guess the answer set candidates, (ii) ensure satisfiability, and (iii) prevent unfoundedness.

BDG obtains its practicality by decoupling dependencies between different predicates in rule bodies. Therefore, BDG provides an efficient alternative to cope with large and dense rule bodies, as body elements are grounded independently. Given a (tight normal) program Π, a maximum arity of a, and a domain (ground terms) of $dom(\Pi)$, the grounding size is in $\mathcal{O}\left(|\Pi| \cdot dom(\Pi)^{2 \cdot a}\right)$ (Theorem 2 of [2]).

However, BDG faces several shortcomings. One is its limited interoperability, when programs Π_1 and Π_2 shall be grounded via BDG and traditional grounding, respectively. So far, BDG is only applicable when the *head predicates of both programs are disjoint*. Further, it only supports a very limited subset of the ASP syntax that is supported by e.g., *Clingo*.

[3] Note that this reduction can be extended to non-tight normal programs. Also a reduction exists for disjunctive ASP programs to epistemic programs (Σ_3^{P} complete) [3].

3 Hybrid Grounding

We address the shortcomings of BDG by proposing *Hybrid Grounding*, which enables the free (manual) splitting between parts of a program Π into Π_1 (grounded along the ideas of BDG), and Π_2 (grounded by traditional approaches).

At a high level, the splitting is enabled by the usage of auxiliary atoms[4] in Π_1. Further, suitable *communication* between Π_1 and Π_2 is implemented to guarantee semantic equivalence. Additionally, correct treatment of cycles in the positive dependency graph spanning Π_1 and Π_2 (thereby being *shared cycles*) is ensured. The following example (taken from [1]) shall illustrate the deficiencies of BDG and sketch the functioning and potential of Hybrid Grounding.

Example 1 Consider the non-ground program $\Pi := \{r_a; r_b\}$. Let G be a graph encoded by edge predicates (e), given as an instance $\mathcal{F} := \{b(1);\text{c}(2)\} \cup G$. Now we define the rules r_a and r_b: $r_a = a(X) \leftarrow b(X)$, $r_b = a(X) \leftarrow c(X), e(A,B), e(A,C), e(B,C)$. Rule r_b is dense in the sense that it has 3 interacting variables (*A,B,C*). To avoid groundings which are cubic in the domain size, one would like to use BDG to ground r_b. However, as r_a is not dense, one would like to use standard approaches to ground r_a.

The problem is, as mentioned, that BDG is so far not able to do this. This comes, as (i) rule r_a justifies $a(1)$. But (ii) BDG checks whether r_b justifies $a(1)$ as well. As r_b does not justify $a(1)$, but $a(1)$ must be part of the answer set due to r_a, the BDG part would wrongly conclude that $a(1)$ is unfounded. Therefore, there would erroneously be no resulting answer set.

Hybrid Grounding circumvents this problem by introducing auxiliary predicates, thereby disentangling the parts grounded by standard and by BDG techniques. In our example, this boils down to the introduction of a' for rule r_b. So, (i) r_a derives $a(1)$ as before. But conceptually, Hybrid Grounding does, in contrast to above, not check whether $a(1)$ is founded by r_b. It checks whether $a'(1)$ is founded by r_b. As (ii) $a'(1)$ is not founded by r_b (and $a'(1)$ is not required to hold), $a'(1)$ is not derived in the first place. Therefore, the result is then correct.

We realized Hybrid Grounding in our prototype `NaGG`[5]. Additionally, we extended the input language of Hybrid Grounding to aggregates, and demonstrated its practicality by conducting experiments.

4 Conclusion

We present Hybrid Grounding, the combination of standard grounding with rewritings based on the ideas of body-decoupled grounding, which we presented at IJCAI24 [1]. Furthermore, it extends the input language of BDG, and finally, we showed its practical significance by conducting a series of experiments. In future work, we want to investigate the potential of automatic splitting and improve the performance of the non-tight reduction.

[4] The details of the hybrid grounding reduction are in the appendix, for both tight and non-tight versions. Auxiliary atomes are indicated by a p'.

