# Learning and Reasoning with Logic Tensor Networks

Luciano Serafini[1(✉)] and Artur S. d'Avila Garcez[2]

[1] Fondazione Bruno Kessler, Trento, Italy
serafini@fbk.eu
[2] City University London, London, UK
a.garcez@city.ac.uk

**Abstract.** The paper introduces real logic: a framework that seamlessly integrates logical deductive reasoning with efficient, data-driven relational learning. Real logic is based on full first order language. Terms are interpreted in $n$-dimensional feature vectors, while predicates are interpreted in fuzzy sets. In real logic it is possible to formally define the following two tasks: (i) learning from data in presence of logical constraints, and (ii) reasoning on formulas exploiting concrete data. We implement real logic in an deep learning architecture, called logic tensor networks, based on Google's TensorFlow™ primitives. The paper concludes with experiments on a simple but representative example of knowledge completion.

**Keywords:** Knowledge representation · Relational learning · Tensor networks · Neural-symbolic computation · Data-driven knowledge completion

## 1 Introduction

The availability of large repositories of resources that combines multiple modalities, (image, text, audio and sensor data, social networks), has fostered various research and commercial opportunities, underpinned by machine learning methods and techniques [1–4]. In particular, recent work in machine learning has sought to combine logical services, such as knowledge completion, approximate inference, and goal-directed reasoning with data-driven statistical and neural network-based approaches [5]. We argue that there are great possibilities for improving the current state of the art in machine learning and artificial intelligence (AI) through the principled combination of knowledge representation, reasoning and learning. Guha's recent position paper [6] is a case in point, as

The first author acknowledges the Mobility Program of FBK, for supporting a long term visit at City University London. He also acknowledges NVIDIA Corporation for supporting this research with the donation of a GPU. We also thank Prof. Marco Gori and his group at the University of Siena for the generous and inspiring discussions on the topic of integrating logical reasoning and statistical machine learning.


G. Adorni et al. (Eds.): AI*IA 2016, LNAI 10037, pp. 334–348, 2016.
DOI: 10.1007/978-3-319-49130-1_25

it advocates a new model theory for real-valued numbers. In this paper, we take inspiration from such recent work in AI, but also less recent work in the area of neural-symbolic integration [7–9] and in semantic attachment and symbol grounding [10] to achieve a vector-based representation which can be shown adequate for integrating machine learning and reasoning in a principled way.

This paper proposes a framework called *logic tensor networks (LTN)* which integrates learning based on tensor networks [5] with reasoning using first-order many-valued logic [11], all implemented in TensorFlow™ [12]. This enables, for the first time, a range of knowledge-based tasks using rich (full first-order logic (FOL)) knowledge representation to be combined with data-driven, efficient machine learning based on the manipulation of real-valued vectors. Given data available in the form of real-valued vectors, logical soft and hard constraints and relations which apply to certain subsets of the vectors can be compactly specified in first-order logic. Reasoning about such constraints can help improve learning, and learning from new data can revise such constraints thus modifying reasoning. An adequate vector-based representation of the logic, first proposed in this paper, enables the above integration of learning and inference, as detailed in what follows.

We are interested in providing a computationally adequate approach to implementing learning and reasoning in an integrated way within an idealized agent. This agent has to manage knowledge about an unbounded, possibly infinite, set of objects $O = \{o_1, o_2, \dots\}$. Some of the objects are associated with a set of quantitative attributes, represented by an $n$-tuple of real values $\mathcal{G}(o_i) \in \mathbb{R}^n$, which we call *grounding*. For example, the constant "john" denoting a person may have a grounding into an $n$-tuple of real numbers describing the "measures" of John, e.g., his height, weight, and number of friends in some social network. ....

Object tuples can participate in a set of relations $\mathcal{R} = \{R_1, \dots, R_k\}$, with $R_i \subseteq O^{\alpha(R_i)}$, where $\alpha(R_i)$ denotes the arity of relation $R_i$. We presuppose the existence of a latent (unknown) relation between the above numerical properties, i.e. groundings, and partial relational structure $\mathcal{R}$ on $O$. Starting from this partial knowledge, an agent is required to: (i) infer new knowledge about the relational structure $\mathcal{R}$ on the objects of $O$; (ii) predict the numerical properties or the class of the objects in $O$.

