Extended abstract: f_{CASP} - A forgetting technique for XAI based on goal-directed constraint ASP models

Luciana Fidilio-Allende, Joaquin Arias

CETINIA, Universidad Rey Juan Carlos, Móstoles, Spain

Abstract

This paper is an extended abstract of: L. Fidilio-Allende, J. Arias, f_{CASP} : A forgetting technique for XAI based on goal-directed constraint ASP models, in: XXIII Jornadas sobre Programación y Lenguajes (PROLE), 2024. URL: https://hdl.handle.net/11705/PROLE/2024/13. [1].

Keywords

Privacy, Value Awareness, XAI, ASP, s(CASP), Forgetting

The automation of all sorts of processes through Artificial Intelligence (AI) systems has made significant progress. More recently, whether through self-regulation and soft law such as guidelines or through legal regulation (e.g., the General Data Protection Regulation (GDPR) or the Regulation on AI, both by the EU), it has become apparent that this development needs to be accompanied by measures that safeguard the fundamental rights and safety of people affected by AI systems. In this sense, Explainable Artificial Intelligence (XAI) [2] is of foremost importance to design trustworthy systems. Proposals such as s(LAW) [3], which are based on Answer Set Programming, have shown their ability to model values and explain the reasons for their decisions, thanks to their rule-based models. But these explanations could lead to the disclosure of sensitive information, such as details about victims of gender-based violence. This could violate the right to privacy and confidentiality, or even cause legal issues, among other concerns. Although explanations can be adjusted to prevent leaks, e.g., using the s(CASP) framework to control which elements are shown and/or hidden [4], adapting the models requires the application of techniques such as *forgetting* (variable elimination) to avoid revealing sensitive information during an audit. However, current forgetting techniques are mostly only applied in propositional ASP programs, and they have limitations dealing with even loops.

In this work, we present f_{CASP}, a new forgetting technique that supports the presence of non-stratified negations in Constraint Answer Set Programs. f_{CASP} is based on the dual rules of s(CASP), a goal-directed CASP reasoner, and therefore, we believe that it can be applied to generic CASP programs without grounding. We have validated our proposal by solving flagship examples from the literature, and we plan to use this technique in the context of school places allocation while preserving the privacy of victims of gender-based violence.

Table 1
Comparison of the more relevant forgetting operators vs. f_{CASP}

	(UP)	(SP)	Loops	Commutative	Predicates	Constraints
f_{SU} [5]	Yes	No	Yes	No	No	No
f_{SP} [6]	Yes	Maybe	No	No	No	No
f^*_{SP} [7]	Yes	Maybe	Yes	Yes	No	No
f_{AC} [8]	Yes	Yes	Yes	Yes	No	No
f_{CASP}	Yes	Yes	Yes	Yes	Maybe	Maybe

4th Workshop on Goal-directed Execution of Answer Set Programs (GDE'24), October 12, 2024
luciana.fidilio@urjc.es (L. Fidilio-Allende); joaquin.arias@urjc.es (J. Arias)
0009-0004-7779-8265 (L. Fidilio-Allende); 0000-0003-4148-311X (J. Arias)
© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

CEUR-WS.org/Vol-3799/abstract2GDE24.pdf

Figure 1: Sketch of the implementation of the predicate `f_casp/4`

```
1  f_casp (Flag, [Pred|Preds], P_0, P_Forgetting) :-
2      add_fact (Pred, P_0, P_11),
3      add_missing (Pred, P_11, P_12),
4      add_even_loop (Pred, P_12, P_13),          % Step 1
5      gen_dual (Pred, P_13, Dual_Pred),
6      add_dual (P_13, Dual_Pred, P_2),           % Step 2
7      forget_pred (Pred, P_2, P_31),
8      restore_even_loop (P_31, P_32),            % Step 3
9      f_casp (Flag, Preds, P_32, P_Forgetting).  % Repeat 1,2,3
10 f_casp (0, [], P_Forgetting, P_Forgetting).    % Skip Step 4
11 f_casp (1, [], P_Forgetting, P_Scasp) :-
12     scasp_even_loop(P_Forgetting, P_Scasp).    % Step 4
```

In Table 1, we **compare $\mathbf{f_{CASP}}$ with other relevant operators** such as those described in [9] and [10], evaluating its preliminary performance and properties. Through validation using literature examples, we believe that f_{CASP} can generate programs with the same answer sets, even when additional facts (when complying with UP) or propositional rules (when complying with SP) are added to both programs, avoiding the removed or auxiliary predicates. Additionally, we have tested the operator with programs involving even loops, and it can generate equivalent programs regardless of the order in which predicates are removed, that is, it is commutative. In the future, there is potential to extend it to programs with variables and constraints, thanks to the use of s(CASP) dual rules.

