Learning to Classify Logical Formulas based on their Semantic Similarity

Ali Ballout[1], Célia da Costa Pereira[2], and Andrea G. B. Tettamanzi[1]

[1] Université Côte d'Azur, CNRS, Inria, I3S, France
ali.ballout@inria.fr, andrea.tettamanzi@univ-cotedazur.fr
[2] Université Côte d'Azur, CNRS, I3S, France
Celia.DA-COSTA-PEREIRA@univ-cotedazur.fr

Abstract. An important task in logic, given a formula and a knowledge base which represents what an agent knows of the current state of the world, is to be able to guess the truth value of the formula. Logic reasoners are designed to perform inferences, that is, to decide whether a formula is a logical consequence of the knowledge base, which is stronger than that and can be intractable in some cases. In addition, under the open-world assumption, it may turn out impossible to infer a formula or its negation. In many practical situations, however, when an agent has to make a decision, it is acceptable to resort to heuristic methods to determine the probable veracity or falsehood of a formula, even in the absence of a guarantee of correctness, to avoid blocking the decision-making process and move forward. This is why we propose a method to train a classification model based on available knowledge in order to be able of accurately guessing whether an arbitrary, unseen formula is true or false. Our method exploits a kernel representation of logical formulas based on a model-theoretic measure of semantic similarity. The results of experiments show that the proposed method is highly effective and accurate.

1 Introduction and Related Work

A defining feature for intelligent agents is their ability to reason, that is to draw logical conclusions from the available premises, which constitute their knowledge [3, 5]. While this capability is very important, an equally important and useful, but weaker, capability for an agent would be to be able to recognize if a given formula is likely to be true or false, given the current knowledge, *in the current state of the world*, even though not necessarily in general.

As a matter of fact, we often experience situations where our incomplete knowledge would not allow us to make exact inferences, yet this does not prevent us to make decisions, because even when we don't know a fact that we need in order to move forward, we are able to make an educated guess (i.e., a prediction based on what we already know) about the veracity of that fact and proceed with our decision making.

It is this weaker task that we are interested in studying here. We propose a simple but effective idea, which is to train a classifier against the knowledge

base of the agent, which may be viewed as a set of formulas labeled with their truth value. The formulas are represented as vectors of similarities to the labeled formulas; to this end, we propose a model-theoretic semantic similarity measure which can be computed efficiently. This kernel-representation and its associated similarity measure are the key ingredients of our proposal.

Recently, a rise of interest in developing connectionist methods for reasoning can be observed, with proposals such the so-called Logic Tensor Networks [4], or the Logic-Integrated Neural Network [14] to integrate the power of deep learning and logic reasoning, or approaches that employ state-of-the-art methods for training deep neural networks to learn to perform some basic ontology reasoning tasks [11]. As a further witness of the attention this research field is attracting, some conferences are beginning to feature tutorials on it, like KDD'21 [15] and there is even an upcoming Dagstuhl seminar on "Machine Learning and Logical Reasoning: The New Frontier".[3]

Unlike these approaches, what we propose does not require sophisticated neural architectures or resource-intensive deep learning; in addition, we do not attack the more ambitious challenge of logical deduction, but just that of heuristically guessing (as human beings do), the truth value of a formula, independently of its being logically entailed by the available knowledge.

Actually, what people do in case of incomplete knowledge is to somehow measure the similarity between known/familiar situations and unknown/unfamiliar situations [16]. Several cognitive tasks, such as *learning* and *interpolation* require the concept of similarity to be performed [9]. There exists a vast literature on similarity measures, with many proposal arising in the field of machine learning [7]. However, it appears that the problem of measuring the similarity of logical formulas has been less investigated and, when it has, that's often in relation with specific contexts.

While not directly addressing the problem of defining similarity among logical formulas, Bowles [6] studies the nature of relevance and irrelevance of a proposition with respect to another. His work shares with the definition of semantic similarity that we propose here, a basic intuition, which is that the probability that one proposition is true given that another one is true should play a central role. The definition of relevance proposed by Makinson in [13], instead, is not in line with what we are proposing here because it is defined in terms of letter-sharing. A way of measuring the similarity of a Boolean vector to a given set of Boolean vectors, motivated in part by certain data mining or machine learning problems, was proposed by Anthony and Hammer [2]. A similarity measure for Boolean function was proposed by Fišer *et al.* [10] in a quite different context, that of circuit synthesis, which explains the differences with our proposal. One measure of similarity between functions is the existence of a Lipschitz mapping (with small constant) between them [12]. A problem somehow related to the one we are dealing with is the problem of measuring the similarity between logical arguments, which has been studied by Amgoud and David [1].

[3] Dagstuhl Seminar 22291, July 17–22, 2022.

Through an empirical validation we show that the framework we propose allows a number of quite standard and unsophisticated classification techniques, like support-vector machines, to learn very accurate models that are capable of "guessing" whether a given, unseen formula is true or false in the current state of affairs, without the need to perform any logical deduction.

The rest of the paper is structured as follows: Section 2 states the problem of formula classification; Section 3 defines a semantic similarity measure for formulas that is the cornerstone of the proposed approach. Section 4 provides an empirical validation of the approach and Section 5 draws some conclusions and suggestions for future work.

2 Problem Statement

Let Φ be a set of formulas in a logical language $\mathcal{L}$ and let $\mathcal{I}$ be an interpretation, which represents a particular state of affairs or the current state of the world. Under interpretation $\mathcal{I}$, the formulas in Φ may be labeled as being true or false. One could thus construct a table

$$\begin{bmatrix} \phi_1, \phi_1^{\mathcal{I}} \\ \phi_2, \phi_2^{\mathcal{I}} \\ \vdots \quad \vdots \end{bmatrix},$$

where $\phi_i \in \Phi \subset \mathcal{L}$, for $i = 1, 2, \ldots$, and $\phi_i^{\mathcal{I}}$ is the truth value of ϕ_i according to interpretation $\mathcal{I}$. This table can be viewed as a representation of a knowledge base K consisting of all the formulas $\phi_i \in \Phi$ such that $\phi_i^{\mathcal{I}} = T$ and all the formulas $\neg\phi_i \in \Phi$ such that $\phi_i^{\mathcal{I}} = F$. K represents what an agent knows (or believes) about the current state of affairs but, of course, $\mathcal{I}$, the actual state of affairs, is not known in full, which is like saying that the open world hypothesis holds.