Classes and relations are not usually independent. For example, it may be the case that if an object $x$ is of class $C$, written as $C(x)$, and it is related to another object $y$ through relation $R(x, y)$ then this other object $y$ should be in the same class $C(y)$. In logic: $\forall x \exists y((C(x) \wedge R(x, y)) \rightarrow C(y))$. Whether or not $C(y)$ holds will depends on the application: through reasoning, one may derive $C(y)$ where otherwise there might not have been evidence of $C(y)$ from training examples only; through learning, one may need to revise such a conclusion once training counterexamples become available. The vectorial representation proposed in this paper permits both reasoning and learning as exemplified above and detailed in the next section.

The above forms of reasoning and learning are integrated in a unifying framework, implemented within tensor networks, and exemplified in relational domains combining data and relational knowledge about the objects. It is expected that, through an adequate integration of numerical properties and relational knowledge, differently from the immediate related literature [13–15], the framework introduced in this paper will be capable of combining in an effective way full first-order logical inference on open domains with efficient relational multi-class learning using tensor networks.

The main contribution of this paper is two-fold. It introduces a novel framework for the integration of learning and reasoning which can take advantage of the representational power of full (multi-valued) first-order logic, and it instantiates the framework using tensor networks into an efficient implementation which shows that the proposed vector-based representation of the logic offers an adequate mapping between symbols and their real-world manifestations, which is appropriate for both rich inference and learning from examples.

## 2 Real Logic

We start from a first order language $\mathcal{L}$, whose signature contains a set $\mathcal{C}$ of constant symbols, a set $\mathcal{F}$ of functional symbols, and a set $\mathcal{P}$ of predicate symbols. The sentences of $\mathcal{L}$ are used to express relational knowledge, e.g. the atomic formula $R(o_1, o_2)$ states that objects $o_1$ and $o_2$ are related to each other through the binary relation $R$; $\forall xy.(R(x, y) \rightarrow R(y, x))$ states that $R$ is a reflexive relation, where $x$ and $y$ are variables; $\exists y.R(o_1, y)$ states that there is an (unknown) object which is related to object $o_1$ through $R$. We assume that all sentences of $\mathcal{L}$ are in prenex conjunctive, skolemised normal form [16], e.g. a sentence $\forall x(A(x) \rightarrow \exists yR(x, y))$ is transformed into an equivalent clause $\neg A(x) \vee R(x, f(x))$, where $f$ is a new (Skolem) function symbol.

As for the semantics of $\mathcal{L}$, we deviate from the standard abstract semantics of FOL, and we propose a *concrete* semantics with sentences interpreted as tuples of real numbers. To emphasise the fact that $\mathcal{L}$ is interpreted in a "real" world we use the term *(semantic) grounding*, denoted by $\mathcal{G}$, instead of the more standard *interpretation*[1].

- $\mathcal{G}$ associates an $n$-tuple of real numbers $\mathcal{G}(t)$ to any closed term $t$ of $\mathcal{L}$; intuitively $\mathcal{G}(t)$ is the set of numeric features of the object denoted by $t$.
- $\mathcal{G}$ associates a real number in the interval $[0, 1]$ to each formula/clause $\phi$ of $\mathcal{L}$. Intuitively, $\mathcal{G}(\phi)$ represents one's confidence in the truth of $\phi$; the higher the value, the higher the confidence.

A grounding is specified only for the elements of the signature of $\mathcal{L}$. The grounding of terms and clauses is defined inductively, as follows.

[1] In logic, the term "grounding" indicates the operation of replacing the variables of a term/formula with constants, or terms that do not contains other variables. To avoid confusion, we use the synonym "instantiation" for this sense.

**Definition 1.** *Let $n > 0$. A* grounding $\mathcal{G}$ *for a first order language $\mathcal{L}$ is a function from the signature of $\mathcal{L}$ to the real numbers that satisfies the following conditions:*

1. $\mathcal{G}(c) \in \mathbb{R}^n$ *for every constant symbol* $c \in \mathcal{C}$;
2. $\mathcal{G}(f) \in \mathbb{R}^{n \cdot m} \longrightarrow \mathbb{R}^n$ *for every* $f \in \mathcal{F}$ *and* $m$ *is the arity of* $f$;
3. $\mathcal{G}(P) \in \mathbb{R}^{n \cdot m} \longrightarrow [0,1]$ *for every* $P \in \mathcal{P}$ *and* $m$ *is the arity of* $P$.