The preliminary **design of the algorithm** involves three steps repeated iteratively for each predicate marked to be forgotten, plus a final optional step.

Figure 1 shows an implementation sketch of the operator's steps:

- The first step (lines $2-4$) involves *adding auxiliary predicates (`neg_x`) and clauses* when the predicate to be forgotten is part of an even loop, is a fact, or is a missing predicate.
- The second step (lines 5 and 6) is *generating the dual rule of the predicate to forget*, being the dual rule the negated version of all the predicate's clauses.
- The third step (lines 7 and 8) is *forgetting the predicate*, replacing its appearances with the content of its clauses and its negation with the content of its dual rule.
- The final step (line 12) is *transforming the double negations (`not not`) into even loops*, as s(CASP) does not explicitly support them. This step is optional.

We have performed an **evaluation of $\mathbf{f_{CASP}}$** using examples from the literature.

In the first example, we forget *predicates in even loops*. To conserve the symmetry in answer sets [11], (preserving the predicates not forgotten in the answer sets even when adding additional rules), it is necessary to add additional predicates (`neg_x`) as strong persistence (SP) cannot be achieved (in some cases) without them [12, 8]. Below we can see the result of forgetting `p` and `q`, and the answer sets of both the original program and the generated one. As they have the same answer sets (ignoring the forgotten and auxiliary predicates), the programs are equivalent.

P_1 = Example 3 from [12]

`{a, p}, {b, q}`

```
1 a :- p.
2 b :- q.
3 p :- not q.
4 q :- not p.
```

$f_{CASP}(P_1, \{p, q\})$

`{a, neg_2}, {b, neg_1}`

```
1 a :- not not neg_2.
2 b :- not not neg_1.
3 neg_1 :- not not neg_1.
4 neg_2 :- not neg_1.
```

In the second example, we forget *predicates present in double negations*. This case cannot be resolved without the use of additional predicates. As we can see on the next example, both the original program and the one generated with f_{CASP} forgetting `p` are equivalent.

P_2 = Example 4 from [11]

```
{p, q}, {r}
```

```
p :- not not p.
q :- p.
r :- not p.
```

$f_{CASP}(P_2, \{p\})$

```
{q}, {r, neg_1}
```

```
q :- not neg_1.
r :- not not neg_1.
neg_1 :- not not neg_1.
```

In the third example we *forget multiple predicates regardless of the order.*

P_3 = Example 1 from [7]

```
{p}, {q}
```

```
a :- p, q.
q :- not p.
p :- not not p.
```

$f_{CASP}(P_3, \{p, q\})$ and $f_{CASP}(P_2, \{q, p\})$

```
{}, {neg_1}
```

```
a :- not neg_1, not not neg_1.
neg_1 :- not not neg_1.
```

In the final example, we *compare the performance of f_{CASP} with f_{AC}*, the operator that is closer to the desired properties. As we can see, f_{AC} may generate a program that outputs redundant answers, while the one generated with f_{CASP} preserves the original count.

P_4 = Example 5 from [8]

```
{c}
```

```
q :- not not q, b.
a :- q.
c :- not q.
```

$f_{AC}(P_4, \{q\})$.

```
{c}, {c, δ_q}
```

```
a :- b, δ_q.
c :- not δ_q.
c :- not b.
δ_q :- not not δ_q.
```

$f_{CASP}(P_4, \{q\})$.

```
{c, neg_1}
```

```
a :- not neg_1, b.
c :- not not neg_1.
c :- not b.
neg_1 :- not not neg_1.
neg_1 :- not b.
```

To evaluate the practicality of f_{CASP}, we have defined two (real) use cases in which we use the operator to remove private and confidential information.

