ForgettingWeb

Matti Berthold
ScaDS.AI, Dresden/Leipzig,
Universität Leipzig,
Germany
berthold@informatik.uni-leipzig.de

Matthias Knorr
NOVA LINCS
Universidade Nova de Lisboa,
Portugal
mkn@fct.unl.pt

Daphne Odekerken
Utrecht University,
The Netherlands
National Police Lab AI,
Netherlands Police
d.odekerken@uu.nl

The relatively young area of *forgetting* is concerned with the removal of selective information, while preserving other knowledge. This might be useful or even necessary, for example, to simplify a knowledge base or to tend legal requests. In the last few years, there has been an ample amount of research in the field, in particular with respect to logic programs, spanning from theoretical considerations to more practical applications, starting at the conceptual proposal of forgetting, to suggestions of properties that should be satisfied, followed by characterizations of abstract classes of operators that satisfy these properties, and finally the definition of concrete forgetting procedures.

In this work we present novel Python implementations of all the forgetting procedures that have been proposed to date on logic programs. We provide them in a web interface, and hope to thereby give anybody who is interested a low-barrier overview of the landscape.

An Overview

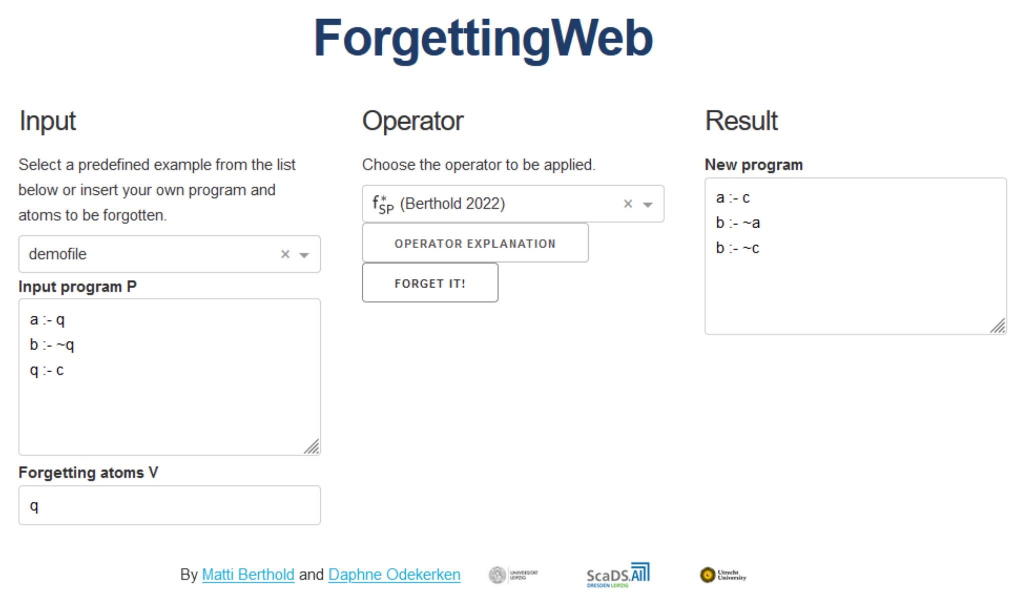
While dynamics in knowledge representation are often examined with respect to the addition of new information, there is an increasing amount of research on how it may be removed. *Forgetting*, or variable elimination, seeks to make a knowledge base independent of elements of the underlying formal language, while keeping as many logical connections between the remaining elements as possible. Though it was initially proposed as a semantical notion over classical formulas [LR94], in the context of logic programming forgotten atoms are usually also required to be removed syntactically. More broadly, there have been several suggestions for properties that a plausible forgetting procedure should satisfy [EW08, ZZ09, WZZZ12, WWZ13, KA14, DW15], cf. [GKL16b] for an extensive overview.

Along the way, there have also been a number of suggestions for concrete syntactical transformations to forget atoms from a program [ZF06, EW08, KA14, BGKL19, GJKL21, ACF$^+$22b, ACF$^+$22a, Ber22] that satisfy some of the proposed properties, or at least satisfy them whenever possible [GKL16a]. The hope for such operators is to produce forgetting results that in some way resemble their origin, while avoiding a computational blow-up as much as possible. Given a semantic definition of a class of forgetting operators, e.g. [GKLW17, GJK$^+$19], the construction of a program from the desired HT-models remains a baseline that can be employed to confirm the existence of a concrete forgetting operator [CF07].

The definitions of the syntactic forgetting procedures are rather involved, sometimes spanning over several pages of text. In order to make them more accessible, we compiled encodings as well as short explanations, of the operators (those that are defined over logic programs) and the construction of canonical programs into a web interface.[1] We hope to thereby give anybody who is interested a good overview of the material.

A screenshot of the interface is shown in Figure 1. In the leftmost column, users can specify a logic program and the atoms to be forgotten, or select a predefined sample input. Subsequently, they select the

[1]ForgettingWeb can be accessed here. Its source code is available here.



ForgettingWeb

Input

Select a predefined example from the list below or insert your own program and atoms to be forgotten.

demofile

Input program P

```
a :- q
b :- ~q
q :- c
```

Forgetting atoms V

q

Operator

Choose the operator to be applied.

f^{*}_{SP} (Berthold 2022)

OPERATOR EXPLANATION

FORGET IT!

Result

New program

```
a :- c
b :- ~a
b :- ~c
```

By Matti Berthold and Daphne Odekerken

Figure 1: Screenshot of ForgettingWeb.

forgetting operator of their choice from the dropdown in the middle column. These are implementations from operators proposed in the literature. For more details, the user presses the "Operator Explanation" button: this triggers a pop-up with an explanation of the operator and a reference to the paper in which it was proposed. After pressing the "Forget it!" button, the result program appears in the rightmost column.