A grounding $\mathcal{G}$ is inductively extended to all the closed terms and clauses, as follows:

$$\begin{aligned}
\mathcal{G}(f(t_1, \ldots, t_m)) &= \mathcal{G}(f)(\mathcal{G}(t_1), \ldots, \mathcal{G}(t_m)) \\
\mathcal{G}(P(t_1, \ldots, t_m)) &= \mathcal{G}(P)(\mathcal{G}(t_1), \ldots, \mathcal{G}(t_m)) \\
\mathcal{G}(\neg P(t_1, \ldots, t_m)) &= 1 - \mathcal{G}(P(t_1, \ldots, t_m)) \\
\mathcal{G}(\phi_1 \vee \cdots \vee \phi_k) &= \mu(\mathcal{G}(\phi_1), \ldots, \mathcal{G}(\phi_k))
\end{aligned}$$

where $\mu$ is an s-norm operator (the co-norm operator associated some t-norm operator). Examples of t-norms are Lukasiewicz, product, and Gödel. Lukasiewicz s-norm is defined as $\mu_{Luk}(x,y) = \min(x+y, 1)$; Product s-norm is defined as $\mu_{Pr}(x,y) = x + y - x \cdot y$; Gödel s-norm is defined as $\mu_{max}(x,y) = \max(x,y)$. See [11] for details on fuzzy logics.

Notice that the previous semantics is defined only for clauses that do not contain variables. As happens in FOL the occurrences of variables in clauses are interpreted as universally quantified. For instance the clause $\neg A(x) \vee B(x)$ is interpreted as $\forall x(\neg A(x) \vee B(x))$. While the semantic of connectives is based on fuzzy logic operators, in giving the semantics of quantifiers, we deviate from fuzzy logic, introducing a different semantic that we call aggregation semantics. Intuitively the degree of truth of $\forall x P(x)$ is obtained by applying some aggregation function (e.g., mean) to the degree of truth of $P(t_1), \ldots, P(t_n)$, where $t_1, \ldots, t_n$ are considered to be a rappresentative sample of the domain of the variable x. The semantics of universally quantified formulas are defined w.r.t. an *aggregation operator* and a set of possible interpretations of the variables. More formually:

**Definition 2.** *Let $\phi(\mathbf{x})$ be a clause of $\mathcal{L}$ that contains a $k$-tuple $\mathbf{x} = \langle x_1, \ldots, x_k \rangle$ of $k$ distinct variables. Let $T$ be the set of all $k$-tuples $\mathbf{t} = \langle t_1, \ldots, t_k \rangle$ of closed terms of $\mathcal{L}$. $\mathcal{G}(\phi(\mathbf{x})) = \alpha_{\mathbf{t} \in T} \mathcal{G}(\phi(\mathbf{t}))$, where $\alpha$ is an aggregation operator from $[0,1]^{|T|} \to [0,1]$.*

Examples of aggregation operator that can be adopted in real logic are the minimum, the arithmetic mean, the geometric and the harmonic mean. At this stage we stay general and we simply suppose that $\alpha$ is a generic function from $[0,1]^{|T|}$ to $[0,1]$,

*Example 1.* Let $O = \{o_1, o_2, o_3, \ldots\}$ be a set of documents on a finite vocabulary $W = \{w_1, ..., w_n\}$ of $n$ words. Let $\mathcal{L}$ be the language that contains a constant (id) for every document, the binary function symbol $concat(x,y)$ denoting the

document resulting from the concatenation of documents $x$ with $y$ and the predicate $Contains(x, y)$ that means that content of $y$ is included in the document $x$. Let $\mathcal{L}$ contain also the binary predicate $Sim$ which is supposed to be *true* if document $x$ is deemed to be similar to document $y$. An example of grounding is the one that associates with each document its bag-of-words vector [17]. As a consequence, a natural grounding of the *concat* function would be the sum of the vectors, and of the *Sim* predicate, the cosine similarity between the vectors. More formally:

- $\mathcal{G}(o_i) = \langle n^{o_i}_{w_1}, \ldots, n^{o_i}_{w_n} \rangle$, where $n^d_w$ is the number of occurrences of word $w$ in document $d$;
- if $\mathbf{v}, \mathbf{u} \in \mathbb{R}^n$, $\mathcal{G}(concat)(\mathbf{u}, \mathbf{v}) = \mathbf{u} + \mathbf{v}$;
- if $\mathbf{v}, \mathbf{u} \in \mathbb{R}^n$, $\mathcal{G}(Sim)(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}||||\mathbf{v}||}$.
- if $\mathbf{v}, \mathbf{u} \in \mathbb{R}^n$, $\mathcal{G}(Contains)(\mathbf{u}, \mathbf{v})$ is a value in $[0, 1]$ that provides a confidence measure that the content of a document associated with the bag of words $\mathbf{u}$ is included in the document associated with the bag of words $\mathbf{v}$.

