Neural Decision–Propagation for Answer Set Programming (Extended Abstract)*

Thomas Eiter[1], Katsumi Inoue[2] and Sota Moriyama[3,2]

[1]Vienna University of Technology, Austria
[2]National Institute of Informatics, Japan
[3]The Graduate University for Advanced Studies, SOKENDAI, Japan

Abstract

Integration of Answer Set Programming (ASP) with neural networks has emerged as a promising tool in Neuro-symbolic AI. While existing approaches extend the capabilities of ASP to real world domains, their reasoning pipelines depend on classical solvers, which is a bottleneck for scalability. To tackle this problem, we propose a new method to compute stable models, called decision–propagation (DProp), which alternates falsity decisions and truth propagations. Successful DProp computations are shown to capture the stable model semantics. We then develop Neural DProp (NDProp), a differentiable extension of DProp with neural computation for decisions and fuzzy evaluation for propagations. We evaluate the capabilities of NDProp for learning decision heuristics as well as neuro-symbolic integration, and compare it with existing neuro-symbolic approaches. The results show that NDProp can learn to efficiently compute stable models, and it improves accuracy and scalability on neuro-symbolic benchmarks.

Keywords

Answer Set Programming, Neuro-symbolic AI, Fuzzy logic, Differentiable solving

1. Introduction

Neuro-symbolic AI aims to develop systems that bring together robust learning in neural networks with reasoning via symbolic representations [1, 2]. There has been a growing interest in using logic programming as a general symbolic reasoning framework for such integrated systems [3, 4, 5]. Answer Set Programming (ASP), a declarative programming paradigm widely used for knowledge representation and reasoning tasks [6, 7], is used in such systems. Neuro-symbolic integration in ASP has been attempted in various domains such as Visual Question Answering [8], compliance checking [9] and many more [10].

One prominent approach to accomplishing neuro-symbolic integration with ASP is to train neural networks with symbolic knowledge expressed with ASP. NeurASP [3] integrates neural atoms into ASP programs, allowing neural network outputs to be directly incorporated into symbolic reasoning and learning. SLASH [11] extends this line of work with neural-probabilistic predicates and removes low-probability predicate outcomes before ASP solving, thereby reducing the number of stable models that must be considered. However, these approaches still largely follow a hybrid design in which neural components and symbolic ASP solvers remain separated by an interface: continuous neural predictions must be translated into symbolic atoms/literals, constraints, or probabilistic facts before reasoning can be performed. This introduces a combinatorial grounding problem, as the system must reason over multiple candidate associations between perceptual entities and symbolic objects before being processed by an ASP solver.

To address these challenges, we propose a novel method for computing stable models called *decision–propagation* (DProp). A DProp run iteratively alternates between the following two phases: (i) decision, which selects undecided atoms to commit to false non-deterministically, and (ii) propagation, which

Joint Proceedings of the Workshops and Doctoral Consortium of the 42nd International Conference on Logic Programming (ICLP 2026), co-located with the Federated Logic Conference (FLoC 2026), July 18–19 and 24–25, 2026, Lisbon, Portugal.

* An extended version of this paper has been accepted at IJCAI-ECAI 2026. The full version is available at https://arxiv.org/abs/2605.01797.

thomas.eiter@tuwien.ac.at (T. Eiter); inoue@nii.ac.jp (K. Inoue); sotam@nii.ac.jp (S. Moriyama)



© 2026 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
published 2026-11-03

CEUR-WS.org/Vol-4311/ASPOCP_short_01.pdf

updates the current partial assignment by deriving consequences of the program under the current assumptions. We then prove that successful DProp computations (runs that end without contradictions) capture the stable model semantics.

Subsequently, we propose *Neural DProp* (NDProp), a differentiable extension of DProp. NDProp uses neural networks for falsity decisions and fuzzy logic operators for propagations. As NDProp is identical to DProp in the crisp case, it is able to prove stability when the outputs are binary under a certain condition, making it easy to train in an unsupervised manner. These make the integration of NDProp into neuro-symbolic systems seamless, while still preserving a clear connection to classical stable model semantics. Furthermore, the learned decision steps and parallel capabilities via GPUs make this approach significantly more scalable.

