Characterizations for Simplifying ASP Programs under Abstraction: Working Abstract

Zeynep G. Saribatur and Stefan Woltran

Institute of Logic and Computation, TU Wien

Simplification of ASP programs while preserving their semantics has always been of interest, with works on equivalence-based rewriting [5, 12], partial evaluation [1, 7], and forgetting (see [8] for a recent survey).

The equivalence of logic programs is considered in the sense of the answer set semantics: a program P is equivalent to a program Q if $AS(P) = AS(Q)$. *Strong equivalence* [9] is a much stricter condition over the two programs: P and Q are strongly equivalent if, for any set R of rules, the programs $P \cup R$ and $Q \cup R$ are equivalent, shown as $AS(P \cup R) = AS(Q \cup R)$. This is the notion that makes it possible to simplify a part of a logic program without looking at the rest of it: if a subprogram Q of Π is strongly equivalent to a simpler program Q', then the Q is replaced by Q'. The works [11, 17, 4, 12] show ways of transforming programs by ensuring that the property holds. A more liberal notion is *uniform equivalence* [10, 13] where R is restricted to a set of facts. Then, a subprogram Q in Π can be replaced by a uniformly equivalent program Q' and the main structure will not be affected [3]. *Relativised* versions of strong and uniform equivalence [18] is defined for the case of having the newly added rules or facts, R, in a specific language. In all of these studies, P and Q are considered to be defined over the same signature. An interesting question then becomes what happens if these programs are defined over different signatures $\mathcal{A}$ and $\mathcal{A}'$, respectively, where $|\mathcal{A}'| \subseteq |\mathcal{A}|$ and a mapping is given to relate them together.

The recently introduced notion of abstraction in ASP [15, 16] considers two programs P and Q, where Q has a reduced signature according to some mapping m. There the aim is to construct an abstract program Q from P so that the answer sets of P are *over-approximated* in Q, i.e., any answer set in P can be mapped to some answer set in Q. Two forms of abstraction mappings have been investigated, one is on the propositional level and omits atoms which can be seen as forgetting [14], while the other is on the first-order level and is on domain abstraction, which clusters the constants in the Herbrand universe of the programs. Q is considered to be a *faithful* abstraction if it does not have any spurious answer sets. However, we are interested in those Q's which could replace P while fully preserving its semantics w.r.t. m, especially when there are newly added rules or facts that need to be considered which also get abstracted.

We are motivated by the following example, which currently cannot be captured by any of the representations existing in the literature. Consider the well-known planning problem blocksworld extended with multiple tables, where for a given initial layout of the blocks, e.g., block b_1 is located on top of table t_1, block b_2 is located on top of b_3 and block b_3 on top of table t_2, the aim is to pile them up on a chosen table, say t_1. Assume that initially the blocks can be on any of the tables. Then one could consider a mapping m that clusters those tables that are different than t_1 into one, say $\hat{t}$, so that all possible initial states I_i (and the goal state G) would be abstracted to some $\hat{I}$ (and $\hat{G}$) containing $\hat{t}$. Then the aim would be to find a Q that computes abstract plans by taking into account t_1 and the cluster table $\hat{t}$ instead, so that the computed abstract plan (described through the answer sets) in $Q \cup \hat{I} \cup \hat{G}$ can be mapped back to some plan in $P \cup I_i \cup G$, and vice versa, i.e., any of the original plans can be mapped to some abstract plan. Since I_i's and G are described through sets of facts, the desired relation hints on a form of uniform equivalence, however there is the difference of P and Q being of different signatures according to some m and R also getting modified with m.

The research on forgetting [6, 2] is actually towards the direction we are interested in, since there for a program P the aim is to construct a program $f(P, V)$ by applying an operator f on P to forget the atoms in V from the vocabulary, so that the resulting program is over $\overline{V}$. Among the many properties that have been investigated in the forgetting literature, the notion of Strong Persistence (**(SP)**) requires the correspondence between answer sets of the result of forgetting and those of the original program be preserved in the presence of any additional set of rules not containing the atoms to be forgotten, shown as $AS(f(P, V) \cup R) = AS(P \cup R)_{|\overline{V}}$, where the vocabulary of R is restricted to $\overline{V}$. Uniform Persistence (**(UP)**) notion considers R to be a set of facts. Such a restriction over the vocabulary of the added set of rules/facts prevents these notions from truly capturing the forgetting of atoms, since it avoids possible interferences between the rules in R and the rules in P containing the atoms to be forgotten.

