Tensor-Based Abduction in Horn Propositional Programs

Yaniv Aspis, Krysia Broda, and Alessandra Russo

Department of Computing, Imperial College London
{yaniv.aspis17,k.broda,a.russo}@imperial.ac.uk

Abstract. This paper proposes an algorithm for computing solutions of abductive Horn propositional tasks using third-order tensors. We first introduce the notion of *explanatory operator*, a single-step operation based on inverted implication, and prove that minimal abductive solutions of a given Horn propositional task can be correctly computed using this operator. We then provide a mapping of Horn propositional programs into third-order tensors, which builds upon recent work on matrix representation of Horn programs. We finally show how this mapping can be used to compute the explanatory operator by tensor multiplication.

Keywords: Abduction · Linear Algebra · High-Order Tensor

1 Introduction

Linear algebra plays a central role in both Artificial Intelligence and Machine Learning. In particular, statistical approaches to AI often involve analysis and manipulation of feature vector spaces for the purpose of learning and inference. For example, high-order tensors and dense vectors have been employed to support logic-based deductive and inductive inference [11,8], but for a limited class of logic programs. The area of symbolic AI, however, has seen only limited usage of linear algebra. Two methods have recently been proposed for embedding logic programs into vector spaces. The first, [10], allows the computation of the truth value of a First-Order formula using tensor products. The second, [9], offers a characterization of logic programs using linear algebra by mapping Horn logic programs into matrices, and disjunctive and normal programs into third-order tensors in order to support deductive inference.

Our work builds upon these recent results as a first step towards a tensor-based approach for inductive learning. Abduction plays an important part in the inference of knowledge completion, and is related to deduction and Clark Completion [1]. It has been used in several inductive logic programming approaches. For instance, XHAIL [7] uses abductive inference to compute suitable head atoms for hypothesis clauses, and both ASPAL [2] and Metagol [6] use an abductive meta-representation of the hypothesis space to search for inductive solutions.

An abductive task includes an *observation*, a *background knowledge*, in the form of logical rules, and a set of *abducible* atoms (or explanations), with which

to construct an explanation to the given observation. Explanations must be consistent with the given background knowledge and *integrity constraints* (if any). An *abductive solution* is a subset of abducibles that, together with the background knowledge, consistently explain the given observation. In this paper we consider abductive tasks where the background knowledge is represented as a set of propositional Horn clauses. We refer to such a task as an *abductive propositional Horn task*. Drawing on the work of Sakama et al. [9], we propose an algorithm for computing solutions of an abductive propositional Horn task using third-order tensors. We first introduce the notion of *explanatory operator*, a single-step operation based on inverted implication. We prove that minimal abductive solutions of an abductive propositional Horn task can be correctly computed using this operator. We then provide a mapping for expressing sets of propositional Horn clauses into third-order tensors and show how this mapping can be used to compute the explanatory operator by tensor multiplication.

The paper is structured as follows. Section 2 introduces the basic notions and terminologies. Section 3 presents our approach with theoretical results and Section 4 discusses its implementation and future research directions.

2 Background

We assume the reader is familiar with logic programming, linear algebra and the basics of high-order tensors [5]. A *Horn rule* is of the form $h \leftarrow b_1 \wedge b_2 \wedge ... \wedge b_n$, where $h, b_1, b_2, ..., b_n$ are propositional variables (called atoms) belonging to an alphabet Σ. The atom h is the *head* of r, denoted $head(r)$, and the set of atoms $\{b_1, b_2, ..., b_n\}$ is the *body* of r, denoted $body(r)$. Constants $\bot$ and $\top$ represent false and true respectively and also belong to Σ. A *fact* is a rule of the form $h \leftarrow \top$, and a *constraint* is a rule of the form $\bot \leftarrow b_1 \wedge ... \wedge b_n$. A Horn *program* P is a finite set of Horn rules, facts and constraints. The set of atoms that appear in P, along with $\bot$ and $\top$, is called the *Herbrand base* of P and denoted B_P. An *Interpretation* I of P is a subset of B_P, containing also $\top$ (implicitly). If I contains $\bot$ we say I is inconsistent. I is a *model* of P if it is consistent and satisfies the property: if $body(r) \subseteq I$ then $head(r) \in I$. We say a program P is *satisfiable*, denoted $sat(P)$ if it has a model, and if not we write $unsat(P)$. Horn programs have at most one minimal model, called the *Least Herbrand Model* ($LHM(P)$) [12]. A program $P \cup I$ is a short hand notation for adding the atoms of I as facts to a program P. The *Immediate Consequence* operator of a program P, T_P, is defined over interpretations by $T_P(I) = \{head(r) | r \in P,\ body(r) \subseteq I\}$. For every Horn Program P there exists a smallest natural number n such that $T_P^{n+1}(\emptyset) = T_P^n(\emptyset)$. $T_P^n(\emptyset)$ is called a *fixed point*. If a program is satisfiable, the fixed point is equal to $LHM(P)$ [12].