Notice that, while in the first three cases the $\mathcal{G}$ is explicitly defined (and can be calculated algorithmically), the grounding of the predicate *Contains* need to be "learned" from a set of examples.

Let us now see how to compute the grounding of terms and formulas. Let $o_1$, $o_2$, $o_3$ be the following three documents:

$o_1$ = "*John studies logic and plays football*"
$o_2$ = "*Mary plays football and logic games*"
$o_3$ = "*John and Mary play football and study logic together*"

We have that W = {*John, Mary, and, football, game, logic, play, study, together*}. The following are examples of the grounding of terms, atomic forulas and clauses.

$$\mathcal{G}(o_1) = \langle 1, 0, 1, 1, 0, 1, 1, 1, 0 \rangle$$
$$\mathcal{G}(o_2) = \langle 0, 1, 1, 1, 1, 1, 1, 0, 0 \rangle$$
$$\mathcal{G}(o_3) = \langle 1, 1, 2, 1, 0, 1, 1, 1, 1 \rangle$$
$$\mathcal{G}(concat(o_1, o_2)) = \mathcal{G}(o_1) + \mathcal{G}(o_2) = \langle 1, 1, 2, 2, 1, 2, 2, 1, 0 \rangle$$
$$\mathcal{G}(Sim(concat(o_1, o_2), o_3) = \frac{\mathcal{G}(concat(o_1, o_2)) \cdot \mathcal{G}(o_3)}{||\mathcal{G}(concat(o_1, o_2))|| \cdot ||\mathcal{G}(o_3)||} \approx \frac{13}{14.83} \approx 0.88$$
$$\mathcal{G}(Sim(o_1, o_3) \vee Sim(o_2, o_3)) = \mu_{max}(\mathcal{G}(Sim(o_1, o_3), \mathcal{G}(Sim(o_2, o_3)) \approx 0.86$$

$\mathcal{G}(Contains)$ cannot be defined directly. It should be learned from a set of examples, as the following:

- $\mathcal{G}(Contains(o_1, o_2))$ should be equal or close to 0
- $\mathcal{G}(Contains(o_1, o_3))$ should be equal or close to 1
- $\mathcal{G}(Contains(concat(o_i, o_j), o_i))$ for any $o_i$, $o_j$ should be equal or close to 1
- as well as $\mathcal{G}(Contains(o_i, o_i))$.

The above desiderata (aka constraints) can be expressed by the following first order formulas $\neg Contains(o_1, o_2)$, $Contains(o_1, o_3)$, $\forall xy Contains(concat(x, y), x)$, and $\forall x Contains(x, x)$.

## 3 Learning as Approximate Satisfiability

We start by defining grounded theory and their satisfiability.

**Definition 3 (Satisfiability of clauses).** *Let $\phi$ be a clause in $\mathcal{L}$, $\mathcal{G}$ a grounding, and $v \leq w \in [0, 1]$. We say that $\mathcal{G}$ satisfies $\phi$ in the confidence interval $[v, w]$, written $\mathcal{G} \models_v^w \phi$, if $\mathcal{G}(\phi) \in [v, w]$.*

A partial grounding, denoted by $\hat{\mathcal{G}}$, is a grounding that is defined on a subset of the signature of $\mathcal{L}$. A grounding $\mathcal{G}$ is said to be an extension of a partial grounding $\hat{\mathcal{G}}$ if $\mathcal{G}$ coincides with $\hat{\mathcal{G}}$ on the symbols where $\hat{\mathcal{G}}$ is defined.

**Definition 4 (Grounded Theory).** *A* grounded theory (GT) *is a pair $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ where $\mathcal{K}$ is a set of pairs $\langle [v, w], \phi(\mathbf{x}) \rangle$, where $\phi(\mathbf{x})$ is a clause of $\mathcal{L}$ containing the set $\mathbf{x}$ of free variables, and $[v, w] \subseteq [0, 1]$, and $\hat{\mathcal{G}}$ is a partial grounding.*

**Definition 5 (Satisfiability of a Grounded Theory).** *A GT $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ is satisfiable if there exists a grounding $\mathcal{G}$, which extends $\hat{\mathcal{G}}$ such that for all $\langle [v, w], \phi(\mathbf{x}) \rangle \in \mathcal{K}$, $\mathcal{G} \models_v^w \phi(\mathbf{x})$.*

From the previous definition it follows that checking if a GT $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ is satisfiable amounts to searching for an extension of the partial grounding $\hat{\mathcal{G}}$ in the space of *all possible groundings*, such that the aggregation according to $\alpha$ of *all* the instantiations of the clauses in $\mathcal{K}$ are satisfied w.r.t. the specified interval. Clearly a direct approach that follows the definition is unfeasible from a practical point of view. We therefore look for a form of partial satisfiability, which is computationally sustainable. There are three dimensions along which we can approximate, namely: (i) the form of $\mathcal{G}$, (ii) the set of instantiations of the free variables $\mathbf{x}$ for each clause that need to be considered to compute the aggregation function associated to the semantics of the universal quantifier and (iii) the closeness of the truth value of the clause to the indicated interval. Let us look at them separately.