Consider now the problem of guessing or predicting whether a new formula $\psi \notin \Phi$ is true or false in $\mathcal{I}$, given K. To be sure, one could use a reasoner to check whether $K \vdash \psi$ or $K \vdash \neg\psi$. If the reasoner is sound and complete, this can even allow one to decide whether $K \models \psi$ or $K \models \neg\psi$. However, even in cases where $K \not\models \psi$ and $K \not\models \neg\psi$, which are entirely possible in an open world, it would be useful for an agent to be able to make educated guesses at the truth value of ψ. By an *educated guess* we mean a prediction, based on the truth values of the formulas the agent already knows. If a model to make that type of predictions existed and were fast and accurate enough, the agent might even used it *instead* of the reasoner, for time-critical tasks where having a quick answer is more important than having an answer that is guaranteed to be always correct.

What we have just described is a classification problem, where given a set of labeled examples (here, formulas with their truth value), a model is sought for that is able to accurately predict the label of an unseen case (i.e., a new formula).

To solve this problem, we propose to use a kernel representation, i.e., to represent formulas **as vectors of similarities to a restricted set of formulas whose label is already known** (Φ) and to train a classification model on these labeled examples, later to be used to classify new, unseen formulas. To this aim, we will stick to very standard and unsophisticated classification methods.

3 Semantic Similarity

We need to define similarity among logical formulas. It is quite obvious that such a similarity should not be based on syntax, due to the fact that formulas with widely different syntactical forms may be equivalent. Now, the semantics of logical formulas is defined in model-theoretic terms. What we are looking for is, therefore, a model-theoretic notion of formula similarity.

To keep technical complications at a minimum and without loss of generality, let us consider propositional logic. As a matter of fact, more expressive logical languages can be mapped to the propositional case (e.g., description logics and first-order logic under the Herbrand semantics).

Definition 1 (Language) *Let $\mathcal{A}$ be a* finite *set of atomic propositions and let $\mathcal{L}$ be the propositional language such that $\mathcal{A} \cup \{\top, \bot\} \subseteq \mathcal{L}$, and, $\forall \phi, \psi \in \mathcal{L}$, $\neg\phi \in \mathcal{L}$, $\phi \wedge \psi \in \mathcal{L}$, $\phi \vee \psi \in \mathcal{L}$.*

Additional connectives can be defined as useful shorthands for combination of connectives of $\mathcal{L}$, e.g., $\phi \supset \psi \equiv \neg\phi \vee \psi$.

We will denote by $\Omega = \{0,1\}^{\mathcal{A}}$ the set of all interpretations on $\mathcal{A}$, which we may also call the "universe". An interpretation $\mathcal{I} \in \Omega$ is a function $\mathcal{I} : \mathcal{A} \to \{0,1\}$ assigning a truth value $p^{\mathcal{I}}$ to every atomic proposition $p \in \mathcal{A}$ and, by extension, a truth value $\phi^{\mathcal{I}}$ to all formulas $\phi \in \mathcal{L}$; $\mathcal{I} \models \phi$ means that $\phi^{\mathcal{I}} = 1$ ($\mathcal{I}$ is a model of ϕ); if $S \subseteq \mathcal{L}$ is a set of formulas, $\mathcal{I} \models S$ means $\mathcal{I} \models \phi$ for all $\phi \in S$; $S \models \phi$ means that $\forall \mathcal{I} \models S$, $\mathcal{I} \models \phi$. The notation $[\phi]$ denotes the set of all models of formula $\phi \in \mathcal{L}$: $[\phi] = \{\mathcal{I} \in \Omega : \mathcal{I} \models \phi\}$. The semantics of a formula $\phi \in \mathcal{L}$ is the set of its models, $[\phi]$.

We might begin by defining the semantic distance between two formulas ϕ and ψ as the Hamming distance between the two binary string that represent their respective sets of models:

$$d(\phi, \psi) = \sum_{\mathcal{I} \in \Omega} [\phi^{\mathcal{I}} \neq \psi^{\mathcal{I}}], \tag{1}$$

where [expr] denotes the indicator function, which equals 1 if expr is true and 0 otherwise.

According to this definition, $d(\phi, \neg\phi) = \|\Omega\|$ and $d(\phi, \phi) = 0$, which is in good agreement with our intuition. Also, two formulas that are totally unrelated,[4] like,

[4] Two formulas may be said to be totally unrelated if knowing the truth value of one does not give any information about the truth value of the other.

say, p and q, where $p, q \in \mathcal{A}$, will have a distance which is half-way in between these two extreme cases, $d(p,q) = \frac{1}{2}\|\Omega\|$.

One problem with this notion of distance is that the distance between two given formulas depends on the number of propositional constants in the language, which is a little counter-intuitive. For instance, $d(p,q) = 2$ if $\mathcal{A} = \{p,q\}$, but $d(p,q) = 4$ if $\mathcal{A} = \{p,q,r\}$, and so on. In addition, to compute it, we have to consider all interpretations in Ω, even though many of them might be indifferent when it comes to two given formulas: for example, pqr and $pq\bar{r}$ are indifferent when comparing p to q.

The former problem disappears if, instead of a distance, we define a similarity, ranging between 0 and 1 based on the same idea, as follows:

$$\mathrm{sim}(\phi,\psi) = \frac{1}{\|\Omega\|}\sum_{\mathcal{I}\in\Omega}[\phi^{\mathcal{I}} = \psi^{\mathcal{I}}]. \tag{2}$$

The latter problem is also solved by defining $\mathcal{A}_\phi \subseteq \mathcal{A}$ as the set of atoms that occur in formula ϕ and by letting $\Omega_{\phi,\psi} = 2^{\mathcal{A}_\phi \cup \mathcal{A}_\psi}$; Equation 2 can now be rewritten as

$$\mathrm{sim}(\phi,\psi) = \frac{1}{\|\Omega\|}\sum_{\mathcal{I}\in\Omega}[\phi^{\mathcal{I}} = \psi^{\mathcal{I}}] = \frac{1}{\|\Omega_{\phi,\psi}\|}\sum_{\mathcal{I}\in\Omega_{\phi,\psi}}[\phi^{\mathcal{I}} = \psi^{\mathcal{I}}]. \tag{3}$$

According to this definition, for all formula $\phi \in \mathcal{L}$, $\mathrm{sim}(\phi,\phi) = 1$, $\mathrm{sim}(\phi,\neg\phi) = 0$ and, no matter how many atoms are involved, if $\mathcal{A}_\phi \cap \mathcal{A}_\psi = \emptyset$, $\mathrm{sim}(\phi,\psi) = \frac{1}{2}$.