[5] Supplementary material including source code, benchmark instances and experimental results, are available online: `https://github.com/alexl4123/newground/releases/tag/v2.0.0`.

Acknowledgements

Research was funded by Vienna Science and Technology Fund (WWTF) grant ICT19-065 and Austrian Science Fund (FWF) grants J 4656 and 10.557766/COE12.

References

1. Beiser, A., Hecher, M., Unalan, K., Woltran, S.: Bypassing the ASP Bottleneck: Hybrid Grounding by Splitting and Rewriting. IJCAI24 pp. 3250–3258 (2024). https://doi.org/10.24963/ijcai.2024/360, `https://www.ijcai.org/proceedings/2024/360`, iSBN: 978-1-956792-04-1
2. Besin, V., Hecher, M., Woltran, S.: Body-Decoupled Grounding via Solving: A Novel Approach on the ASP Bottleneck. IJCAI22 pp. 2546–2552 (2022). https://doi.org/10.24963/ijcai.2022/353, `https://www.ijcai.org/proceedings/2022/353`, iSBN: 978-1-956792-00-3
3. Besin, V., Hecher, M., Woltran, S.: On the Structural Complexity of Grounding – Tackling the ASP Grounding Bottleneck via Epistemic Programs and Treewidth. ECAI23 **372**, 247–254 (2023). https://doi.org/10.3233/FAIA230277, `https://ebooks.iospress.nl/doi/10.3233/FAIA230277`, iSBN: 978-1-64368-436-9 978-1-64368-437-6
4. Calimeri, F., Fuscà, D., Perri, S., Zangari, J.: I-DLV: The new intelligent grounder of DLV. IA **11**(1), 5–20 (2017). https://doi.org/10.3233/IA-170104, `https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/IA-170104`
5. Cuteri, B., Dodaro, C., Ricca, F., Schüller, P.: Overcoming the Grounding Bottleneck Due to Constraints in ASP Solving: Constraints Become Propagators. IJCAI20 pp. 1688–1694 (2020). https://doi.org/10.24963/ijcai.2020/234, `https://www.ijcai.org/proceedings/2020/234`, iSBN: 978-0-9992411-6-5
6. Dantsin, E., Eiter, T., Gottlob, G.: Complexity and expressive power of logic programming. ACM Comput. Surv. **33**(3), 374–425 (2001). https://doi.org/10.1145/502807.502810
7. Dodaro, C., Mazzotta, G., Ricca, F.: Compilation of Tight ASP Programs. ECAI23 **372**, 557–564 (2023). https://doi.org/10.3233/FAIA230316, `https://ebooks.iospress.nl/doi/10.3233/FAIA230316`, iSBN: 978-1-64368-436-9 978-1-64368-437-6
8. Eiter, T., Faber, W., Fink, M., Woltran, S.: Complexity results for answer set programming with bounded predicate arities and implications. Ann. Math. Artif. Intell. **51**(2-4), 123–165 (2007). https://doi.org/10.1007/s10472-008-9086-5, `http://link.springer.com/10.1007/s10472-008-9086-5`
9. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory Solving Made Easy with Clingo 5. ICLP16 **52**, 1–15 (2016). https://doi.org/10.4230/OASICS.ICLP.2016.2, `https://drops.dagstuhl.de/entities/document/10.4230/OASIcs.ICLP.2016.2`, artwork Size: 15 pages, 632693 bytes ISBN: 9783959770071 Medium: application/pdf Publisher: Schloss Dagstuhl – Leibniz-Zentrum für Informatik
10. Gebser, M., Kaminski, R., Schaub, T.: Grounding Recursive Aggregates: Preliminary Report. GTTV15 (2012), `http://arxiv.org/abs/1603.03884`
11. Gebser, M., Leone, N., Maratea, M., Perri, S., Ricca, F., Schaub, T.: Evaluation Techniques and Systems for Answer Set Programming: a Survey. IJCAI18 pp. 5450–5456 (2018). https://doi.org/10.24963/ijcai.2018/769, `https://www.ijcai.org/proceedings/2018/769`, iSBN: 978-0-9992411-2-7