Sakama et al. have proposed in [9] a mapping of Horn programs into matrices. This is defined only for programs that satisfy a *Multiple Definitions* (MD) condition: for every two rules r_1 and r_2 such that $head(r_1) = head(r_2)$ then $|body(r_1)| \leq 1$ and $|body(r_2)| \leq 1$. In other words, if an atom is head of more than one rule, then those rules must have no more than one propositional vari-

able in their body. It has been shown that every program can be mapped into a program satisfying the MD condition [9].

We now describe a modified version of the mapping proposed in [9]. Let P be a Horn program with Herbrand base $B_P = \{\bot, \top, p_3, ..., p_N\}$. First, elements of B_P are mapped into one hot-vectors in $\mathbb{R}^N$. An interpretation I can be mapped into a vector $\boldsymbol{v^I}$ by setting $v_i = 1$ if $p_i \in I$, else $v_i = 0$. A program P is mapped into a matrix $\boldsymbol{D^P} \in \mathbb{R}^{N \times N}$ as follows. For every rule r of the form $p_i \leftarrow p_{j_1}, p_{j_2}, ..., p_{j_m}$, the elements $D^P_{ij_1} = D^P_{ij_2} = ... = D^P_{ij_m} = \frac{1}{m}$. Elements $D^P_{11} = ... = D^P_{N1} = 1$, indicating that "$\bot$ implies everything", and elements $D^P_{21} = ... = D^P_{2N} = 1$, denoting that "everything implies $\top$". Finally, differently from [9], we also set $D^P_{11} = D^P_{22} = ... = D^P_{NN} = 1$, representing the tautologies $p_i \leftarrow p_i$ for every atom p_i in P. This addition is required for the abductive procedure we propose. As an example, for the program $P = \{p \leftarrow q \wedge r, q \leftarrow t, r \leftarrow \top, t \leftarrow s, \bot \leftarrow s\}$, the corresponding matrix $\boldsymbol{D^P}$ can be seen on the right.

	$\bot$	$\top$	p	q	r	t	s
$\bot$	1	0	0	0	0	0	1
$\top$	1	1	1	1	1	1	1
p	1	0	1	$\frac{1}{2}$	$\frac{1}{2}$	0	0
q	1	0	0	1	0	1	0
r	1	1	0	0	1	0	0
t	1	0	0	0	0	1	1
s	1	0	0	0	0	0	1

Let now $H_1 : \mathbb{R} \to \{0, 1\}$ be defined as $H_1(x) = 1,$ if $x \geq 1$, else 0. Analogously to Proposition 3.2 in [9] we can prove the following proposition:

Proposition 1. *Let I and J be two interpretations. Then $J = T_P(I) \cup I$ if and only if $\boldsymbol{v^J} = H_1(\boldsymbol{D^P} \cdot \boldsymbol{v^I})$ where H_1 is applied element-wise.*

Proposition 2. *Let $T'_P(I) = T_P(I) \cup I$. Then $T'^k_P(I)$ is a fixed point for a large enough k. If $sat(P \cup I)$ then $T'^k_P(I) = LHM(P \cup I)$, otherwise $T'^k_P(I)$ is inconsistent.*

Hence, given a program P and an interpretation I, we can construct the corresponding matrix $\boldsymbol{D^P}$ and vector $\boldsymbol{v^I}$. By repeatedly multiplying by $\boldsymbol{D^P}$ and applying H_1, we can compute a vector $\boldsymbol{v}^*$ corresponding to the fixed point of $T'_P(I)$. If P is satisfiable, by Proposition 2, $\boldsymbol{v}^*$ corresponds to the $LHM(P \cup I)$. Otherwise $v^*_1 = 1$, meaning the interpretation includes $\bot$.