Another interesting property of this semantic similarity is the following, which ensures that the proposed similarity is consistent with logical negation.

Theorem 1. *Let ϕ ψ be any two formulas of $\mathcal{L}$. Then*

$$\mathrm{sim}(\phi,\psi) = 1 - \mathrm{sim}(\neg\phi,\psi).$$

Proof. For all interpretation $\mathcal{I}$, $\phi^{\mathcal{I}} = \psi^{\mathcal{I}} \Leftrightarrow \neg\phi^{\mathcal{I}} \neq \psi^{\mathcal{I}}$ and $\phi^{\mathcal{I}} \neq \psi^{\mathcal{I}} \Leftrightarrow \neg\phi^{\mathcal{I}} = \psi^{\mathcal{I}}$. Therefore, $\{\mathcal{I} : \phi^{\mathcal{I}} = \psi^{\mathcal{I}}\} = \{\mathcal{I} : \neg\phi^{\mathcal{I}} \neq \psi^{\mathcal{I}}\} = \Omega \setminus \{\mathcal{I} : \neg\phi^{\mathcal{I}} = \psi^{\mathcal{I}}\}$ and we can thus write

$$\begin{aligned}
\mathrm{sim}(\phi,\psi) &= \frac{1}{\|\Omega\|}\sum_{\mathcal{I}\in\Omega}[\phi^{\mathcal{I}} = \psi^{\mathcal{I}}] = \frac{1}{\|\Omega\|}\|\{\mathcal{I} : \phi^{\mathcal{I}} = \psi^{\mathcal{I}}\}\| \\
&= \frac{1}{\|\Omega\|}\|\Omega \setminus \{\mathcal{I} : \neg\phi^{\mathcal{I}} = \psi^{\mathcal{I}}\}\| = \frac{\|\Omega\|}{\|\Omega\|} - \frac{1}{\|\Omega\|}\|\{\mathcal{I} : \neg\phi^{\mathcal{I}} = \psi^{\mathcal{I}}\}\| \\
&= 1 - \frac{1}{\|\Omega\|}\sum_{\mathcal{I}\in\Omega}[\neg\phi^{\mathcal{I}} = \psi^{\mathcal{I}}] = 1 - \mathrm{sim}(\neg\phi,\psi).
\end{aligned}$$

□

Another interesting property of the semantic similarity, as we have defined it, is that, if $\Omega_{\phi,\psi}$ is too large, we are not obliged to perform an exact computation of $\mathrm{sim}(\phi,\psi)$, but we can approximate it with acceptable accuracy by randomly sampling n interpretations from $\Omega_{\phi,\psi}$ and counting for how many of

them $\phi^{\mathcal{I}} = \psi^{\mathcal{I}}$. Indeed, $\mathrm{sim}(\phi, \psi)$ may be construed as a probability, namely the probability that, in a random interpretation, ϕ and ψ are both true or both false. What we get is an unbiased estimator of $\mathrm{sim}(\phi, \psi)$, which behaves like a binomial parameter $\hat{s}_{\phi,\psi}$, whose confidence interval is given by the Wald confidence interval, based on the asymptotic normality of $\hat{s}_{\phi,\psi}$ and estimating the standard error. This $(1-\alpha)$ confidence interval for $\mathrm{sim}(\phi, \psi)$ would be

$$\hat{s}_{\phi,\psi} \pm z_{\alpha/2}\sqrt{\hat{s}_{\phi,\psi}(1-\hat{s}_{\phi,\psi})/n}, \tag{4}$$

where z_c denotes the $1-c$ quantile of the standard normal distribution.

For example, if we set $n = 30$, with a 99% confidence, the actual similarity will be within a deviation of $2.576\sqrt{1/120} = 0.2351$ from $\hat{s}_{\phi,\psi}$, in the worst case, which corresponds to $\hat{s}_{\phi,\psi} = 0.5$; for $n = 100$, the approximation error will be less than 0.1288 and for $n = 1000$ it will be less than 0.0407. As a matter of fact, a precise computation of the similarity between formulas is not really required for the proposed approach to work.

This also suggests a way to deal with non-finite interpretations, which might arise in expressive languages involving variables and functions.

4 Experiments and Results

4.1 An Example from the Block World

As a first test and example of our proposal, we define a language with four individual constants, A, B, C, *Table*, one unary predicate, $\mathsf{covered}(\cdot)$, and one binary predicate $\mathsf{on}(\cdot,\cdot)$. The Herbrand base of this language is finite and consists of twenty ground atoms, but we can only consider a subset of it, after dropping atoms like $\mathsf{on}(A, A)$, $\mathsf{covered}(\mathit{Table})$, and $\mathsf{on}(\mathit{Table}, A)$, which would always be false in every state of the block world:

$$\begin{aligned}\mathcal{A}_{12} = \{\ &\mathsf{covered}(A),\ \mathsf{on}(A,B),\ \mathsf{on}(A,C),\ \mathsf{on}(A,\mathit{Table}),\\ &\mathsf{covered}(B),\ \mathsf{on}(B,A),\ \mathsf{on}(B,C),\ \mathsf{on}(B,\mathit{Table}),\\ &\mathsf{covered}(C),\ \mathsf{on}(C,A),\ \mathsf{on}(C,B),\ \mathsf{on}(C,\mathit{Table})\ \ \}.\end{aligned}$$

Notice that, given this $\mathcal{A}_{12}$, $\|\Omega_{12}\| = 2^{12} = 4,096$. By adding another block D to this world we can obtain a larger set of atoms

$$\begin{aligned}\mathcal{A}_{20} = \{\ &\mathsf{covered}(A),\ \mathsf{on}(A,B),\ \mathsf{on}(A,C),\ \mathsf{on}(A,D),\ \mathsf{on}(A,\mathit{Table}),\\ &\mathsf{covered}(B),\ \mathsf{on}(B,A),\ \mathsf{on}(B,C),\ \mathsf{on}(B,D),\ \mathsf{on}(B,\mathit{Table}),\\ &\mathsf{covered}(C),\ \mathsf{on}(C,A),\ \mathsf{on}(C,B),\ \mathsf{on}(C,D),\ \mathsf{on}(C,\mathit{Table}),\\ &\mathsf{covered}(D),\ \mathsf{on}(D,A),\ \mathsf{on}(D,B),\ \mathsf{on}(D,C),\ \mathsf{on}(D,\mathit{Table})\ \ \},\end{aligned}$$

of size 20, with $\|\Omega_{20}\| = 2^{20} = 1,048,576$, and, similarly, by adding a further block E, a set $\mathcal{A}_{30}$ of 30 atoms, with $\|\Omega_{30}\| = 2^{30} = 1,073,741,824$.