To demonstrate the effectiveness of NDProp, we conduct neuro-symbolic integration to determine whether using NDProp has merits over existing approaches that use classical solvers. The results show that NDProp performs better than the respective baselines; it is superior with respect to accuracy, and has much higher inference speed than existing ASP-based approaches. Overall, this approach is the first to provide a fully differentiable solver for ASP based on a novel fixpoint semantics, and is effective at mitigating the problems that arise from the interface between components in existing approaches.

2. Preliminaries

Answer Set Programming. A *rule* r in a *(normal logic) program* P has the form

$$a \leftarrow b_1, \ldots, b_m, not\, c_1, \ldots, not\, c_l, \tag{1}$$

where a, b_i, and c_j are atoms and not denotes default negation. For a rule r of the form (1), the atom a is the *head* of the rule denoted as $H(r)$, and $b_1, \ldots, b_m, not\, c_1, \ldots, not\, c_l$ is the *body* of the rule denoted as $B(r)$. The sets $B^+(r) = \{b_1, \ldots, b_m\}$ and $B^-(r) = \{c_1, \ldots, c_l\}$ denote the positive body atoms and the atoms occurring under default negation, respectively. Given two interpretations I, J and a normal logic program P, an immediate consequence operator with respect to J [12] is defined as

$$C_{P,J}(I) = \left\{ H(r) \in \mathcal{A} \mid B^+(r) \subseteq I, B^-(r) \cap J = \emptyset \right\}.$$

Fuzzy Interpretations. Fuzzy Answer Set Programming (FASP) extends ASP by interpreting atoms with a truth degree in the interval $[0, 1]$, making it possible to reason with vague or incomplete information [13, 14]. Let $\mathcal{A}$ be a set of propositional atoms. A *fuzzy interpretation* is a mapping $I : \mathcal{A} \to [0, 1]$ assigning each atom a degree of truth. To evaluate rule bodies, FASP fixes a *t-norm* $\otimes : [0, 1]^2 \to [0, 1]$ (fuzzy conjunction), i.e., a commutative, associative and monotone operator with unit 1. Typical t-norms include Gödel, Product, and Łukasiewicz. Similarly, *t-conorm* $\oplus : [0, 1]^2 \to [0, 1]$ represents fuzzy disjunction. For a rule r of the form (1), the degree to which the body is satisfied under I is

$$I(B(r)) \;:=\; \Big(\bigotimes_{i=1}^{m} I(b_i)\Big) \;\otimes\; \Big(\bigotimes_{j=1}^{l} \big(1 - I(c_j)\big)\Big),$$

where default negation is interpreted using $I(not\, c) = 1 - I(c)$.

3. Decision–Propagation for ASP

In this section, we describe DProp computations, and show that successful DProp computations capture stable model semantics. DProp uses a forward chaining approach similar to [15, 16], with the biggest difference being that the decisions are made at an atom and not at a rule level, making it much simpler to differentiably extend.

Formally, DProp operates on an atom set $\mathcal{A}$ of P, with each state splitting $\mathcal{A}$ into disjoint true, false, and undecided sets T, F, and $U = \mathcal{A} \setminus (T \cup F)$, respectively. With initial sets $T^0 = C^{\infty}_{P,\mathcal{A}}(\emptyset)$, $F^0 = \emptyset$, and $U^0 = \mathcal{A} \setminus (T^0 \cup F^0)$, a *run* in DProp performs the following iterations ($k \geq 1$) until termination:

1. **Decision.** Select an undecided atom $a \in U^{k-1}$ (via some policy), then update $F^k = F^{k-1} \cup \{a\}$.
2. **Propagation.** With F^k fixed, compute $T^k = C^{\infty}_{P,\mathcal{A}\setminus F^k}(T^{k-1})$ and set $U^k = U^{k-1} \setminus (T^k \cup F^k)$.
3. **Termination.** If $U^k = \emptyset$, stop and return T^k; otherwise continue to the next decision.

When a run terminates, we write $k = \infty$, obtaining T^∞, F^∞, and U^∞. Here, both T and F increase monotonically because Propagation and Decision never remove atoms from them. The following theorem states that successful DProp computations (computations that end with $U^\infty = \emptyset$ and $T^\infty \cap F^\infty = \emptyset$) capture stable model semantics, and vice versa.

Theorem 1. *Let P be a finite program and $S \subseteq \mathcal{A}_P$, where $\mathcal{A}_P$ is the set of atoms occurring in P. Then S is a stable model of P if and only if S is the final true set of a successful DProp computation.*

4. Neural Decision–Propagation

We now instantiate DProp with differentiable operators and neural decision modules, yielding a neural solver NDProp that is an extension of DProp. Corresponding to the true and false sets T and F, we consider true and false *degrees* $(\tau, \phi) \in [0,1]^n \times [0,1]^n$.

