A Unified View on Forgetting and Strong Equivalence Notions in Answer Set Programming

Zeynep G. Saribatur and Stefan Woltran

Institute of Logic and Computation, TU Wien

Abstract. Desiderata for AI systems that arise from legal or ethical requirements have gained increasing importance over the last years. Examples include the protection of private information within a knowledge base or the right of the eliminination of data on request, following the EU General Data Protection Regulation. In Answer Set Programming (ASP), a prominent rule-based language with roots in logic programming and non-monotonic logics, advanced reasoning problems in this direction include well-studied problems like forgetting or certain forms of program simplification. In particular, strong persistence **(SP)** forgetting, faithful abstractions, and, recently, strong simplifications (where the latter two can be seen as relaxed and strengthened notions of forgetting) appear closely related, especially given that they have characterizations through the semantics for strong equivalence. Yet, it remained unclear whether they can be captured in a uniform manner. In this extended abstract, we report on recent work that aims to bridge this gap by introducing a novel relativized equivalence notion, which is a relaxation of the recent simplification notion.

1 Introduction

Forgetting or discarding information that are not deemed necessary is crucial in human reasoning, as it allows to focus on the important details and to abstract over the rest. Such active or *intentional* forgetting is argued to enhance decision-making through flexibility under changing conditions and the ability to generalize [12]. Getting rid of (ir)relevant details through forgetting continues to motivate works in different subfields of AI [2], such as knowledge representation and reasoning (KR) [5] and symbolic machine learning [16]. Recent examples of forgetting within KR appear in action theories [11], explanations for planning [18] and argumentation [3, 1].

The theoretical underpinnings of forgetting has been investigated for classical logic and logic programming for over decades. Answer Set Programming (ASP), is a well established logic programming language, characterized by non-monotonic declarative semantics. Its non-monotonic nature resulted in various forgetting operators satisfying different desirable properties (see recent survey [7]). The property *strong persistence* **(SP)** [9] is considered to best capture the essence of forgetting in the context of ASP. The aim is to preserve all existing relations between the remaining atoms, by requiring that there be a correspondence between the answer sets of a program before and after forgetting a set of atoms, which is preserved in the presence of additional rules. This correspondence is formally defined as

$$AS(P \cup R)_{|\overline{A}} = AS(f(P, A) \cup R) \tag{1}$$

for all programs R over the universe $\mathcal{U}$ without containing atoms from A, where $f(P, A)$ is the resulting program of applying an operator f on P to forget about the set A of atoms, $AS(\cdot)$ denotes the collection of answer sets of a program, and $AS(\cdot)_{|\overline{A}}$ is their projection onto the remaining atoms.

When nothing is forgotten, **(SP)** matches the notion of *strong equivalence (SE)* [10] among programs, denoted as $AS(P \cup R) = AS(Q \cup R)$ for all programs R. [8] showed that **(SP)**-forgetting can only be done when the SE-models of the program adheres to certain conditions, which is motivated by *relativized* strong equivalence [20, 4], a relaxation of strong equivalence where the context programs can exclude some atoms.

The motivation to obtain ASP programs with a reduced signature also led to notion of abstraction by omission [13] by means of *over-approximation*, i.e., any answer set in program P can be mapped to some answer set in the abstracted program Q, which is denoted by $AS(P)_{|\overline{A}} \subseteq AS(Q)$, and also has been referred as *weakened Consequence* **(wC)** within forgetting [6]. [13] introduce a syntactic operator that obtains abstracted programs, and an automated abstraction and refinement methodology, that starts with a coarse abstraction and refines it upon encountering *spurious* answer sets (which do not have correspondence in P) until a fine-grained abstraction is achieved.

A desired abstraction property was considered to be *faithfulness* where Q does not contain a spurious answer set, i.e.,

$$AS(P)_{|\overline{A}} = AS(Q), \tag{2}$$

matching an instance of *Consequence Persistence* **(CP)**-forgetting [19]. The notion however does not truly preserve the semantics w.r.t. projection. The recent equivalence notion, called *strong simplification* [15], defined as[1]

$$AS(P \cup R)|_{\overline{A}} = AS(Q \cup R_{|\overline{A}}) \tag{3}$$

for all programs R, allows to capture the atoms that can be disregarded from the original program and also the context program, so that the simplified program can reason over the reduced vocabulary while ensuring that the semantics of the original program is preserved w.r.t. projection.

It is known that strong simplifications imply **(SP)**-forgetting [15] and the relation between omission abstraction and forgetting has also been studied [14].

[1] $R_{|\overline{A}}$ projects the positive body of the rules in R onto $\overline{A}$ and removes the rules with a negative body or head containing an atom from A.

A	B	Strong A-simplification relative to B
$\emptyset$	$\emptyset$	equivalence
$\emptyset$	$\mathcal{U}$	Strong Equivalence [17]
$\emptyset$	B	relativized Strong Equivalence [20]
A	$\mathcal{U}$	Strong Simplification [15]
A	$\overline{A}$	Strong Persistence [9]
A	C	$C \subseteq \overline{A}$, relativized Strong Persistence (newly defined)
A	$\emptyset$	Faithful Abstraction [13]

Table 1. Overview of the full spectrum of the relativized strong simplification notion introduced in this paper.