An *abductive task* is a triplet $< P, Ab, g >$, where program P consists of background knowledge and constraints. $Ab \subseteq B_P$ is a set of atoms called *abducibles*. g is an atom called an *observation* or *goal term*. We define an *abductive solution* to be a set $\Delta \subseteq Ab$ such that $sat(P \cup \Delta)$ and $g \in LHM(P \cup \Delta)$. A solution Δ is *minimal* if there does not exist $\Delta' \subseteq \Delta$ such that Δ' is also an abductive solution. We are interested in finding minimal solutions of an abductive task.

3 Method

We first define a single-step operator, the *explanatory operator* (E_P). This is similar in spirit to T_P, and it allows us to generate abductive solutions in a "deductive" fashion. For simplicity, we assume $Ab = B_P$. We will later consider filtering solutions according to a given specific set of abducibles. The idea behind the explanatory operator is inverting implications. For instance, suppose

P contains the rules $\{g \leftarrow q, g \leftarrow r, r \leftarrow s \wedge t\}$. Then possible (minimal) explanations for g are $\{g\}$ (since $g \in LHM(P \cup \{g\})$), $\{q\}$, $\{r\}$ and $\{s,t\}$. To arrive at these solutions, our operator takes an interpretation such as $\{g\}$ and for each rule with g as its head, creates a new interpretation consisting of the atoms in the body of that rule. So in the example above, the explanatory operator maps $\{g\}$ to the two interpretations $\{q\}$ and $\{r\}$. An interpretation I generated in this manner may be inconsistent with the program (that is, $unsat(P \cup I)$). So we remove from its output any interpretations that are inconsistent with P. The explanatory operator output includes I as well, as we would like to maintain solutions that we have already found as we repeatedly perform the operation.

Definition 1. *Let P be a program and I an interpretation such that $sat(P \cup I)$. The* explanatory operator *E_P applied to I, denoted $E_P(I)$ is given by*

$$E_P(I) = \Big(\{I\} \cup \big\{ I \backslash \{head(r)\} \cup body(r) \big| r \in P, head(r) \in I \big\} \Big) \cap \big\{ I' \big| sat(P \cup I') \big\}$$

To apply E_P iteratively, we extend its definition to $\hat{E}_P$, which can be applied to a set of interpretations, $\mathcal{I}$, all of which should be consistent with P. Then $\hat{E}_P(\mathcal{I}) = \bigcup_{I \in \mathcal{I}} E_P(I)$. We can now define the powers of $\hat{E}_P(\mathcal{I})$ by setting $\hat{E}^0_P(\mathcal{I}) = \mathcal{I}$ and $\hat{E}^{n+1}_P(\mathcal{I}) = \hat{E}_P(\hat{E}^n_P(\mathcal{I}))$. For the program above, $\hat{E}^0_P(\{\{g\}\}) = \{\{g\}\}$, $\hat{E}^1_P(\{\{g\}\}) = \{\{g\}, \{q\}, \{r\}\}$, $\hat{E}^2_P(\{\{g\}\}) = \{\{g\}, \{q\}, \{r\}, \{s,t\}\}$ and $\hat{E}^3_P(\{\{g\}\}) = \hat{E}^2_P(\{\{g\}\})$. $\hat{E}^2_P(\{\{g\}\})$ is therefore a fixed point. The next proposition ensures that a fixed point will always be reached.

Proposition 3. *Let $\mathcal{I}$ be a set of interpretations all consistent with a program P. Then there exists a smallest natural number n for which $\hat{E}^{n+1}_P(\mathcal{I}) = \hat{E}^n_P(\mathcal{I})$.*

Proof. Note that, by the definition of E_P, for every $I \in \mathcal{I}$ we have $I \in E_P(I)$, meaning $\mathcal{I} \subseteq \hat{E}_P(\mathcal{I})$. By induction it follows that $\hat{E}^n_P(\mathcal{I}) \subseteq \hat{E}^{n+1}_P(\mathcal{I})$. Hence $|\hat{E}^n_P(\mathcal{I})|$ is monotonically increasing in n. But for a finite program P, the Herbrand base B_P is finite and therefore $|\hat{E}^n_P(\mathcal{I})|$ is bounded from above by $2^{|B_P|}$. Therefore, there exists a smallest n for which $\hat{E}^n_P(\mathcal{I}) = \hat{E}^{n+1}_P(\mathcal{I})$. □