4.1. Neural Extensions of DProp

Each atom a_i carries a true and false degree τ_i and ϕ_i. Literal evaluations are done with the val operator, which outputs for positive and negative literals $\mathrm{val}(a_i) = \tau_i$ and $\mathrm{val}(not\, a_i) = \phi_i$, respectively. Regarding (τ_i, ϕ_i), values near $(0,0)$ indicate undecided atoms, $(1,0)$ denotes derived truth, $(0,1)$ denotes explicit falsity, and $(1,1)$ encodes contradiction. For NDProp, we combine these into a single membership score

$$p_i = \tfrac{1}{2}(\tau_i + 1 - \phi_i) \in [0,1],$$

which serves as the atom's soft truth value indicating whether the atom should be included in the answer set or not. This becomes binary iff (τ_i, ϕ_i) is $(1,0)$ or $(0,1)$. We denote by (τ^t, ϕ^t) the degrees at iteration t.

Decision Phase. Decisions update only ϕ while holding τ fixed, as in DProp. Given a t-norm $\otimes$ and t-conorm $\oplus$, the undecided degree μ_i for atom a_i at iteration t is defined as $\mu_i^t := 1 - (\tau_i^t \oplus \phi_i^t) = (1 - \tau_i^t) \otimes (1 - \phi_i^t) \in [0,1]$, which is the fuzzy conjunction of the degree to which a_i is not true and the degree to which a_i is not false. We then produce decisions via a neural decision module (RNN) that takes the current input and hidden state s^{t-1} and yields decision logits d^t and updated hidden state s^t:

$$(d^t, s^t) = \mathrm{RNN}_{\mathrm{dec}}\big([\tau^t \,\|\, \phi^t \,\|\, \mu^t],\, s^{t-1}\big), \qquad \delta^t = \sigma(d^t),$$

where $\|$ denotes concatenation and $\sigma : \mathbb{R} \to (0,1)$ is the sigmoid function. RNNs retain historical information with the hidden states s^t, making it suitable for our approach that requires producing decisions over multiple steps. The falsity degrees move towards 1 according to the decision δ^t and the undecided degree μ^t:

$$\phi^{t+1} = \phi^t + (\mu^t \odot \delta^t), \tag{2}$$

with $\odot$ denoting elementwise multiplication. Note that ϕ^{t+1} remains in $[0,1]$ since $\mu^t \le 1 - \phi^t$ and $\delta^t \in [0,1]$. Because μ^t becomes zero for already decided atoms, this additive rule directly mirrors the discrete falsity decision when (τ^t, ϕ^t) and δ^t are binary.

Propagation Phase. Propagation updates only τ while holding ϕ fixed, as in DProp. Given a t-norm $\otimes$ and t-conorm $\oplus$, for each rule r_j, its rule body contributes support

$$\rho_j(\tau, \phi) := \bigotimes_{\ell \in B(r_j)} \mathrm{val}(\ell), \qquad (\mathcal{C}_{P,\phi}(\tau))_i := \bigoplus_{j : H(r_j) = a_i} \rho_j(\tau, \phi).$$

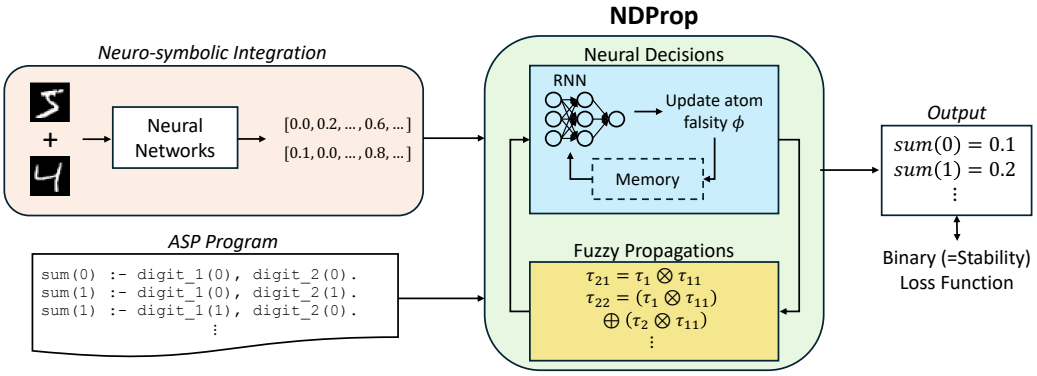


Figure 1: NDProp pipeline. The neuro-symbolic integration is added for specific instantiations, and the arrow leading from the neuro-symbolic block to the ASP program is added only when the backbone neural networks are being trained.