12. Gelfond, M., Leone, N.: Logic programming and knowledge representation—The A-Prolog perspective. Artif. Intell. **138**(1-2), 3–38 (Jun 2002). https://doi.org/10.1016/S0004-3702(02)00207-2, https://linkinghub.elsevier.com/retrieve/pii/S0004370202002072
13. Gelfond, M., Lifschitz, V.: The Stable Model Semantics for Logic Programming. JICSLP88 pp. 1070–1080 (1988), https://www.cs.utexas.edu/~ai-lab/?gel88, editors: Kowalski, Robert and Bowen and Kenneth
14. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Gener Comput **9**(3-4), 365–385 (1991). https://doi.org/10.1007/BF03037169, http://link.springer.com/10.1007/BF03037169
15. Kaminski, R., Romero, J., Schaub, T., Wanko, P.: How to Build Your Own ASP-based System?! Theory Pract. Log. Program. **23**(1), 299–361 (2023). https://doi.org/10.1017/S1471068421000508, https://www.cambridge.org/core/product/identifier/S1471068421000508/type/journal_article
16. Kaminski, R., Schaub, T.: On the Foundations of Grounding in Answer Set Programming. TPLP23 **23**(6), 1138–1197 (2023). https://doi.org/10.1017/S1471068422000308, https://www.cambridge.org/core/product/identifier/S1471068422000308/type/journal_article
17. Pearce, D.: A new logical characterisation of stable models and answer sets. NMELP96 **1216**, 57–70 (1996). https://doi.org/10.1007/BFb0023801, iSBN : 978-3-540-68702-3
18. Tsamoura, E., Gutierrez-Basulto, V., Kimmig, A.: Beyond the Grounding Bottleneck: Datalog Techniques for Inference in Probabilistic Logic Programs. AAAI20 **34**(06), 10284–10291 (2020). https://doi.org/10.1609/aaai.v34i06.6591, https://ojs.aaai.org/index.php/AAAI/article/view/6591
19. Weinzierl, A.: Blending Lazy-Grounding and CDNL Search for Answer-Set Solving. In: Balduccini, M., Janhunen, T. (eds.) Logic Programming and Nonmonotonic Reasoning, vol. 10377, pp. 191–204. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-61660-5_17, http://link.springer.com/10.1007/978-3-319-61660-5_17, series Title: Lecture Notes in Computer Science

Appendix

In the Appendix, we provide the reader with selected details of our paper [1]. First we discuss the details of the hybrid grounding procedures for tight (Figure 1), and then for non-tight programs (Figure 2). In the IJCAI paper [1], the equivalent Figures can be found as Figure 1 (main paper), and Figure 7 (supplementary material).

(Tight) Hybrid Grounding Procedure

First, we discuss a simplified variant of hybrid grounding, where we only permit cyclic dependencies within Π_2, but not between Π_1 and Π_2. Crucially, we ensure that atoms are correctly derived, despite shared predicates between Π_1 and Π_2, thereby bijectively preserving all answer sets.

Figure 1 depicts hybrid grounding procedure $\mathcal{H}$. First, by Rules (1) we guess whether an atom can be derived by means of Π_1. Further, Rules (2) ensure that atoms over auxiliary predicates p' (i.e., those derived by means of Π_1) are copied to atoms over p. This is crucial, as these rules ensure the link between Π_1 and Π_2, but also contribute to rule satisfiability of Π_1, as discussed below. Then, Rules (3) ground Π_2 by classical means (i.e., rule instantiation). Note that in this step, practical grounders might simplify or remove rules of Π_2. However, this can be avoided by grounding Π_2 together with (non-ground versions of) Rules (1) and (2).

Then, Rules (4)–(10) ensure rule satisfiability of Π_1. This is achieved by guessing every potential instantiation in Rules (4), applying the saturation technique in Rules (9), and deriving rule satisfiability via avoiding rule body combinations due to Rules (6)–(8). Observe that the latter rules only use a *single (body) predicate* of Π_1. After that, rule satisfiability can be derived for all rules of Π_1 by Rules (5), which is mandatory due to (10).