In the first use case, we model the Spanish Organic Law 2/2004, May 3, Articles 116 and 117 and the Spanish Constitution Articles 27 and 149.1.30. This legislation establishes the criteria used for assigning public school places in the Comunidad de Madrid, Spain, when the number of applications for a given center is greater than the offer. After applying f_{CASP}, we can successfully remove the students' medical, socio-economical and gender-based violence related private information while conserving the original model decisions and explainability, complying with the need for transparency required by the applicable regulations such as the recent European AI Act.

In the second use case [13], we propose an automated decision-making system for energy assignment in agricultural cooperatives. In this case, the energy is assigned based on how fairly the workers are paid, pondering their salary and productivity, which are considered confidential and, in some cases, private. After applying forgetting, we can preserve the confidentiality of the stakeholders without affecting the decisions and justifications of the model, crucial to make these answers trustworthy.

In conclusion, we have presented a new forgetting operator, f_{CASP}, designed to work with goal-directed Answer Set Programs and support dual rules and double negations while being commutative. As potential lines of work for the future, we have identified the extension of the algorithm to support variables and constraints, formally determining and proving f_{CASP}'s properties, and applying the operator to real use cases.

References

[1] L. Fidilio-Allende, J. Arias, f_{CASP}: A forgetting technique for XAI based on goal-directed constraint ASP models, in: XXIII Jornadas sobre Programación y Lenguajes (PROLE), 2024. URL: https://hdl.handle.net/11705/PROLE/2024/13.

[2] D. Gunning, D. Aha, DARPA's Explainable Artificial Intelligence (XAI) Program, AI Magazine 40 (2019) 44–58. doi:10.1609/aimag.v40i2.2850.

[3] J. Arias, M. Moreno-Rebato, J. A. Rodriguez-García, S. Ossowski, Automated legal reasoning with discretion to act using s(LAW), 2023. doi:10.1007/s10506-023-09376-5.

[4] J. Arias, M. Carro, Z. Chen, G. Gupta, Justifications for goal-directed constraint answer set programming, 2020. doi:10.4204/EPTCS.325.12.

[5] R. Gonçalves, T. Janhunen, M. Knorr, J. Leite, On Syntactic Forgetting under Uniform Equivalence, in: European Conference on Logics in Artificial Intelligence, Springer, 2021, pp. 297–312.

[6] R. Gonçalves, M. Knorr, J. Leite, S. Woltran, When you must forget: Beyond Strong Persistence when Forgetting in Answer Set Programming, 2017.

[7] M. Berthold, On Syntactic Forgetting with Strong Persistence, in: Proceedings of the Int. Conf. on Principles of Knowledge Representation and Reasoning, volume 19, 2022, pp. 43–52.

[8] M. Berthold, R. Gonçalves, M. Knorr, J. Leite, Forgetting in Answer Set Programming with Anonymous Cycles, in: EPIA, Springer, 2019, pp. 552–565. doi:10.1007/978-3-030-30244-3_46.

[9] T. Eiter, G. Kern-Isberner, A Brief Survey on Forgetting from a Knowledge Representation and Reasoning Perspective, 2019.

[10] R. Gonçalves, M. Knorr, J. Leite, Forgetting in Answer Set Programming–A Survey, 2023.

[11] M. Knorr, J. J. Alferes, Preserving Strong Equivalence while Forgetting, in: Logics in Artificial Intelligence: 14th European Conference, JELIA 2014, Springer, 2014, pp. 412–425. doi:10.1007/978-3-319-11558-0_29.

[12] R. Gonçalves, M. Knorr, J. Leite, You can't always forget what you want: on the limits of forgetting in Answer Set Programming, in: Proceedings of the Twenty-second European Conference on Artificial Intelligence, 2016, pp. 957–965.

[13] L. Fidilio-Allende, J. Arias, Private-safe (logic-based) decision systems for energy assignment in agricultural cooperatives, in: Highlights in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection: International Workshops of PAAMS 2024, Salamanca, Spain, June 26—28, 2024, Proceedings, Springer, 2024.