*Remark 1 (Limiting to regular groundings).* To check satisfiability we search in all the possible functions on real numbers. However, we have said at the beginning of the paper that a grounding captures some latent correlation between the quantitative attribute of an object and its relational properties. For instance, the fact that a document is classified in the field of Artificial Intelligence depends on its bag-of-words grounding. If the language $\mathcal{L}$ contains the unary predicate $AI(x)$, standing for $x$ is a paper about Artificial Intelligence, then the grounding of $AI$ (which is a function from bag-of-words to $[0, 1]$) should present some

regularities such that it prefers bag-of-words vectors with high values associated to terms semantically closed to "artificial intelligence". Furthermore, if two bag-of-words are rather similar (e.g., according to the cosine similarity) their classification should agree (i.e., either they are both AI or both not). This imposes some regularity constraints on the grounding associated to the predicate $AI$, i.e., on $\mathcal{G}(AI)$. More in general, we are interested in groundings that present some regularities, and we search them within a specific class of functions (e.g., based on neural networks, linear/polinomial functions, gaussian mistures, etc.) In this paper we will concentrate on functions based on tensor networks, but the approach could be applied to any family of functions which can be characterised by a set $\theta$ of parameters. (For instance if we choose second order polynomial functions the parameters in $\theta$ are the coefficients of the polynom).

*Remark 2 (Limiting the number of clauses instantiations).* In general the set of closed terms of $\mathcal{L}$ might be infinite. It's enough to have one constant $c$ and a functional symbol $f$ to have an infinite set of closed terms $\{c, f(c), f(f(c)), \dots\}$. If we have one single open clause $\langle [v, w], \phi(x)\rangle$ in $\mathcal{K}$, then for every grounding $\mathcal{G}$ which is a completion of $\hat{\mathcal{G}}$, we have to perform an infinite set of tests to check if $\mathcal{G}(\phi(c)) \in [v, w]$, $\mathcal{G}(\phi(f(c))) \in [v, w]$, $\mathcal{G}(\phi(f(f(c)))) \in [v, w], \dots$ This is clearly impossible, and therefore we should implement some form of sampling. The obvious way to sample, is to consider the instantiations of $\phi$ to terms up to a certain complexity.

*Remark 3 (Degree of satisfiability).* Following some intuition described in [18], we see the satisfiability problem as an optimisation problem that try to maximize the level of satisfiability of a grounded theory, by minimising some loss function. So instead of looking for a grounding that satisfies all the formulas, we search for a grounding that satisfies *as much a as possible.*

For any $\langle [v, w], \phi(\mathbf{x})\rangle \in \mathcal{K}$ we want to find a grounding $\mathcal{G}$ that minimizes the *satisfiability error* with respect to an approximation of the domain of $\mathbf{x}$. An error occurs when a grounding $\mathcal{G}$ assigns a value $\mathcal{G}(\phi)$ to a clause $\phi$ which is outside the interval $[v, w]$ prescribed by $\mathcal{K}$. The measure of this error can be defined as the minimal distance between the points in the interval $[v, w]$ and $\mathcal{G}(\phi)$. For a closed clause $\phi$ we define

$$\text{Loss}(\mathcal{G}, \langle [v, w], \phi\rangle) = \min_{v \le x \le w} |x - \mathcal{G}(\phi)| \tag{1}$$

Notice that if $\mathcal{G}(\phi) \in [v, w]$, $\text{Loss}(\mathcal{G}, \phi) = 0$. For open clauses we have also to identify a subset of terms in which the variables need to be interpreted. We therefore define

$$\text{Loss}(\mathcal{G}, \langle [v, w], \phi\rangle, T_{\phi(\mathbf{x})}) = \min_{v \le x \le w} \left| x - \alpha_{\mathbf{t} \in T_{\phi(\mathbf{x})}} \mathcal{G}(\phi(\mathbf{t})) \right| \tag{2}$$

where, $T_{\phi(\mathbf{x})}$ is a finite set of $k$-tuples of grounded terms of $\mathcal{L}$, with $|\mathbf{x}| = k$, and $\alpha$ is the aggregation function used for interpreting the universally quantified variables. The above gives rise to the following definition of approximate satisfiability w.r.t. a family $\mathbb{G}$ of grounding functions on the language $\mathcal{L}$.