Finally, Rules (11)–(14) of Figure 1 ensure that, in addition to the satisfiability of rules in Π_1, every atom that is guessed via Rules (1) is indeed founded. To this end, it is sufficient to find a suitable rule instantiation via Rules (11) for such an atom, and derive unfoundedness for a rule, again, by decoupling body atoms due to Rules (12) and (13), such that *not* every rule yields unfoundedness for that atom, see Rules (14). Note that the latter rules are relaxed such that these rules are only generated for atoms that are derived due to Π_1, i.e., if the atom over the auxiliary predicate p' holds.

Despite Rules (1), we bijectively preserve all answer sets.

Glue $\mathit{\Pi}_2$ to $\mathit{\Pi}_1$ and Ground $\mathit{\Pi}_2$		
$h'(\mathbf{D}) \vee \overline{h'(\mathbf{D})} \leftarrow$	for every $h(\mathbf{X}) \in \mathrm{hpred}(\mathit{\Pi}_1)$, $\mathbf{D} \in \mathrm{dom}(\mathbf{X})$	(1)
$h(\mathbf{D}) \leftarrow h'(\mathbf{D})$	for every $h(\mathbf{X}) \in \mathrm{hpred}(\mathit{\Pi}_1)$, $\mathbf{D} \in \mathrm{dom}(\mathbf{X})$	(2)
r	for every $r \in \mathcal{G}(\mathit{\Pi}_2)$	(3)
Satisfiability of $\mathit{\Pi}_1$		
$\bigvee_{d \in \mathrm{dom}(\mathbf{x})} \mathrm{sat}_x(d) \leftarrow$	for every $r \in \mathit{\Pi}_1, x \in \mathrm{var}(r)$, where $\mathit{\Pi}_1 = \{r_1, \ldots, r_n\}$	(4)
$\mathrm{sat} \leftarrow \mathrm{sat}_{r_1}, \ldots, \mathrm{sat}_{r_n}$	for every $r \in \mathit{\Pi}_1, x \in \mathrm{var}(r)$, where $\mathit{\Pi}_1 = \{r_1, \ldots, r_n\}$	(5)
$\mathrm{sat}_r \leftarrow \mathrm{sat}_{x_1}(\mathbf{D}_{\langle x_1 \rangle}), \ldots, \mathrm{sat}_{x_\ell}(\mathbf{D}_{\langle x_\ell \rangle}), \neg p(\mathbf{D})$	for every $r \in \mathit{\Pi}_1, p(\mathbf{X}) \in B_r^+$, $\mathbf{D} \in \mathrm{dom}(\mathbf{X}), \mathbf{X} = \langle x_1, \ldots, x_\ell \rangle$	(6)
$\mathrm{sat}_r \leftarrow \mathrm{sat}_{x_1}(\mathbf{D}_{\langle x_1 \rangle}), \ldots, \mathrm{sat}_{x_\ell}(\mathbf{D}_{\langle x_\ell \rangle}), p(\mathbf{D})$	for every $r \in \mathit{\Pi}_1, p(\mathbf{X}) \in B_r^-$, $\mathbf{D} \in \mathrm{dom}(\mathbf{X}), \mathbf{X}=\langle x_1, \ldots, x_\ell \rangle$	(7)
$\mathrm{sat}_r \leftarrow \mathrm{sat}_{x_1}(\mathbf{D}_{\langle x_1 \rangle}), \ldots, \mathrm{sat}_{x_\ell}(\mathbf{D}_{\langle x_\ell \rangle}), h'(\mathbf{D})$	for every $r \in \mathit{\Pi}_1, h(\mathbf{X}) \in H_r$, $\mathbf{D} \in \mathrm{dom}(\mathbf{X}), \mathbf{X}=\langle x_1, \ldots, x_\ell \rangle$	(8)
$\mathrm{sat}_x(d) \leftarrow \mathrm{sat}$	for every $r \in \mathit{\Pi}_1, x \in \mathrm{var}(r)$, $d \in \mathrm{dom}(\mathbf{x})$	(9)
$\leftarrow \neg\mathrm{sat}$		(10)
Prevent Unfoundedness of Atoms in $\mathit{\Pi}_1$		
$\bigvee_{d \in \mathrm{dom}(y)} \mathrm{uf}_y(\langle \mathbf{D}, d \rangle) \leftarrow h'(\mathbf{D})$	for every $r \in \mathit{\Pi}_1, h(\mathbf{X}) \in H_r$, $\mathbf{D} \in \mathrm{dom}(\mathbf{X}), y \in \mathrm{var}(r), y \notin \mathbf{X}$	(11)
$\mathrm{uf}_r(\mathbf{D}_{\mathbf{X}}) \leftarrow \mathrm{uf}_{y_1}(\mathbf{D}_{\langle \mathbf{X}, y_1 \rangle}), \ldots, \mathrm{uf}_{y_\ell}(\mathbf{D}_{\langle \mathbf{X}, y_\ell \rangle}), \neg p(\mathbf{D}_{\mathbf{Y}})$	for every $r \in \mathit{\Pi}_1, h(\mathbf{X}) \in H_r$, $p(\mathbf{Y}) \in B_r^+, \mathbf{D} \in \mathrm{dom}(\langle \mathbf{X}, \mathbf{Y} \rangle)$, $\mathbf{Y}=\langle y_1, \ldots, y_\ell \rangle$	(12)
$\mathrm{uf}_r(\mathbf{D}_{\mathbf{X}}) \leftarrow \mathrm{uf}_{y_1}(\mathbf{D}_{\langle \mathbf{X}, y_1 \rangle}), \ldots, \mathrm{uf}_{y_\ell}(\mathbf{D}_{\langle \mathbf{X}, y_\ell \rangle}), p(\mathbf{D}_{\mathbf{Y}})$	for every $r \in \mathit{\Pi}_1, h(\mathbf{X}) \in H_r$, $p(\mathbf{Y}) \in B_r^- \cup (H_r \setminus \{h(\mathbf{X})\})$, $\mathbf{D} \in \mathrm{dom}(\langle \mathbf{X}, \mathbf{Y} \rangle), \mathbf{Y}=\langle y_1, \ldots, y_\ell \rangle$	(13)
$\leftarrow \mathrm{uf}_{r_1}(\mathbf{D}), \ldots, \mathrm{uf}_{r_m}(\mathbf{D}), h'(\mathbf{D})$	for every $h(\mathbf{X}) \in \mathrm{hpred}(\mathit{\Pi}_1)$, $\mathbf{D} \in \mathrm{dom}(\mathbf{X})$, $\{r_1, \ldots, r_m\}=\{r \in \mathit{\Pi}_1 \mid h(\mathbf{X}) \in H_r\}$	(14)