The language may then be completed by a minimal set of logical operators, $\neg$, $\wedge$, $\vee$. Then we select a reference interpretation, for example

$$\mathcal{I}^*_{12} = \{\mathsf{on}(A,\mathit{Table}), \mathsf{on}(C,\mathit{Table}), \mathsf{on}(B,A), \mathsf{covered}(A)\},$$

corresponding to a given state of a very simple block world containing one table and three blocks, arranged as in Figure 1a.

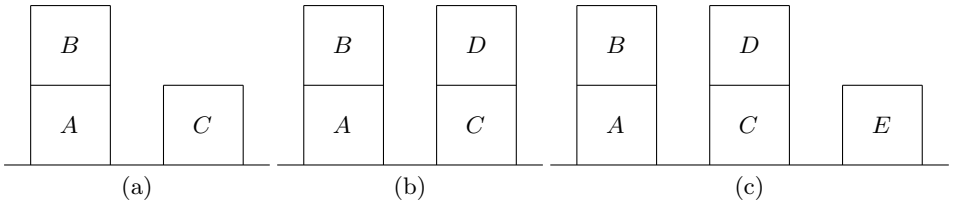


Fig. 1: The three block worlds corresponding to the reference interpretations, respectively, (a) to $\mathcal{I}^*_{12}$, (b) to $\mathcal{I}^*_{20}$, and (c) to $\mathcal{I}^*_{30}$.

We then generate a set Φ of random logical formulas and we assign them a truth label based on $\mathcal{I}^*$ and train various models on it.

4.2 Experimental Protocol

The experiment was divided into two parts. In the first part, we created 3 datasets, each of which is based on a different universe generated using the language explained in Section 4.1. The universes are depicted in Figure 1. We used these sets to test the performance of models learned using our proposed similarity measure. We considered different universe complexities and used very small training sets to simulate a realistic scenario. For this part, no sampling was done, all interpretations in $\Omega_{\phi,\psi}$ were considered as per Equation 3. This can be very time-consuming when the two formulas involve many atoms. For the second part, we created 3 additional sets for each of the universes used in the first part. These additional sets contain the exact same formulas as those in the first part, the only difference being the way the similarity was calculated. To investigate what we mentioned in Section 3 regarding the ability to approximate the similarity with acceptable accuracy by randomly sampling n interpretations, we approximated the similarities for each of the 3 additional sets using $n = 30$, $n = 100$, and $n = 1000$ respectively. We then compared the performance of each of these sets with the base one created in the first part.[5]

Part One: Baseline Experiment. To see how the proposed method performs, we created the 3 universes depicted in Figure 1, and denoted by Ω_{12}, Ω_{20}, and Ω_{30}. The universes consist of 12, 20, and 30 atoms respectively (sets $\mathcal{A}_{12}$, $\mathcal{A}_{20}$, and $\mathcal{A}_{30}$ as defined above). The following are the reference interpretations:

[5] All the code and data used for the experiments described in this paper can be found in the following repository: `https://github.com/ali-ballout/Learning-to-Classify-Logical-Formulas-based-on-their-Semantic-Similarity`.

– $\mathcal{I}^*_{12} = \{\mathsf{on}(A, \mathit{Table}), \mathsf{on}(C, \mathit{Table}), \mathsf{on}(B, A), \mathsf{covered}(A)\}$, cf. Figure 1a;
– $\mathcal{I}^*_{20} = \{\mathsf{on}(A, \mathit{Table}), \mathsf{on}(C, \mathit{Table}), \mathsf{on}(B, A), \mathsf{on}(D, C), \mathsf{covered}(A), \mathsf{covered}(C)\}$, cf. Figure 1b;
– $\mathcal{I}^*_{30} = \{\mathsf{on}(A, \mathit{Table}), \mathsf{on}(C, \mathit{Table}), \mathsf{on}(E, \mathit{Table}), \mathsf{on}(B, A), \mathsf{on}(D, C), \mathsf{covered}(A), \mathsf{covered}(C)\}$, cf. Figure 1c;

Following that, we generated 500 random formulas for each of the universes using Algorithm 1. Algorithm 1 generates a given number N of random formulas, taking as input a list of ground atoms $\mathcal{A}$. It is recursive and chooses its next step and which symbols to add at random. It uses a variable that reduces the probability of adding a nested subformula the more complex a formula becomes.

We then labeled each of the formulas with its truth, based on the reference interpretation of its universe. The next step was to create the similarity matrix for each set of formulas. For this part, no sampling was done, in other words all interpretations were taken into account and no noise was added. The similarity between each formula and all other formulas in the set is calculated using Equation 3. To simplify, we compare formula ϕ to all other formulas in the set of 500 formulas. At each comparison we check all the unique atoms included in the compared formulas ϕ and ψ, for an example of what atoms are refer to $\mathcal{A}_{12}$ in Section 4.1. We then generate all interpretations for this set of unique atoms extracted from both formulas. Then we record the truth for each of the formulas based on each of the generated interpretations. After that, we count the instances where the truth of formulas ϕ and ψ are the same. We divide that number by the total number of generated interpretations and the result obtained is the similarity between ϕ and ψ. We do this, once, for all pairs of formulas to obtain a symmetric similarity matrix of the shape 500×500. Figure 2 depicts the

$Truth$	$Formula$	ϕ_0	ϕ_1	$\dots$	ϕ_m
$Truth_0$	ϕ_0	1	$S_{0,1}$	$\dots$	$S_{0,m}$
$Truth_1$	ϕ_1	$S_{1,0}$	1	$\dots$	$S_{1,m}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$Truth_{m-1}$	ϕ_{m-1}	$S_{m-1,0}$	$S_{m-1,1}$	$\dots$	$S_{m-1,m}$
$Truth_m$	ϕ_m	$S_{m,0}$	$S_{m,1}$	$\dots$	1

Fig. 2: Formula similarity matrix with truth labels.

similarity matrix between formulas of a set of formulas of size m with S being the similarity between each pair. The diagonal is all 1 since it is the similarity between a formula ϕ and itself. The truth labels of all the formulas are attached as column *Truth* to the similarity matrix to obtain the final product of the process, which is the input to be used for training and testing a machine learning model.