**Definition 6 (Best satisfiability problem).** *Let $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ be a grounded theory. Let $\mathbb{G}$ be a family of grounding functions. For every $\langle [v, w], \phi(\mathbf{x}) \rangle \in \mathcal{K}$, let $T_{\phi(\mathbf{x})}$ be a subset of all the $k$-tuples of ground terms of $\mathcal{L}$. We define the* best satisfiability problem *as the problem of finding an extensions $\mathcal{G}^*$ of $\hat{\mathcal{G}}$ in $\mathbb{G}$ that minimizes the satisfiability error on the clauses of $\mathcal{K}$ interpreted w.r.t the relative domain*

$$\mathcal{G}^* = \underset{\hat{\mathcal{G}} \subseteq \mathcal{G} \in \mathbb{G}}{\operatorname{argmin}} \sum_{\langle [v,w], \phi(\mathbf{x}) \rangle \in \mathcal{K}} Loss(\mathcal{G}, \langle [v, w], \phi(\mathbf{x}) \rangle, T_{\phi(\mathbf{x})})$$

## 4 Implementing Real Logic in Tensor Networks

Specific instances of Real Logic can be obtained by selecting the space $\mathbb{G}$ of groundings and the specific s-norm for the interpretation of disjunction and the $\alpha$ aggregation function for the interpretation of universally quantified variables. In this section, we describe a realization of real logic where $\mathbb{G}$ is the space of real tensor transformations of order $k$ (where $k$ is a parameter). In this space, function symbols are interpreted as linear transformations. More precisely, if $f$ is a function symbol of arity $m$ and $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{R}^n$ are real vectors corresponding to the grounding of $m$ terms then $\mathcal{G}(f)(\mathbf{v}_1, \ldots, \mathbf{v}_m)$ is of the form:

$$\mathcal{G}(f)(\mathbf{v}_1, \ldots, \mathbf{v}_m) = M_f \mathbf{v}^T + N_f$$

for some $n \times mn$ matrix $M_f$ and $n$-vector $N_f$, where $\mathbf{v} = \langle \mathbf{v}_1, \ldots, \mathbf{v}_n \rangle$.

The grounding of an $m$-ary predicate $P$, $\mathcal{G}(P)$, is defined as a generalization of the neural tensor network [5] (which has been shown effective at knowledge compilation in the presence of simple logical constraints), as a function from $\mathbb{R}^{mn}$ to $[0, 1]$, as follows:

$$\mathcal{G}(P) = \sigma \left( u_P^T \tanh \left( \mathbf{v}^T W_P^{[1:k]} \mathbf{v} + V_P \mathbf{v} + B_P \right) \right) \tag{3}$$

where $W_P^{[1:k]}$ is a 3-D tensor in $\mathbb{R}^{mn \times mn \times k}$, $V_P$ is a matrix in $\mathbb{R}^{k \times mn}$, and $B_P$ is a vector in $\mathbb{R}^k$, and $\sigma$ is the sigmoid function. With this encoding, the grounding (i.e. truth-value) of a clause can be determined by a neural network which first computes the grounding of the literals contained in the clause, and then combines them using the specific s-norm. An example of tensor network for $\neg P(x, y) \rightarrow A(y)$ is shown in Fig. 1.

This architecture has been used in combination with vector embedding, i.e., a way to associate a vectorial semantics to concepts and relations, which is, in our terms, a specific example of grounding function. In the above architecture the node shown in gray contains the parameters that need to be learned in order to minimize the loss function, or equivalently, to maximize the satisfaction degree of a grounded theory. In the above tensor network formulation, $W_*, V_*, B_*$ and $u_*$ with $* \in \{P, A\}$ are parameters to be learned by minimizing the loss function or, equivalently, to maximize the satisfiability of the clause $P(x, y) \rightarrow A(y)$. We