Given ϕ^{t+1}, we update τ from τ^t by iterating $\mathcal{C}_{P,\phi^{t+1}}$ until convergence: $\tau^{t+1} = \mathcal{C}^{\infty}_{P,\phi^{t+1}}(\tau^t)$. In practice, we stop the iteration early when the maximum change between successive applications of $\mathcal{C}_{P,\phi^{t+1}}$ falls below a fixed convergence threshold, or the iteration reaches a given pre-defined value.

Based on the above definitions, we can show that NDProp coincides with DProp in the crisp case.

Proposition 1 (Crisp Case). *Let $(\tau, \phi) \in \{0,1\}^n \times \{0,1\}^n$, $\delta \in \{0,1\}^n$, and propagation compute the fixpoint of $\mathcal{C}_{P,\phi}$. Then, one update of NDProp coincides with one iteration of DProp.*

4.2. Loss Alignment

We prove that enforcing NDProp to output binary values is equivalent to enforcing stability.

Theorem 2 (Binary terminal state yields stability). *Let P be a finite program and $S \subseteq \mathcal{A}_P$, where $\mathcal{A}_P$ is the set of atoms occurring in P. Then S is a stable model of P if and only if propagation is applied until convergence with the resulting membership score $p_i \in \{0,1\}$ and $S = \{a_i \mid p_i = 1\}$.*

From Theorem 2, we can conclude that when the output p of NDProp is trained with a loss function that enforces binary outputs, it inherently forces the NDProp to learn to output stable models, given that the final propagation is done until convergence.

5. Experiments

As NDProp can approximate answer sets differentiably, it is natural to combine with existing neural networks based architectures to perform neuro-symbolic inference. In all settings, the input consists of one or more MNIST images, and supervision is provided only through arithmetic relations between the underlying concepts, rather than through per-image labels [4, 17, 18]. The goal is therefore to jointly learn perception using symbolic reasoning from weak, relational supervision.

Setup. In this experiment, we evaluate NDProp on MNIST-based arithmetic reasoning tasks that are commonly used as neuro-symbolic benchmarks [3, 4]. These include one- and two-digit addition, Add2x2, Apply2x2, and three- and five-digit membership. In the addition tasks, the input consists of digit images, and the output is their sum. In Add2x2, the input is a 2×2 grid of digit images, and the outputs are the sums of the rows and columns. Apply2x2 is similar, but the arithmetic operators are also given as handwritten images. In the membership tasks, the system predicts whether a queried digit is included in a set of digit images. For NDProp, we use the setup with iteration of 1, hidden dimension of 8, logical dimension of 2, learning rate of 1×10^{-3}, and product t-norm.

We compare the performance of NDProp with ASP-based neuro-symbolic systems: NeurASP [3] and SLASH [11]. Experiments are conducted with a maximum of 10 epochs for NeurASP, SLASH

	Addition			Addition2			Add2x2			Apply2x2			Member3			Member5		
	NDProp	NASP	SLASH	NDProp	NASP	SLASH	NDProp	NASP	SLASH	NDProp	NASP	SLASH	NDProp	NASP	SLASH	NDProp	NASP	SLASH
Train	**160**	1612	430	**235**	TO	2374	**692**	TO	TO	346	385	**166**	**110**	2829	TO	**969**	TO	TO
Test	**0.37**	24.25	9.94	**1.28**	33.90	3.73	**1.15**	41.59	3.75	**1.52**	5.31	4.32	**0.79**	5.13	3.08	**4.40**	496.4	6.38
R.Acc	**99.0**	98.5	98.9	98.9	98.5	**99.0**	**99.0**	98.6	98.9	**100.0**	**100.0**	**100.0**	**97.3**	96.8	97.2	**97.0**	11.6	8.9
I.Acc	**98.2**	97.1	97.8	96.1	94.6	**96.5**	**95.7**	93.70	95.2	**100.0**	**100.0**	**100.0**	**99.0**	98.6	98.7	**97.5**	59.7	60.3

Table 1
Neuro-symbolic MNIST results for NDProp, NeurASP (NASP) and SLASH, all trained for 10 epochs. Metrics are Train: Training time, Test: Testing time, R.Acc: Recognition Accuracy (%), and I.Acc: Inference accuracy (%).

and NDProp, setting timeout to 3000 seconds; the training is cut off when the budget is reached, and accuracy is checked at that point.