Now that we have created our labeled datasets, we need to choose a machine learning method that is suitable for the task. Through a process of model selec-

tion we decided to use a support vector classifier, it performed the best with the small training sets that we provided. After performing a grid search we determined the best hyper parameters, we set the regularization parameter C to 0.1 and the kernel type to *polynomial*, of degree 3. We ran, for each of the 3 datasets, a 20-fold cross validation processes to establish the baseline performance of our proposed method. All results presented in Table 1 are averages of the scores obtained from all the runs for each dataset. Similarly, confusion matrices displayed in Figure 3 are the result of summing up all confusion matrices of the 20 runs and then normalizing them. We set the number of formulas included in the training sets of each universe to less than or equal to $\|\mathcal{A}\|$ of said universe, the training set sizes were as follows: 10 for Ω_{12}, 20 for Ω_{20}, and 30 for Ω_{30}. Table 2 presents the labeled formulas used in the training set of one of the runs for Ω_{12}.

No agent, human or artificial, has in its knowledge the exhaustive list of all possible formulas. The rationale for using small training sets in our experiments is that the knowledge base of an agent is unlikely to contain many formulas; some of them might be handcrafted and be part of the "background knowledge" of the agent, and the others acquired through sensors or messages received from other agents. In any case, it is a principle of economy that the knowledge of an agent be encoded using as few and as simple formulas as required. We want to test our proposal against a realistic scenario in which the available knowledge (the formulas whose truth value is known) is very small compared to the number of semantically distinct formulas that can be stated in the language. This also has the advantage of demonstrating the generalization capability of our models.

Notice that, for a language whose set of interpretations is Ω, there are $2^{\|\Omega\|}$ semantically distinct formulas, which is a really huge number, although most of them would be very complicated formulas that one would not expect to find in a real knowledge base. However, even factoring out very complicated formulas, the number of possible formulas would be exceedingly large.

To be sure, there exists a choice of formulas that would make the problem we are studying absolutely trivial. That is when the training set contains at least $\|\mathcal{A}\|$ formulas, each consisting of a single literal (i.e., a positive or negated atom), such that the atoms of these formulas are all distinct. It is easy to see that those formulas, with their truth labeling, would directly give the reference interpretation, from which the truth value of any other formula can be mechanically computed in linear time with respect to the length of the formula, *without performing any logical deduction or reasoning.*

This is the reason why the formulas of each dataset are extracted from a distribution that is skewed in favor of simpler (i.e., realistic), but not too simple formulas. Indeed, we ensure that the training set does not contain literals for all the atoms, by counting the number of literals that are randomly generated and rejecting additional literals once the maximum number of literals has been reached. That maximum is set to $\|\mathcal{A}\|/2$, well below $\|\mathcal{A}\|$.

Since the dataset consists of randomly generated formulas, it is unbalanced,[6] so in addition to the accuracy score, which we report because it gives an intuitive idea of the probability that the prediction is correct, we will provide the Matthews correlation coefficient (MCC) which is a statistical rate between -1 and 1 that produces a high score only if the prediction obtained good results in all of the four confusion matrix categories (true positives, false negatives, true negatives, and false positives), proportionally both to the size of positive elements and the size of negative elements in the dataset. So its a very good metric when we don't have a perfectly balanced dataset [8]. Results of this part of the experiment are presented in Table 1.

Algorithm 1 Generating random formulas

Require: A set of atoms $\mathcal{A}$
Ensure: $formulas$, a list of generated formulas
  $N \leftarrow Number\ of\ random\ formulas\ to\ generate$
  $f \leftarrow one\ randomly\ generated\ formula$
  $literals \leftarrow 0$ ▷ A counter used to limit the number of literals as sentences
  for $i = 1 \rightarrow N$ **do**
    $i \leftarrow i + 1$
    $f \leftarrow$ RANDOM_FORMULA$(\mathcal{A}, 0)$
    if $f\ in\ formulas$ **then**
      $i = i - 1$
      $continue$
    else if $length(f.symbols) == 1$ **then**
      if $literals < length(\mathcal{A})/2$ **then**
        $literals = literals + 1$
      else
        $i = i - 1$
        $continue$
      end if
    else
      $formulas.append(f)$
    end if
  end for

  function RANDOM_FORMULA($\mathcal{A}$, $level$)
    ▷ Recursive; the nesting $level$, initially $= 0$, is used to progressively reduce the probability of adding nested subformulas
    if RANDRANGE$(level + 4)$ **then** ▷ $\frac{level+3}{level+4}$ probability of stopping
      return CHOICE$(\mathcal{A})$
    end if
    if RANDRANGE$(3) = 0$ **then** ▷ with probability $\frac{1}{3}$
      return $\neg$RANDOM_FORMULA$(\mathcal{A}, level + 1)$
    end if
    return RANDOM_FORMULA$(\mathcal{A}, level + 1)$ $\cdot$ CHOICE$(\wedge, \vee)\cdot$
      RANDOM_FORMULA$(\mathcal{A}, level + 1)$
  end function
  Note: The generation process is slightly biased towards sentences that are neither too simple, nor too complex.

[6] The dataset for universe Ω_{12} has 272 false formulas and 227 true ones (1 missing because it was a duplicate), for universe Ω_{20} 278 false and 222 true, and for universe Ω_{30} 300 false and 200 true; of course, these figures vary (between training and test set) for every fold obtained from these datasets.

Algorithm 2 Approximating the similarity by sampling interpretations

```
Require: 2 formulas φ and ψ to be compared
Require: a sample size n
Ensure: φ and ψ, in the same universe Ω
Ensure: n > 0
  A ← φ.atoms ∪ ψ.atoms
  interpretations ← SAMPLE_INTERPRETATIONS(A, n)
  counter ← 0
  for w in interpretations do
      if φ.truth(w) == ψ.truth(w) then
          counter ← counter + 1
      end if
  end for
  ŝ_{φ,ψ} ← counter / n

  function SAMPLE_INTERPRETATIONS(A, n)
      interpretations ← array(size = n)          ▷ Array to store n interpretations
      for i = 0 → n − 1 do
          b ← array(size = A.length)             ▷ Array the size of the list of atoms
          for j = 0 → A.length − 1 do
              b[j] ← RANDRANGE(2)
          end for
          interpretations[i] ← b
      end for
      return interpretations
  end function
```

Part Two: Sampling Experiment. In this part of our experiment we investigate a property of our proposed similarity, which is the ability to approximate $\text{sim}(\phi, \psi)$ with acceptable accuracy by randomly sampling n interpretations from $\Omega_{\phi,\psi}$. We will name this approximation $\hat{s}_{\phi,\psi}$.