To see this, consider again the blocksworld problem with multiple tables, where the tables are also colored. As the tables can be of different colors, there are multiple initial states for the planning problem. However, if colors are not of relevance to the computation of the plan, then one would expect to forget the details about the table colors and still obtain the same plans. However the current definition of **(UP)** does not capture this setting, since the R needs to be restricted to the remaining atoms, while in our setting the given initial state description given with R might still have to contain the color details. So what we want to achieve is actually $AS(f(P, V) \cup R_{|\overline{V}}) = AS(P \cup R)_{|\overline{V}}$ for all sets of facts $R \in \mathcal{C}$ describing the initial states.

In our ongoing work, we aim to provide characterizations for simplifying programs according to an abstraction mapping over their signature while preserving their semantics under the abstraction. By respecting the motivation behind strong/uniform equivalence, we aim to extend these notions to consider programs over different signatures related with an abstraction mapping and considering context programs which have their vocabularies also modified with the abstraction. We begin with focusing on abstraction by omission to relate the notions to forgetting, then will lift the results for the general abstraction mapping that clusters the atoms.

Acknowledgements

This research is supported by Austrian Science Fund (FWF) project T-1315.

References

1. Brass, S., Dix, J.: Characterizations of the disjunctive stable semantics by partial evaluation. J. Log. Program. **32**(3), 207–228 (1997)
2. Delgrande, J.P.: A knowledge level account of forgetting. Journal of Artificial Intelligence Research **60**, 1165–1213 (2017)
3. Eiter, T., Fink, M.: Uniform equivalence of logic programs under the stable model semantics. In: International Conference on Logic Programming. pp. 224–238. Springer (2003)
4. Eiter, T., Fink, M., Tompits, H., Woltran, S.: Simplifying logic programs under uniform and strong equivalence. In: Lifschitz, V., Niemelä, I. (eds.) Logic Programming and Nonmonotonic Reasoning. pp. 87–99. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
5. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Thiele, S.: Engineering an incremental ASP solver. In: Proc. ICLP. pp. 190–205 (2008)
6. Gonçalves, R., Knorr, M., Leite, J.: The ultimate guide to forgetting in answer set programming. In: Proc. of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR). pp. 135–144 (2016)
7. Janhunen, T., Niemelä, I., Seipel, D., Simons, P., You, J.H.: Unfolding partiality and disjunctions in stable model semantics. ACM TOCL **7**(1), 1–37 (Jan 2006)
8. Leite, J.: A bird's-eye view of forgetting in answer-set programming. In: Balduccini, M., Janhunen, T. (eds.) Proc. LPNMR. LNCS, vol. 10377, pp. 10–22. Springer (2017)
9. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic **2**(4), 526–541 (Oct 2001)
10. Maher, M.J.: Equivalences of logic programs. In: Shapiro, E. (ed.) Third International Conference on Logic Programming. pp. 410–424. Springer Berlin Heidelberg, Berlin, Heidelberg (1986)
11. Osorio, M., Navarro, J.A., Arrazola, J.: Equivalence in answer set programming. In: Pettorossi, A. (ed.) Logic Based Program Synthesis and Transformation. pp. 57–75. Springer Berlin Heidelberg (2002)
12. Pearce, D.: Simplifying logic programs under answer set semantics. In: Demoen, B., Lifschitz, V. (eds.) Logic Programming. pp. 210–224 (2004)
13. Sagiv, Y.: Optimizing datalog programs. In: Proceedings of the 6th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. pp. 349–362. PODS '87, ACM, New York, NY, USA (1987)
14. Saribatur, Z.G., Eiter, T.: A semantic perspective on omission abstraction in ASP. In: Proc. of KR (2020)
15. Saribatur, Z.G., Eiter, T.: Omission-based abstraction for answer set programs. Theory Pract. Log. Program. **21**(2), 145–195 (2021)
16. Saribatur, Z.G., Eiter, T., Schüller, P.: Abstraction for non-ground answer set programs. Artif. Intell. **300**, 103563 (2021)
17. Turner, H.: Strong equivalence made easy: nested expressions and weight constraints. Theory and Practice of Logic Programming **3**(4–5), 609–622 (2003)

18. Woltran, S.: Characterizations for relativized notions of equivalence in answer set programming. In: Alferes, J.J., Leite, J.A. (eds.) Logics in Artificial Intelligence, 9th European Conference, JELIA 2004, Lisbon, Portugal, September 27-30, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3229, pp. 161–173. Springer (2004)