ASP Encodings. In contrast to prior neuro-symbolic ASP systems that use neural predicates to interface with perception modules [3], we adopt a different strategy. We embed the output of neural models as fuzzy facts into a fixed ASP program that encodes the arithmetic constraints, as shown in Figure 1. While this may seem to force NDProp to perform as a FASP solver, the objective is to produce binary outputs, explicitly forcing the backbone perception model to predict binary values as well.

Results. Table 1 reports recognition accuracy, inference accuracy (accuracy of final output), and runtime (seconds) for both training and testing splits, averaged over 3 runs. Across the six neuro-symbolic tasks, NDProp achieves higher recognition and inference accuracy than NeurASP and SLASH on five, with the exception of Addition2, where SLASH has slightly higher recognition and inference accuracy. Empirically, this shows the benefits of using end-to-end trainable systems: jointly optimizing perception and reasoning leads to more cohesive and consistent outputs. Furthermore, NDProp has substantially lower training time than NeurASP and SLASH in most settings. In particular, for Member5, both NeurASP and SLASH fail to complete even one epoch within the time budget, leading to very low accuracy. This is more prominent in the testing speed, which is an average of 41.8 times faster than NeurASP, and 6.8 times faster than SLASH. This is mainly due to the GPU capabilities of NDProp; GPUs allow for higher parallelization, allowing our model to process multiple instances with one forward pass.

6. Conclusion

We have presented decision–propagation (DProp), a novel method for computing stable models. DProp alternates between falsity decisions and truth propagations, and a successful DProp computation is shown to capture the stable model semantics. We then proposed NDProp, a differentiable extension of DProp that uses neural decisions and fuzzy propagations, and show that it coincides with DProp at the binary limit. Experiments on neuro-symbolic benchmarks show that NDProp outperforms existing ASP-based neuro-symbolic systems in terms of accuracy and runtime, demonstrating the benefits of end-to-end trainable systems for neuro-symbolic inference. The results display the benefits of using a differentiable solver that minimizes the interfacing between the neural and symbolic components.

Throughout the experiments, we have noticed that some t-norms work better than others in each setting. An extensive study of t-norms in particular settings is left out for future work. Another direction to explore is whether the semantics of NDProp aligns with FASP. At the moment, NDProp is aimed at normal logic programs, but with modifications, it could be used to solve fuzzy variants. Finally, extending the framework to support often used constructs such as constraints, choices and aggregates is left for future work as well.

Acknowledgements

This work has been supported by JSPS KAKENHI Grant Number JP25K03190, JST CREST Grant Number JPMJCR22D3 and Grant-in-Aid for JSPS Fellows Grant Number JP26KJ1236, Japan. This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/COE12. The authors thank colleagues in the KBS Group at TU Wien for helpful discussions.

Declaration of Generative AI

During the preparation of this work, the author used ChatGPT for Grammar and spelling checks. After using this tool, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] A. d'Avila Garcez, L. C. Lamb, Neurosymbolic AI: the 3rd wave, Artif. Intell. Rev. 56 (2023) 12387–12406. URL: https://doi.org/10.1007/s10462-023-10448-w. doi:10.1007/S10462-023-10448-W.

[2] W. Wang, Y. Yang, F. Wu, Towards data-and knowledge-driven AI: A survey on neuro-symbolic computing, IEEE Trans. Pattern Anal. Mach. Intell. 47 (2025) 878–899. URL: https://doi.org/10.1109/TPAMI.2024.3483273. doi:10.1109/TPAMI.2024.3483273.

[3] Z. Yang, A. Ishay, J. Lee, Neurasp: Embracing neural networks into answer set programming, in: C. Bessiere (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, ijcai.org, 2020, pp. 1755–1762. URL: https://doi.org/10.24963/ijcai.2020/243. doi:10.24963/IJCAI.2020/243.