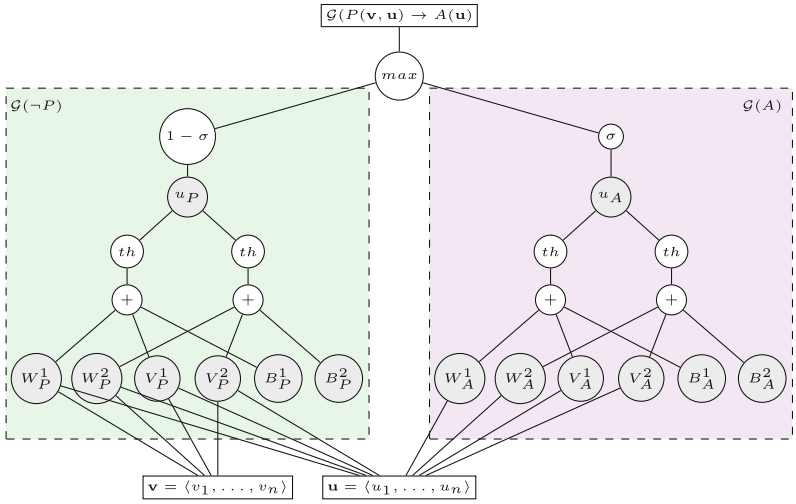


**Fig. 1.** Tensor net for $P(x, y) \to A(y)$, with $\mathcal{G}(x) = \mathbf{v}$ and $\mathcal{G}(y) = \mathbf{u}$ and $k = 2$.

have developed a python library that supports the definition, and the parameter learning of the structure described above. Such a library is based on Google's TensorFlow™[2].

## 5 An Example of Knowledge Completion

To test our idea, in this section we use the well-known *friends and smokers*[3] example [19] to illustrate the task of knowledge completion in LTN. There are 14 people divided into two groups $\{a, b, \ldots, h\}$ and $\{i, j, \ldots, n\}$. Within each group of people we have complete knowledge of their smoking habits. In the first group, we have complete knowledge of who has and does not have cancer. In the second group, this is not known for any of the persons. Knowledge about the friendship relation is complete within each group only if symmetry of friendship is assumed. Otherwise, it is incomplete in that it may be known that, e.g., $a$ is a friend of $b$, but not known whether $b$ is a friend of $a$. Finally, there is also general knowledge about smoking, friendship and cancer, namely, that smoking causes cancer, friendship is normally a symmetric and anti-reflexive relation, everyone has a friend, and that smoking propagates (either actively or passively) among friends. All this knowledge can be represented by the knowledge-bases shown in Fig. 2.

Notice that, if we adopt classical FOL semantics the knowledge base $\mathcal{K}^{SFC} = \mathcal{K}^{SFC}_{x,y} \cup \mathcal{K}^{SF}_{a,\ldots,h} \cup \mathcal{K}^{SFC}_{i,\ldots,n}$ is inconsistent. Indeed the axiom $\forall x(S(x) \to C(x))$

[2] https://www.tensorflow.org/.

[3] Normally, a probabilistic approach is taken to solve this problem, and one that requires instantiating all clauses to remove variables, essentially turning the problem into a propositional one; `ltn` takes a different approach.

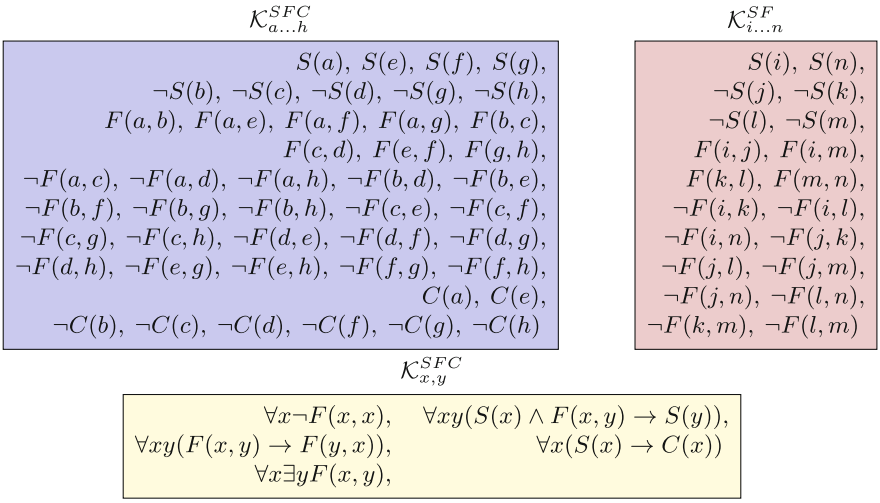


**Fig. 2.** Knowledge-bases for the friends-and-smokers example.

contained in $\mathcal{K}^{SFC}_{x,y}$ contraddicts the facts $S(f)$ and $\neg C(f)$ contained in $\mathcal{K}^{SFC}_{a,...,h}$. If instead we admit the possibility for facts in $\mathcal{K}^{SFC}$ to be true "to a certain extent", i.e., they are associated to an (unknown) degree of truth smaller than 1, then $\mathcal{K}^{SFC}$ is consistent and admits a grounding. Our main tasks are:

(i) find the maximum degree of truth of the facts contained so that $\mathcal{K}^{SFC}$ is consistent;
(ii) find a truth-value for all the ground atomic facts not explicitly mentioned in $\mathcal{K}^{SFC}$;
(iii) find the grounding of each constant symbol $a, ..., n$[4].