The characterizations for all of the mentioned notions have been established through the SE-models of programs, which characterizes strong equivalence. However until now it remained unclear how these notions come together.

In this extended abstract we report on our recent results that bridge this gap through a relaxation of the recent simplification notion, where on the context programs we allow for excluding some dedicated atoms: for sets A, B of atoms, we define the notion of *strong A-simplification relative to B* where (3) holds for all programs R over B. All of the above mentioned notions such as (relativized) strong equivalence, strong persistence, faithful abstractions and strong simplifications, then become special cases of this novel relativized equivalence notion, of which a summary can be seen in Table 1. Furthermore we show the conditions for relativized simplifiability and observe that the challenging part is for when the context programs do not contain all the atoms to remove/forget. We then show how the desired simplifications can be obtained by an operator that combines projection and a relaxation of **(SP)**-forgetting.

Our main contributions in this regard are as follows (i) We propose the novel concept of relativized strong simplification between programs, provide the necessary and sufficient conditions for testing relativized strong simplifiability, and give semantical characterizations of relativized strong simplifications; (ii) we discuss the full spectrum of this notion; (iii) we introduce a novel forgetting operator which is a combination of projection and a relaxation of SP-forgetting, which we introduce as relativized SP-forgetting; (iv) we conclude with complexity results. Details can be found in the full version at https://www.dbai.tuwien.ac.at/user/saribat/pub/sw23_unified.pdf

References

1. Baumann, R., Berthold, M.: Limits and possibilities of forgetting in abstract argumentation. In: Raedt, L.D. (ed.) Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022. pp. 2539–2545. ijcai.org (2022). https://doi.org/10.24963/ijcai.2022/352
2. Beierle, C., Timm, I.J.: Intentional forgetting: An emerging field in ai and beyond. KI-Künstliche Intelligenz **33**, 5–8 (2019)

3. Berthold, M., Rapberger, A., Ulbricht, M.: Forgetting aspects in assumption-based argumentation. In: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning. pp. 86–96. IJCAI Organization (2023)
4. Eiter, T., Tompits, H., Woltran, S.: On Solution Correspondences in Answer Set Programming. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. pp. 97–102 (2005)
5. Eiter, T., Kern-Isberner, G.: A brief survey on forgetting from a knowledge representation and reasoning perspective. KI – Künstliche Intelligenz (2018). https://doi.org/0.1007/s13218-018-0564-6
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. pp. 135–144 (2016)
7. Gonçalves, R., Knorr, M., Leite, J.: Forgetting in answer set programming a survey. Theory and Practice of Logic Programming **23**(1), 111156 (2023). https://doi.org/10.1017/S1471068421000570
8. Gonçalves, R., Knorr, M., Leite, J., Woltran, S.: On the limits of forgetting in answer set programming. Artif. Intell. **286**, 103307 (2020). https://doi.org/10.1016/j.artint.2020.103307
9. Knorr, M., Alferes, J.J.: Preserving strong equivalence while forgetting. In: Logics in Artificial Intelligence: 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings 14. pp. 412–425. Springer (2014)
10. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic **2**(4), 526–541 (Oct 2001). https://doi.org/10.1145/383779.383783
11. Luo, K., Liu, Y., Lespérance, Y., Lin, Z.: Agent abstraction via forgetting in the situation calculus. In: Giacomo, G.D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., Lang, J. (eds.) Proceedings of the 24th European Conference on Artificial Intelligence, ECAI 2020. Frontiers in Artificial Intelligence and Applications, vol. 325, pp. 809–816. IOS Press (2020). https://doi.org/10.3233/FAIA200170
12. Richards, B.A., Frankland, P.W.: The persistence and transience of memory. Neuron **94**(6), 1071–1084 (2017)
13. Saribatur, Z.G., Eiter, T.: Omission-based abstraction for answer set programs. In: Proc. of the 16th International Conference on Principles of Knowledge Representation and Reasoning. pp. 42–51. AAAI Press (2018)
14. Saribatur, Z.G., Eiter, T.: A semantic perspective on omission abstraction in ASP. In: Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning. pp. 733–737. IJCAI Organization (2020)
15. Saribatur, Z.G., Woltran, S.: Foundations for projecting away the irrelevant in asp programs. In: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning. pp. 614–624. IJCAI Organization (2023)
16. Siebers, M., Schmid, U.: Please delete that! why should i? explaining learned irrelevance classifications of digital objects. KI-Künstliche Intelligenz **33**(1), 35–44 (2019)
17. Turner, H.: Strong equivalence for logic programs and default theories (made easy). In: Logic Programming and Nonmotonic Reasoning: 6th International Conference, LPNMR 2001 Vienna, Austria, September 17–19, 2001 Proceedings 6. pp. 81–92. Springer (2001)

18. Vasileiou, S.L., Yeoh, W.: On generating abstract explanations via knowledge forgetting. In: ICAPS 2022 Workshop on Explainable AI Planning (2022)
19. Wang, Y., Wang, K., Zhang, M.: Forgetting for answer set programs revisited. In: IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013. pp. 1162–1168. IJCAI/AAAI (2013)
20. 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). https://doi.org/10.1007/978-3-540-30227-8_16