[4] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, L. D. Raedt, Neural probabilistic logic programming in DeepProbLog, Artif. Intell. 298 (2021) 103504. URL: https://doi.org/10.1016/j.artint.2021.103504. doi:10.1016/J.ARTINT.2021.103504.

[5] R. Evans, E. Grefenstette, Learning explanatory rules from noisy data, J. Artif. Intell. Res. 61 (2018) 1–64. URL: https://doi.org/10.1613/jair.5714. doi:10.1613/JAIR.5714.

[6] V. Lifschitz, What is answer set programming?, in: D. Fox, C. P. Gomes (Eds.), Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008, AAAI Press, 2008, pp. 1594–1597. URL: http://www.aaai.org/Library/AAAI/2008/aaai08-270.php.

[7] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Commun. ACM 54 (2011) 92–103. URL: https://doi.org/10.1145/2043174.2043195. doi:10.1145/2043174.2043195.

[8] T. Eiter, N. Higuera, J. Oetsch, M. Pritz, A neuro-symbolic ASP pipeline for visual question answering, Theory Pract. Log. Program. 22 (2022) 739–754. URL: https://doi.org/10.1017/S1471068422000229. doi:10.1017/S1471068422000229.

[9] V. Barbara, M. Guarascio, N. Leone, G. Manco, A. Quarta, F. Ricca, E. Ritacco, Neuro-symbolic AI for compliance checking of electrical control panels, Theory Pract. Log. Program. 23 (2023) 748–764. URL: https://doi.org/10.1017/s1471068423000170. doi:10.1017/S1471068423000170.

[10] M. Borroto, A. Ielo, G. Mazzotta, F. Ricca, Answer set programming and neurosymbolic ai: Applications and future perspectives (invited talk), in: P. Bruno, F. Calimeri, F. Cauteruccio, G. Terracina (Eds.), Hybrid Models for Coupling Deductive and Inductive Reasoning, Springer Nature Switzerland, Cham, 2025, pp. 1–21.

[11] A. Skryagin, D. Ochs, D. S. Dhami, K. Kersting, Scalable neural-probabilistic answer set programming, J. Artif. Intell. Res. 78 (2023) 579–617. URL: https://doi.org/10.1613/jair.1.15027. doi:10.1613/JAIR.1.15027.

[12] A. Van Gelder, The alternating fixpoint of logic programs with negation, J. Comput. Syst. Sci. 47 (1993) 185–221. URL: https://doi.org/10.1016/0022-0000(93)90024-Q. doi:10.1016/0022-0000(93)90024-Q.

[13] D. V. Nieuwenborgh, M. D. Cock, D. Vermeir, Fuzzy answer set programming, in: M. Fisher, W. van der Hoek, B. Konev, A. Lisitsa (Eds.), Logics in Artificial Intelligence, 10th European Conference, JELIA 2006, Liverpool, UK, September 13-15, 2006, Proceedings, volume 4160 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 359–372. URL: https://doi.org/10.1007/11853886_30. doi:10.1007/11853886_30.

[14] M. Alviano, R. Peñaloza, Fuzzy answer sets approximations, Theory Pract. Log. Program. 13 (2013) 753–767. URL: https://doi.org/10.1017/S1471068413000471. doi:10.1017/S1471068413000471.

[15] C. Lefèvre, C. Béatrix, I. Stéphan, L. Garcia, Asperix, a first-order forward chaining approach for answer set computing, Theory Pract. Log. Program. 17 (2017) 266–310. URL: https://doi.org/10.1017/S1471068416000569. doi:10.1017/S1471068416000569.

[16] A. Dal Palù, A. Dovier, E. Pontelli, G. Rossi, GASP: answer set programming with lazy grounding, Fundam. Informaticae 96 (2009) 297–322. URL: https://doi.org/10.3233/FI-2009-180. doi:10.3233/FI-2009-180.

[17] A. L. Gaunt, M. Brockschmidt, N. Kushman, D. Tarlow, Differentiable programs with neural libraries, in: D. Precup, Y. W. Teh (Eds.), Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of *Proceedings of Machine Learning Research*, PMLR, 2017, pp. 1213–1222. URL: http://proceedings.mlr.press/v70/gaunt17a.html.

[18] E. Tsamoura, T. M. Hospedales, L. Michael, Neural-symbolic integration: A compositional perspective, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, AAAI Press, 2021, pp. 5051–5060. URL: https://doi.org/10.1609/aaai.v35i6.16639. doi:10.1609/AAAI.V35I6.16639.