Fig. 1: Hybrid grounding procedure $\mathcal{H}(\mathit{\Pi}_2, \mathit{\Pi}_1)$, which creates a disjunctive *ground* program from a given non-ground (HCF) program $\mathit{\Pi}_1 \cup \mathit{\Pi}_2$ such that $\mathit{\Pi}_1$ is tight. We thereby interleave classical grounding on $\mathit{\Pi}_2$ with body-decoupled grounding on $\mathit{\Pi}_1$.

(Non-tight) Hybrid Grounding Procedure

Figure 2 depicts the extended hybrid grounding procedure $\mathcal{H}_{lv}$, which is similar to $\mathcal{H}$ of Figure 1, but adds level mappings to deal with shared cycles between Π_2 and Π_1. More precisely, Rules (15), (16) and (17) replace Rules (3), where non-ground rules contained in Π_2, i.e., those that shall be grounded via traditional grounding, are adapted to consider the level mapping. Further, by adding Rules (18), (19), (20) and (21), we ensure total level mappings, transitivity, and foundedness, respectively. Additionally, Rules (22), (23), (25), and (26) (and thereby replacing Rules (1), (2) and (14)) we ensure *local foundedness* for each rule in Π_1 s.t. it is known which h_r founds which h, which is necessary for level mappings. Lastly, we replace Rule (8) with Rule (24), so a rule is satisfied when h is in the answer set. This is important as it ensures that a rule is not "forced to found" some h_r, when it cannot do that due to ordering constraints.