We denote the fixed point by $\hat{E}^*_P(\{\{g\}\})$. A generic abductive algorithm would construct $\hat{E}^*_P(\{\{g\}\})$ by iteratively computing $\hat{E}_P$ over interpretations. This is connected to the relationship between abduction and Clark Completion [1].

Proposition 4. *Suppose $\Delta \in \hat{E}^*_P(\{\{g\}\})$. Then $sat(P \cup \Delta)$ and $g \in LHM(P \cup \Delta)$ and Δ is an abductive solution.*

Proposition 4 shows that $\hat{E}^*_P(\{\{g\}\})$ contains only abductive solutions. Although it does not in general find all solutions, it does contain all the minimal ones. To show this, we first define a more general version of minimality.

Definition 2. *Given an abductive task $< P, Ab, g >$, a solution Δ is n-step minimal if $g \in T^n_{P \cup \Delta}(\emptyset)$ and for every proper subset $\Delta' \subsetneq \Delta$ we have $g \notin T^n_{P \cup \Delta'}(\emptyset)$.*

Intuitively, Δ is n-step minimal if g can be derived from $P \cup \Delta$ in *exactly* n steps, and every atom in Δ is necessary to do so. For example, consider the program $P = \{g \leftarrow p \wedge q, p \leftarrow t, p \leftarrow q\}$. We have $\hat{E}_P^*(\{\{g\}\}) = \{\{g\}, \{p, q\}, \{t, q\}, \{q\}\}$. $\{p, q\}$ is a solution that is not minimal, in fact it is 2-step minimal. $\{q\}$ is a minimal solution that is not 2-step minimal, but is 3-step minimal. $\{q, t\}$ is another solution that is not minimal, nor is it n-step minimal for any n. It is easy to see that all minimal solutions are n-step minimal for some natural number n.

Proposition 5. *Suppose that for the abductive task* $< P, B_P, g >$ *we have that* Δ *is an* n*-step minimal solution for some natural number* n*. Then* $\Delta \in \hat{E}_P^*(\{\{g\}\})$.

We now move from a generic algorithm to a linear algebraic one. First, we construct a third-order tensor which realises the $\hat{E}_P$ operator. Interpretations, as before, can be represented as vectors. To represent a set of interpretations, we simply stack these vectors as columns of a matrix. Using 2-mode multiplication [5] on this matrix with the tensor realises application of the $\hat{E}_P$ operator. This operation, however, does not produce quite the desired result. Firstly, there may be duplicate columns, which can simply be removed. More importantly, some of the columns will represent interpretations inconsistent with P. In essence, tensor multiplication will realise the following part of the E_P operation: $\{I\} \cup \left\{I \backslash \{head(r)\} \cup body(r) \middle| r \in P, head(r) \in I\right\}$. Luckily, we have a simple way to remove inconsistent solutions: We can compute the fixed point for each column of the matrix using matrix multiplication as in Section 2. Filtering these out, we get exactly the behaviour of E_P and achieve linear algebraic abduction.

In more detail, let P be a program with $K-1$ non-constraint rules such that B_P has size N. We map a program P to an *Abductive Tensor* $A^P \in \mathbb{R}^{N \times N \times K}$. The first $K-1$ frontal slices of A^P correspond to its (inverted) rules. If the $k-th$ rule has $head(r) = p_j$ and $body(r) = \{p_{i_1}, p_{i_2}, ..., p_{i_m}\}$ then we set $A^P_{i_1 jk} = A^P_{i_2 jk} = ... = A^P_{i_m jk} = 1$. This ensures that if p_j is "turned on" in an interpretation, then multiplication with this frontal slice will "turn on" the atoms in the body of the rule. We also set $A^P_{11k} = A^P_{22k} = ... = A^P_{i-1,i-1,k} = A^P_{i+1,i+1,k} = ... = A^P_{N,N,k} = 1$, with all other elements set to 0 (note that $A^P_{i,i,k} = 0$). This will "turn off" p_j while leaving all other atoms intact. We do this for all $K-1$ rules. Finally, the last slice is equal to the identity matrix $\boldsymbol{A^P_{::K}} = \mathbb{I}$, and corresponds to keeping the current interpretation. As an example, consider the program $P = \{g \leftarrow q \wedge r, q \leftarrow t, \bot \leftarrow t\}$. The frontal slices of the corresponding tensor are:

$$\boldsymbol{A^P_{::1}} = \begin{array}{c} \\ \bot \\ \top \\ g \\ q \\ r \\ t \end{array} \begin{array}{c} \begin{array}{cccccc} \bot & \top & g & q & r & t \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \quad \boldsymbol{A^P_{::2}} = \begin{array}{c} \\ \bot \\ \top \\ g \\ q \\ r \\ t \end{array} \begin{array}{c} \begin{array}{cccccc} \bot & \top & g & q & r & t \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \end{array} \quad \boldsymbol{A^P_{::3}} = \begin{array}{c} \\ \bot \\ \top \\ g \\ q \\ r \\ t \end{array} \begin{array}{c} \begin{array}{cccccc} \bot & \top & g & q & r & t \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Our generic algorithm can then be realised as follows: Given a program P and goal g, construct the tensor A^P and matrix $\boldsymbol{D^P}$. For a set of interpretations $\mathcal{I}$ construct the corresponding matrix $\boldsymbol{U^{\mathcal{I}}}$ and then compute $\hat{E}_P(\mathcal{I})$ as follows. Let $\boldsymbol{W} = H_1(A^P \times_2 \boldsymbol{U^{\mathcal{I}}})$, where duplicate columns are removed from $\boldsymbol{W}$. Next compute the fixed point for each column of $\boldsymbol{W}$, using $\boldsymbol{D^P}$ and the same method outlined in Section 2. The first element of each column corresponds to $\bot$ and if set to 1 in the fixed point it indicates the column can be removed since it represents an inconsistent interpretation. If we begin with $\mathcal{I} = \{\{g\}\}$ and perform the above steps iteratively, we compute $\hat{E}^*_P(\{\{g\}\})$. We only need to ensure $\{g\}$ is consistent with P. Each column of the final matrix corresponds to an abductive solution, with all minimal solutions assured to be included. For example, for the above program the abductive process is as follows:

$$H_1\big(A^P \times_2 \begin{bmatrix}0\\1\\1\\0\\0\\0\end{bmatrix}\big) = \begin{bmatrix}0&0&0\\1&1&1\\0&1&1\\1&0&0\\1&0&0\\0&0&0\end{bmatrix} \quad H_1\big(A^P \times_2 \begin{bmatrix}0&0\\1&1\\0&1\\1&0\\1&0\\0&0\end{bmatrix}\big) = \begin{bmatrix}0&0&0&0&0&0\\1&1&1&1&1&1\\0&0&0&0&1&1\\1&0&1&1&0&0\\1&1&1&1&0&0\\0&1&0&0&0&0\end{bmatrix} \Rightarrow \boldsymbol{W} = \begin{bmatrix}0&0\\1&1\\0&1\\1&0\\1&0\\0&0\end{bmatrix}$$

The matrix $\boldsymbol{W}$ is the fixed point, its columns corresponding to $\{q, r\}$ and $\{g\}$, while the column corresponding to $\{r, t\}$ has been removed as it is inconsistent with the program. The next proposition ensures the correctness of the tensor representation.

Proposition 6. *Let I be an interpretation consistent with a program P. Denote $\mathcal{J} = \{I\} \cup \big\{I \backslash \{head(r)\} \cup body(r) \big| r \in P, head(r) \in I\big\}$. Let $\boldsymbol{U} = H_1(A^P \times_2 \boldsymbol{v^I})$. Then the columns of $\boldsymbol{U}$ correspond to the interpretations in $\mathcal{J}$.*

Finally, we consider the case where we have a general set of abducibles $Ab \subseteq B_P$. After computing $\hat{E}^*_P(\{\{g\}\})$ we can filter out solutions that are not a subset of Ab. Note that $\Delta \subseteq Ab$ if and only if $\boldsymbol{v^{\Delta}} \times \boldsymbol{v^{Ab}} = \boldsymbol{v^{\Delta}}$ where $\times$ means element-wise multiplication. This is one such way to achieve post-filtering of solutions.