An input program P is specified by one rule per line, where the head atoms are separated by '`;`', the head and the body are separated by '`:-`', and the body literals are separated by commas. As negation signs both the traditional '`not`' and the shorter '$\sim$' can be used. No dot at the end of a rule is required. The atoms of the forgetting set V are separated by commas. In the "Result" column on the right side, the new program has the same syntax as the input program.

The input of the counter-models construction aux_{cm} are tuples with delimiters '<' and '>', e.g.:

```
<ab,ab><a,ab><b,ab><,ab>
<a,a><,>
```

Each character within an element of a tuple is assumed to be a single element of a set.

Acknowledgements

The authors have been arranged alphabetically. The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany and by the Sächsische Staatsministerium für Wissenschaft Kultur und Tourismus in the program Center of Excellence for AI-research "Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig", project identification number: ScaDS.AI. This work was partially supported by FCT project FORGET (PTDC/CCI-INF/32219/2017) and by FCT project NOVA LINCS (UIDB/04516/2020).

References

[ACF$^+$22a] F. Aguado, P. Cabalar, J. Fandinno, D. Pearce, G. Pérez & C. Vidal (2022): *A polynomial reduction of forks into logic programs*. Artif. Intell. 308, p. 103712, doi:10.1016/j.artint.2022.103712.

[ACF$^+$22b] F. Aguado, P. Cabalar, J. Fandinno, D. Pearce, G. Pérez & C. Vidal (2022): *Syntactic ASP Forgetting with Forks*. In: Proceedings of (LPNMR-22), Lecture Notes in Computer Science 13416, Springer, pp. 3–15, doi:10.1007/978-3-031-15707-3_1.

[Ber22] M. Berthold (2022): *On Syntactic Forgetting with Strong Persistence*. In: Proceedings of (KR-22), doi:10.24963/kr.2022/5.

[BGKL19] M. Berthold, R. Gonçalves, M. Knorr & J. Leite (2019): *A Syntactic Operator for Forgetting that Satisfies Strong Persistence*. Theory and Practice of Logic Programming 19(5-6), p. 1038–1055, doi:10.1017/S1471068419000346.

[CF07] P. Cabalar & P. Ferraris (2007): *Propositional theories are strongly equivalent to logic programs*. Theory and Practice of Logic Programming 7(6), pp. 745–759, doi:10.1017/S1471068407003110.

[DW15] J.P. Delgrande & K. Wang (2015): *A Syntax-Independent Approach to Forgetting in Disjunctive Logic Programs*. In: Proceedings of (AAAI-15), pp. 1482–1488, doi:10.1609/aaai.v29i1.9402.

[EW08] T. Eiter & K. Wang (2008): *Semantic forgetting in answer set programming*. Artificial Intelligence 172(14), pp. 1644–1672, doi:10.1016/j.artint.2008.05.002.

[GJK$^+$19] R. Gonçalves, T. Janhunen, M. Knorr, J. Leite & S. Woltran (2019): *Forgetting in Modular Answer Set Programming*. In: Proceedings of (AAAI-19), AAAI Press, pp. 2843–2850, doi:10.1609/aaai.v33i01.33012843.

[GJKL21] R. Gonçalves, T. Janhunen, M. Knorr & J. Leite (2021): *On Syntactic Forgetting Under Uniform Equivalence*. In: Proceedings of (JELIA-21), Lecture Notes in Computer Science 12678, Springer, pp. 297–312, doi:10.1007/978-3-030-75775-5_20.

[GKL16a] R. Gonçalves, M. Knorr & J. Leite (2016): *The Ultimate Guide to Forgetting in Answer Set Programming*. In: Proceedings of (KR-16), pp. 135–144.

[GKL16b] R. Gonçalves, M. Knorr & J. Leite (2016): *You Can't Always Forget What You Want: On the Limits of Forgetting in Answer Set Programming*. In: Proceedings of (ECAI-16), pp. 957–965, doi:10.3233/978-1-61499-672-9-957.

[GKLW17] R. Gonçalves, M. Knorr, J. Leite & S. Woltran (2017): *When you must forget: Beyond strong persistence when forgetting in answer set programming*. Theory and Practice of Logic Programming 17(5-6), pp. 837–854, doi:10.1017/S1471068417000382.

[KA14] M. Knorr & J.J. Alferes (2014): *Preserving Strong Equivalence while Forgetting*. In: Proceedings of (JELIA-14), LNCS 8761, Springer, pp. 412–425, doi:10.1007/978-3-319-11558-0_29.

[LR94] F. Lin & R. Reiter (1994): *Forget it*. In: Working Notes of AAAI Fall Symposium on Relevance, pp. 154–159.

[WWZ13] Y. Wang, K. Wang & M. Zhang (2013): *Forgetting for Answer Set Programs Revisited*. In: Proceedings of (IJCAI-13), IJCAI/AAAI, pp. 1162–1168. Available at `http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6807`.

[WZZZ12] Y. Wang, Y. Zhang, Y. Zhou & M. Zhang (2012): *Forgetting in Logic Programs under Strong Equivalence*. In: Proceedings of (KR-12), AAAI Press.

[ZF06] Y. Zhang & N.Y. Foo (2006): *Solving logic program conflict through strong and weak forgettings*. Artificial Intelligence 170(8-9), pp. 739–778, doi:10.1016/j.artint.2006.02.002.

[ZZ09] Y. Zhang & Y. Zhou (2009): *Knowledge forgetting: Properties and applications*. Artificial Intelligence 173(16), pp. 1525–1537, doi:10.1016/j.artint.2009.07.005.