Replacing Rules (3) for (Shared) SCCs by:

$$\mathcal{G}(\{r'\}) \qquad \text{for every } r' \in \Pi_2, \text{ with } \forall S \in \text{scc}(\Pi) : (|E(S_{\Pi_1})| = 0 \text{ or } |E(S_{\{r'\}})| = 0) \tag{15}$$

$$a \leftarrow a_{l+1}, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n, (a_{l+1} \prec a), \ldots, (a_m \prec a) \qquad \text{for every } r' \in \Pi_2, \text{ with } \exists S \in \text{scc}(\Pi) : (|E(S_{\Pi_1})| \geq 1 \text{ and } |E(S_{\{r'\}})| \geq 1), r \in \mathcal{G}(\{r'\}), H_r = \{a\}, B_r^+ = \{a_{l+1}, \ldots, a_m\}, a \cup B_r^- = \{a, a_{m+1}, \ldots, a_n\} \tag{16}$$

$$\leftarrow a_{l+1}, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n, \neg a \qquad \text{for every } r' \in \Pi_2, \text{ with } \exists S \in \text{scc}(\Pi) : (|E(S_{\Pi_1})| \geq 1 \text{ and } |E(S_{\{r'\}})| \geq 1), r \in \mathcal{G}(\{r'\}), H_r = \{a\}, B_r^+ = \{a_{l+1}, \ldots, a_m\}, a \cup B_r^- = \{a, a_{m+1}, \ldots, a_n\} \tag{17}$$

Additional Rules for Foundedness of (Shared) SCCs:

$$[p_1(\mathbf{D}_1) \prec p_2(\mathbf{D}_2)] \vee [p_2(\mathbf{D}_2) \prec p_1(\mathbf{D}_1)] \leftarrow \qquad \text{for every SCC } S \in \text{scc}(\Pi) \text{ with } |E(S_{\Pi_1})| \geq 1, p_1(\mathbf{X}_1), p_2(\mathbf{X}_2) \in S, \mathbf{D}_1 \in \text{dom}(\mathbf{X}_1), \mathbf{D}_2 \in \text{dom}(\mathbf{X}_2), p_1(\mathbf{D}_1) \neq p_2(\mathbf{D}_2) \tag{18}$$

$$\leftarrow [p_1(\mathbf{D}_1) \prec p_2(\mathbf{D}_2)], [p_2(\mathbf{D}_2) \prec p_3(\mathbf{D}_3)], [p_3(\mathbf{D}_3) \prec p_1(\mathbf{D}_1)] \qquad \text{for every SCC } S \in \text{scc}(\Pi) \text{ with } |E(S_{\Pi_1})| \geq 1, p_1(\mathbf{X}_1), p_2(\mathbf{X}_2), p_3(\mathbf{X}_3) \in S, \mathbf{D}_1 \in \text{dom}(\mathbf{X}_1), \mathbf{D}_2 \in \text{dom}(\mathbf{X}_2), \mathbf{D}_3 \in \text{dom}(\mathbf{X}_3), p_1(\mathbf{D}_1) \neq p_2(\mathbf{D}_2), p_2(\mathbf{D}_2) \neq p_3(\mathbf{D}_3), p_1(\mathbf{D}_1) \neq p_3(\mathbf{D}_3) \tag{19}$$

$$\text{uf}_r(\mathbf{D}_{\mathbf{X}}) \leftarrow \text{uf}_{y_1}(\mathbf{D}_{\langle \mathbf{X}, y_1 \rangle}), \ldots, \text{uf}_{y_\ell}(\mathbf{D}_{\langle \mathbf{X}, y_\ell \rangle}), \neg [p(\mathbf{D}_Y) \prec h_r(D_{\mathbf{X}})] \qquad \text{for every SCC } S \in \text{scc}(\Pi) \text{ with } |E(S_{\Pi_1})| \geq 1, h(\mathbf{X}) \in S, p(\mathbf{Y}) \in B_r^+, \mathbf{D} \in \text{dom}(\langle \mathbf{X}, \mathbf{Y} \rangle), \mathbf{Y} = \langle y_1, \ldots, y_\ell \rangle, p(\mathbf{D}_{\mathbf{Y}}) \notin \mathcal{F} \tag{20}$$