Theorem 1. *Given an abductive task $< P, Ab, g >$, the algorithm above produces a subset of the abductive solutions that contains all minimal solutions.*

Proof. Propositions 1 and 6 ensure that linear algebraic operations allow the computation of $\hat{E}^*_P(\{\{g\}\})$. Proposition 3 ensures the algorithm always halts and correctness can be seen from Propositions 4 and 5.

4 Discussion

We have introduced a linear algebraic approach to abduction in propositional Horn programs via an explanatory operator and outline the correctness of the approach. We have also provided a characterization of the operator by use of third-order tensors.

A straight-forward unoptimised implementation has been constructed and here we consider two possible optimisations: First, to prevent repeated computations, we can maintain a list of interpretations that have already been multiplied by the tensor. At each step of the algorithm we can remove columns from the interpretation matrix $\boldsymbol{U}^{\mathcal{I}}$ if they already appear in this list. Second, our tensor may be broken down to a sequence of tensors $A_1^P, A_2^P, ..., A_n^P$, each embedding rules that share the same head. Multiplying $\boldsymbol{U}^{\mathcal{I}}$ by each tensor in sequence will produce the same result but can create fewer duplicate columns. While finding abductive solutions in Horn propositional programs is NP-complete [3], our bottom-up approach together with the suggested optimisations can eliminate many unnecessary computational steps.

In the future, we plan to investigate extending our method to first-order logic, for instance by combining it with Sato's tensor representation of Tarskian Semantics [10], initially starting with unary predicates. A second extension is to allow for normal logic programs P and exploit the connection between negation-as-failure and abduction [4] to compute models under stable semantics.

References

1. Console, U., Dupr, D., Torasso, P.: On the relationship between abduction and deduction. Journal of Logic and Computation **1**(5), 661–690 (1991). https://doi.org/10.1093/logcom/1.5.661
2. Corapi, D., Russo, A., Lupu, E.: Inductive Logic Programming in Answer Set Programming. In: Muggleton, S.H., Others (eds.) Inductive Logic Programming. pp. 91–97. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
3. Eiter, T., Gottlob, G.: The complexity of logic-based abduction. J. ACM **42**(1), 3–42 (Jan 1995). https://doi.org/10.1145/200836.200838
4. Eshghi, K., Kowalski, R.: Abduction Compared with Negation by Failure. pp. 234–254 (01 1989)
5. Kolda, T.G., Bader, B.W.: Tensor Decompositions and Applications. SIAM REVIEW **51**(3), 455–500 (2009)
6. Muggleton, S.H., Lin, D., Pahlavi, N., Tamaddoni-Nezhad, A.: Meta-interpretive learning: application to grammatical inference. Machine Learning **94**(1), 25–49 (Jan 2014). https://doi.org/10.1007/s10994-013-5358-3
7. Ray, O.: Nonmonotonic Abductive Inductive Learning. Journal of Applied Logic **7**(3), 329–340 (2009). https://doi.org/10.1016/j.jal.2008.10.007, special Issue: Abduction and Induction in Artificial Intelligence
8. Rocktäschel, T., Riedel, S.: End-to-end differentiable proving. In: Guyon, I., Others (eds.) Advances in Neural Information Processing Systems 30, pp. 3788–3800. Curran Associates, Inc. (2017)
9. Sakama, C., Inoue, K., Sato, T.: Linear Algebraic Characterization of Logic Programs. In: KSEM. Lecture Notes in Computer Science, vol. 10412, pp. 520–533. Springer (2017), `https://doi.org/10.1007/978-3-319-63558-3_44`
10. Sato, T.: Embedding Tarskian Semantics in Vector Spaces (Mar 2017), `http://arxiv.org/abs/1703.03193v1`
11. Serafini, L., Garcez, A.S.d.: Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge. CoRR **abs/1606.04422** (2016)
12. van Emden, M.H., Kowalski, R.A.: The Semantics of Predicate Logic as a Programming Language. Journal ACM **23**(4), 733–742 (Oct 1976)