To this end, we created 3 new matrices for each set of formulas used in Section 4.2. We ended up with 9 new datasets, 3 for each of the universes depicted in Figure 1 and describe in Section 4.2. The similarity in the 9 new matrices was calculated differently than in Section 4.2. For this part, we approximated the similarity between formulas by randomly sampling a set number n of all interpretations, instead of taking all of them into account as we did in Section 4.2. The number of random samples n considered for creating the matrices is $n = 30$, $n = 100$, and $n = 1000$. This allows us to simulate the scenario of having a machine with low computational capacity trying to process a set of interpretations that is too large, and utilizing sampling as a solution. It also allows us to see how the method performs when noise is introduced.

The way the similarity is approximated using sampling is not much different from how it is calculated: we still count the instances where formulas ϕ and ψ have the same truth, but for n randomly sampled interpretations instead of *all* interpretations. Algorithm 2 is used to approximate the similarity between two formulas. In simple terms, when Algorithm 2 compares two formulas ϕ and ψ, instead of generating all interpretations corresponding to the set of unique atoms $\mathcal{A}$ composing those formulas, it generates a number n of these interpretations randomly. This sampling is done with replacement, which means that it is possible that a given interpretation gets sampled multiple times, especially in case n is larger than the number of all interpretations. We then proceed to count

the instances where formulas ϕ and ψ have the same truth out of these sampled interpretations. After that, we divide the obtained number by n, the size of the sample, since we are now dealing with n interpretations and not all of them. The result from that division is the approximation $\hat{s}_{\phi,\psi}$ of the similarity $\text{sim}(\phi, \psi)$ between ϕ and ψ. We do this for the sets of formulas we randomly generated in Section 4.2 for each of our universes 3 times, once for each number n of samples we mentioned.

With these 9 new matrices we are able to study how the sample size n might affect the performance of the method when dealing with different universe complexities. It will also show us how well an approximation of the similarity performs. We used the same model to test the performance and the same scoring metrics as the baseline. We used the same training set sizes for each universe as in the baseline. The results of this part of the experiment are available in Table 1.

Universe	Training set Size	sample size	Accuracy score	MCC
Ω_{12}	10	no sampling	0.77	0.56
		30	0.76	0.54
		100	0.77	0.56
		1000	0.77	0.55
Ω_{20}	20	no sampling	0.81	0.63
		30	0.81	0.62
		100	0.82	0.63
		1000	0.82	0.65
Ω_{30}	30	no sampling	0.83	0.66
		30	0.79	0.56
		100	0.82	0.62
		1000	0.83	0.64

Table 1: Accuracy and MCC for experiments done on each universe.

4.3 Results and Analysis

Baseline Results We start our analysis with the first part of our experiment, detailed in Section 4.2. The results can be found in Table 1 and the corresponding confusion matrices to offer support in Figure 3. A small sample of formulas from the test set for the smallest universe, with the labels predicted by the model, is provided in Table 3. From this experiment we can determine:

1. The overall performance of our proposed method without sampling while dealing with universes of different complexities, using very small training sets.

2. The effect the training set size has on performance with respect to the complexity of the universe addressed.

Regarding the first, Table 1 shows that the overall performance of our proposed method is good. The highest accuracy achieved was 83% for a training set size of 30 formulas and a universe of complexity 30, MCC being 0.66 which is a very good result when training using an unbalanced set. As a worst case, the method achieved 77% accuracy with a minimal training set size of 10 formulas and universe complexity of 12, MCC of 0.56 is acceptable considering the small training set relative to the complexity of the universe. Indeed, 30 formulas for a language with 30 propositional symbols is a really sparse training set, when one thinks that this language has $\sim 10^9$ interpretations and there exist $\sim 10^{300,000,000}$ semantically distinct formulas one can construct!

Formula	Label
$(\mathsf{on}(C,B) \wedge \mathsf{on}(B,C)) \vee (\neg\neg\neg\neg\mathsf{on}(A,B) \vee \mathsf{on}(B,A))$	True
$\neg(\neg(\mathsf{on}(A,B) \wedge ((\mathsf{on}(C,Tbl) \vee \neg\neg\mathsf{on}(C,B)) \wedge \mathsf{covered}(A))) \vee (\mathsf{on}(C,A) \vee \mathsf{on}(B,C)))$	False
$\neg(((\mathsf{covered}(B) \vee \neg\mathsf{on}(B,A)) \vee (\mathsf{on}(B,Tbl) \wedge (\mathsf{on}(B,A) \wedge \mathsf{on}(C,A)))) \wedge \mathsf{covered}(B))$	True
$\neg\neg\neg\mathsf{on}(A,C)$	True
$(\mathsf{on}(B,C) \vee \mathsf{covered}(A)) \vee \mathsf{covered}(A)$	True
$\neg(\neg\mathsf{on}(A,C) \wedge \neg\mathsf{covered}(C))$	False
$(\mathsf{on}(A,C) \vee \mathsf{on}(C,A)) \wedge \neg(\mathsf{covered}(C) \wedge \mathsf{on}(A,Tbl))$	False
$\mathsf{on}(C,Tbl) \wedge \mathsf{on}(C,B)$	False
$\mathsf{on}(C,B) \vee \mathsf{on}(B,C)$	False
$\neg\mathsf{on}(C,B) \wedge (\neg\neg\neg\mathsf{on}(C,B) \wedge \mathsf{covered}(B))$	False

Table 2: A sample training set made of 10 formulas from universe Ω_{12}.

After demonstrating that our proposed method is capable of achieving good results with very small training sets, we move on to the second point. We can see that the proposed method can achieve an average accuracy of 80% throughout the runs that use a very small training set of 10 formulas for Ω_{12}, 20 formulas for Ω_{20}, and 30 formulas for Ω_{30}. It would be natural to think that as the universe complexity increases, a model would require a larger training set to maintain performance, which is what the results shown in Table 1 and Figure 3 confirm. From the results see in Table 1 where no sampling was considered, we can see that increasing the number of formulas included in the training set had a very significant effect on performance. This effect was not limited to maintaining performance, but it resulted in an improvement of up to 8% in accuracy and 0.14 in terms of MCC.