$$\text{uf}_{r_r}(\mathbf{D}_{\mathbf{X}}) \leftarrow \neg [h_r(\mathbf{D}_Y) \prec h(D_{\mathbf{X}})] \qquad \text{for every SCC } S \in \text{scc}(\Pi) \text{ with } |E(S_{\Pi_1})| \geq 1, h(\mathbf{X}) \in S, p(\mathbf{Y}) \in B_r^+, \mathbf{D} \in \text{dom}(\langle \mathbf{X}, \mathbf{Y} \rangle), \mathbf{Y} = \langle y_1, \ldots, y_\ell \rangle, p(\mathbf{D}_{\mathbf{Y}}) \notin \mathcal{F} \tag{21}$$

Replace Rules (1) and (2) with:

$$h_r(\mathbf{D}) \vee \overline{h_r(\mathbf{D})} \leftarrow \qquad \text{for every } r \in \Pi_1, h(\mathbf{X}) \in \text{hpred}(r), \mathbf{D} \in \text{dom}(\mathbf{X}) \tag{22}$$

$$h(\mathbf{D}) \leftarrow h_r(\mathbf{D}) \qquad \text{for every } r \in \Pi_1, h(\mathbf{X}) \in \text{hpred}(r), \mathbf{D} \in \text{dom}(\mathbf{X}) \tag{23}$$

Replace Rules (8) with:

$$\text{sat}_r \leftarrow \text{sat}_{x_1}(\mathbf{D}_{\langle x_1 \rangle}), \ldots, \text{sat}_{x_\ell}(\mathbf{D}_{\langle x_\ell \rangle}), h(\mathbf{D}) \qquad \text{for every } r \in \Pi_1, h(\mathbf{X}) \in H_r, \mathbf{D} \in \text{dom}(\mathbf{X}), \mathbf{X} = \langle x_1, \ldots, x_\ell \rangle \tag{24}$$

Replace Rules (14) with:

$$\leftarrow \text{uf}_r(\mathbf{D}), h_r(\mathbf{D}) \qquad \text{for every } h(\mathbf{X}) \in \text{hpred}(\Pi_1), \mathbf{D} \in \text{dom}(\mathbf{X}), \{r_1, \ldots, r_m\} = \{r \in \Pi_1 \mid h(\mathbf{X}) \in H_r\} \tag{25}$$

$$\leftarrow \text{uf}_{r_r}(\mathbf{D}), h_r(\mathbf{D}) \qquad \text{for every } h(\mathbf{X}) \in \text{hpred}(\Pi_1), \mathbf{D} \in \text{dom}(\mathbf{X}), \{r_1, \ldots, r_m\} = \{r \in \Pi_1 \mid h(\mathbf{X}) \in H_r\} \tag{26}$$

Fig. 2: We extend the hybrid grounding procedure $\mathcal{H}$ of Figure 1 to the hybrid grounding procedure for shared cycles $\mathcal{H}_{lv}$. Consider $\Pi = \Pi_1 \cup \Pi_2$ s.t. Π is a normal (HCF) program. $\mathcal{H}_{lv}$, although similar to $\mathcal{H}$, adds level mappings to deal with shared cycles between Π_2 and Π_1. More precisely, Rules (15), (16) and (17) replace Rules (3), where non-ground rules contained in Π_2, i.e., those that shall be grounded via traditional grounding, are adapted to consider the level mapping. Further, by adding Rules (18), (19), (20) and (21), we ensure total level mappings, transitivity, and foundedness, respectively. Additionally, Rules (22), (23), (25), and (26) (and thereby replacing Rules (1), (2) and (14)) we ensure *local foundedness* for each rule in Π_1 s.t. it is known which h_r founds which h, which is necessary for level mappings. Lastly, we replace Rule (8) with Rule (24), so a rule is satisfied when h is in the answer set. This is important as it ensures that a rule is not "forced to found" some h_r, when it can not do that due to ordering constraints.