To put things into perspective, for Ω_{12} we used for training 10 formulas out of a possible $\sim 10^{1233}$ compared to 30 out of a possible $\sim 10^{300,000,000}$ for

Formula	Actual	Predicted
$\neg(\neg(\neg(\mathsf{on}(B,A) \wedge \mathsf{covered}(B)) \vee ((\mathsf{covered}(C) \vee \mathsf{on}(A,)B) \wedge \mathsf{on}(C,Tbl))) \vee \neg((\mathsf{on}(B,A) \wedge \mathsf{on}(A,Tbl)) \wedge \neg\mathsf{covered}(A)))$	False	False
$(\mathsf{on}(A,Tbl) \wedge \mathsf{on}(B,A)) \vee \neg\neg((\mathsf{on}(A,C) \wedge ((\mathsf{on}(A,Tbl) \vee \mathsf{covered}(A)) \wedge \mathsf{covered}(A))) \vee \mathsf{on}(C,B))$	True	True
$\mathsf{covered}(B) \wedge \mathsf{on}(A,Tbl)$	False	True
$((\mathsf{covered}(B) \vee (\neg\mathsf{on}(A,B) \wedge \mathsf{on}(C,Tbl))) \wedge (((\mathsf{on}(B,A) \wedge (\mathsf{on}(A,C) \wedge \mathsf{on}(C,Tbl))) \vee (\mathsf{on}(C,A) \vee \mathsf{on}(A,Tbl))) \wedge \mathsf{on}(A,Tbl))) \vee \mathsf{on}(C,A)$	True	True
$((\mathsf{covered}(A) \wedge \mathsf{covered}(B)) \vee (\mathsf{covered}(A) \vee \neg(((\mathsf{covered}(A) \vee \mathsf{on}(B,C)) \vee (\mathsf{on}(B,C) \wedge \mathsf{on}(B,Tbl))) \vee \mathsf{on}(C,A)))) \vee ((\mathsf{on}(B,A) \vee \mathsf{on}(A,C)) \wedge (((\mathsf{on}(C,Tbl) \wedge \mathsf{on}(B,C)) \wedge \mathsf{on}(A,C)) \wedge \mathsf{covered}(A)))$	True	True
$\mathsf{covered}(B) \wedge (((\mathsf{on}(C,B) \vee ((\mathsf{on}(C,B) \wedge \mathsf{on}(B,Tbl)) \wedge ((\neg\mathsf{covered}(B) \vee (\mathsf{on}(A,B) \wedge \neg\mathsf{on}(A,C))) \vee (\neg\neg\mathsf{on}(B,C) \wedge (\mathsf{covered}(C) \vee \mathsf{on}(A,B)))))) \vee \mathsf{covered}(C)) \wedge (\neg(\mathsf{on}(A,C) \vee \mathsf{on}(B,C)) \wedge ((\mathsf{covered}(A) \vee \mathsf{on}(C,B)) \vee (\mathsf{on}(A,Tbl) \wedge (\mathsf{on}(A,Tbl) \vee \mathsf{on}(C,B))))))$	False	False

Table 3: A small sample of the test set with formulas varying in complexity from universe Ω_{12}.

Ω_{30}. In other words, the transition from Ω_{12} with 4096 interpretations to Ω_{30} with $\sim 10^9$, resulted in a gain of 8% accuracy by just adding 20 formulas to the training set. The increase in the size of the training set is modest relative to the size of Ω_{30} or the number of semantically distinct formulas that can be constructed, while the gain of accuracy and balance in predictions in terms of MCC is significant.

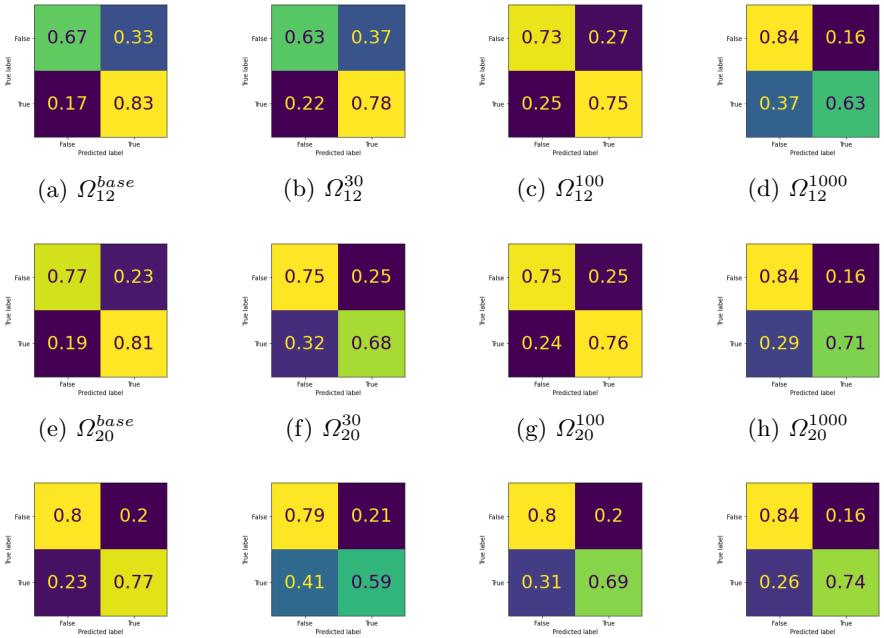
We observed that the truth value of "simple" formulas turns out to be harder to predict for the trained models than that of "complicated" formulas (see, e.g., Table 3). While this must have to do with the geometry of the space of the kernel representation of formulas induced by the semantic similarity, this phenomenon will have to be the object of further investigation.

Sampling Results We now shift our attention to part two of the experiment detailed in Section 4.2. The results of this experiment are also shown in Table 1 and the corresponding confusion matrices to offer support are found in Figure 3.

At first glance at Table 1, we can tell that the overall performance of the model does not degrade much when the similarity is approximated using the lowest number of samples $n = 30$. Indeed, we have a loss of accuracy of almost 4% for 2 of our universes. But this is an acceptable result when considering that in this case we would no longer have to calculate the exact similarity especially when we are limited by computational power. In fact, in this case, we would be looking at 30 random interpretations instead of $\sim 10^9$ for a universe the size of Ω_{30}.

The degradation in accuracy and MCC decreases as we increase the number of samples from 30 to 100 and then to 1000, it even approaches baseline performance. This proves what we mentioned in Section 3, we are able to approximate the similarity with very high accuracy even with a low number of sampled interpretations when compared to the number of *all* interpretations.

On the other hand, another increase in the number of samples from 100 to 1000 has no significant effect on performance, which is interesting considering that this introduces noise (since we allow for repetitions) yet it does not degrade performance. It also means that for a universe of complexity $\|\mathcal{A}\|$ there exists an optimal number of samples n that achieves baseline-similar performance.



(a) Ω_{12}^{base} (b) Ω_{12}^{30} (c) Ω_{12}^{100} (d) Ω_{12}^{1000}

(e) Ω_{20}^{base} (f) Ω_{20}^{30} (g) Ω_{20}^{100} (h) Ω_{20}^{1000}

(i) Ω_{30}^{base} (j) Ω_{30}^{30} (k) Ω_{30}^{100} (l) Ω_{30}^{1000}

Fig. 3: Confusion matrices of the similarity approximation using sampling experiment for each universe. Each row represents 4 cases for each universe, starting with the baseline (no sampling) and then $n = 30$, $n = 100$, and $n = 1000$ respectively. Each sub-figure is captioned by the universe notation $\Omega_{12,20,30}$ and in superscript the sample size n used to approximate similarity.

5 Conclusion

We have proposed a framework that allows an agent to train, based on a set of formulas whose truth values are known, a classification model that predicts the truth-value of a new, arbitrary formula. This framework uses a semantic similarity between formulas, which is a key ingredient of our proposal, to perform

a kernel encoding of the formulas, which is then exploited by the classification model. We have tested an implementation of this framework using SVM, showing that the classification model is highly accurate (with accuracy around 80%) even when the similarity is approximated by severely undersampling the interpretations. The practical implications of these results are that the proposed approach is tractable even for languages with a large (or infinite but enumerable) number of atoms; indeed, computing a good approximation of the similarity of two formulas can be done in linear time, as it depends only on the size of the (random) interpretations sampled.

There is no guarantee that all the predictions made by a model be altogether consistent. There is no built-in mechanism to ensure that and the mutual consistency of all the prediction is not part of the measure of the quality of a classifier: every prediction is made by the model and assessed independently of the others. Of course, if the predictions were all correct, they would also be consistent and that's what we observe empirically, that the predictions tend to be mostly consistent.

Since the knowledge of an agent may not be complete, some formulas, which are not entailed by it and whose negation is not entailed either, both predictions would be acceptable, and one might be tempted to count them as correct. However, this is not what we did: for the purpose of testing our method, what we did was to arbitrarily fix one interpretation and say it corresponded to the actual state of affairs; use it to label the training set and evaluate the predictions of the classifiers against the label that would be thus assigned, even for those formulas whose truth value is not constrained by the available knowledge. In a sense, we were as strict as one can be when judging a classifier.

Future work might involve testing other more sophisticated classification methods and applying the proposed framework to real-world scenarios.

Acknowledgments

This work has been partially supported by the French government, through the 3IA Côte d'Azur "Investments in the Future" project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002, as well as through the ANR CROQUIS (Collecte, représentation, complétion, fusion et interrogation de données de réseaux d'eau urbains hétérogènes et incertaines) project, grant ANR-21-CE23-0004 of the French National Research Agency (ANR).

References

1. Amgoud, L., David, V.: Measuring similarity between logical arguments. In: Thielscher, M., Toni, F., Wolter, F. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018. pp. 98–107. AAAI Press (2018)

2. Anthony, M., Hammer, P.L.: A boolean measure of similarity. Discrete Applied Mathematics 154(16), 2242–2246 (November 2006)
3. Antoniou, G., Ghose, A.: What is default reasoning good for? applications revisited. In: 32nd Annual Hawaii International Conference on System Sciences (HICSS-32), January 5-8, 1999, Maui, Hawaii, USA. IEEE Computer Society (1999)
4. Badreddine, S., d'Avila Garcez, A., Serafini, L., Spranger, M.: Logic tensor networks. Artif. Intell. 303, 103649 (2022), https://doi.org/10.1016/j.artint.2021.103649
5. Blee, J., Billington, D., Sattar, A.: Reasoning with levels of modalities in BDI logic. In: Ghose, A.K., Governatori, G., Sadananda, R. (eds.) Agent Computing and Multi-Agent Systems, 10th Pacific Rim International Conference on Multi-Agents, PRIMA 2007, Bangkok, Thailand, November 21-23, 2007. Revised Papers. Lecture Notes in Computer Science, vol. 5044, pp. 410–415. Springer (2007)
6. Bowles, G.: Propositional relevance. Informal Logic 2(12), 65–77 (Spring 1990)
7. Cheruvu, A., Radhakrishna, V.: A survey of similarity measures for time stamped temporal datasets. In: DATA. pp. 193–197. ACM (2021)
8. Chicco, D., Jurman, G.: The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. BMC Genomics 21 (2020)
9. Esteva, F., Godo, L., Rodríguez, R.O., Vetterlein, T.: On Ruspini's models of similarity-based approximate reasoning. In: IPMU (1). Communications in Computer and Information Science, vol. 1237, pp. 3–13. Springer (2020)
10. Fišer, P., Kubalík, P., Kubátová, H.: Output grouping method based on a similarity of boolean functions. In: Proc. of 7th Int. Workshop on Boolean Problems (IWSBP), Freiberg (Germany), September 21–22. pp. 107–113 (2006)
11. Hohenecker, P., Lukasiewicz, T.: Ontology reasoning with deep neural networks. J. Artif. Intell. Res. 68, 503–540 (2020)
12. Johnston, T., Scott, A.: Lipschitz bijections between boolean functions. Combinatorics, Probability and Computing 30, 513–525 (2021)
13. Makinson, D.: Propositional relevance through letter-sharing. J. Appl. Log. 7(4), 377–387 (2009)
14. Shi, S., Chen, H., Ma, W., Mao, J., Zhang, M., Zhang, Y.: Neural logic reasoning. In: d'Aquin, M., Dietze, S., Hauff, C., Curry, E., Cudré-Mauroux, P. (eds.) CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020. pp. 1365–1374. ACM (2020), https://doi.org/10.1145/3340531.3411949
15. Tran, T., Le, V., Le, H., Le, T.M.: From deep learning to deep reasoning. In: KDD '21: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 4076–4077 (2021)
16. Wendler, J., Bach, J.: Recognizing and predicting agent behavior with case based reasoning. In: RoboCup. Lecture Notes in Computer Science, vol. 3020, pp. 729–738. Springer (2003)