To answer (i)-(iii), we use LTN to find a grounding that best approximates the complete KB. To show the role of background knolwedge in the learning-inference process, we run two experiments. In the first ($exp1$), we seek to complete KB consisting of only factual knowledge: $\mathcal{K}^{SFC}_{exp1} = \mathcal{K}^{SFC}_{a...h} \cup \mathcal{K}^{SF}_{i...n}$. In the second ($exp2$), we include background knowledge, that is: $\mathcal{K}^{SFC}_{exp2} = \mathcal{K}_{exp1} \cup \mathcal{K}^{SFC}_{x,y}$.

We configure the network as follows: each constant (person, in this case) has 30 real features. The number of layers $k$ in the tensor network equal to 10, and the regularization parameter[5] $\lambda = 1^{-10}$. We choose the Lukasiewicz t-norm[6], and use the harmonic mean as aggregation operator. An estimation of the optimal grounding is obtained by 5,000 runs of the RMSProp optimisation algorithm [20] available in TensorFlow™.

The results of the two experiments are reported in Table 1. For readability, we use boldface for truth-values greater than 0.5. The truth-values of the facts

[4] Notice how no grounding is provided about the signature of the knowledge-base.
[5] A smoth factor $\lambda||\mathbf{\Omega}||_2^2$ is added to the loss to limit the size of parameters.
[6] $\mu(a,b) = \min(1, a+b)$.

listed in a knowledge-base are highlighted with the same background color of the knowledge-base in Fig. 2. The values with white background are the result of the knowledge completion produced by the LTN learning-inference procedure. To evaluate the quality of the results, one has to check whether (i) the truth-values of the facts listed in a KB are indeed close to 1.0, and (ii) the truth-values associated with knowledge completion correspond to expectation. An initial analysis shows that the LTN associated with $\mathcal{K}_{exp1}$ produces the same facts as $\mathcal{K}_{exp1}$ itself. In other words, the LTN fits the data. However, the LTN also learns to infer additional positive and negative facts about the predicates $F$ (friends) and $C$ (cancer) not derivable from $\mathcal{K}_{exp1}$ by pure logical reasoning; for example: $F(c,b)$, $F(g,b)$ and $\neg F(b,a)$. These facts are derived by exploiting similarities between the groundings of the constants generated by the LTN. For instance, $\mathcal{G}(c)$ and $\mathcal{G}(g)$ happen to present a high cosine similarity measure. As a result, facts about the friendship relations of $c$ affect the friendship relations of $g$ and vice-versa, for instance $F(c,b)$ and $F(g,b)$. The level of satisfiability associated with $\mathcal{K}_{exp1} \approx 1$, which indicates that $\mathcal{K}_{exp1}$ is classically satisfiable.

The results of the second experiment show that more facts can be learned with the inclusion of background knowledge. For example, the LTN now predicts that

**Table 1.** Learning and reasoning in $\mathcal{K}_{exp1}$ and $\mathcal{K}_{exp2}$

| | $S$ | $C$ | $F$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $g$ | $h$ |
| $a$ | **1.00** | **1.00** | 0.00 | **1.00** | 0.00 | 0.00 | **1.00** | **1.00** | **1.00** | 0.00 |
| $b$ | 0.00 | 0.00 | 0.00 | 0.00 | **1.00** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| $c$ | 0.00 | 0.00 | 0.00 | **0.82** | 0.00 | **1.00** | 0.00 | 0.00 | 0.00 | 0.00 |
| $d$ | 0.00 | 0.00 | 0.00 | 0.00 | 0.06 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| $e$ | **1.00** | **1.00** | 0.00 | 0.33 | 0.21 | 0.00 | 0.00 | **1.00** | 0.00 | 0.00 |
| $f$ | **1.00** | 0.00 | 0.00 | 0.00 | 0.05 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| $g$ | **1.00** | 0.00 | 0.03 | **1.00** | **1.00** | **1.00** | 0.11 | **1.00** | 0.00 | **1.00** |
| $h$ | 0.00 | 0.00 | 0.00 | 0.23 | 0.01 | 0.14 | 0.00 | 0.02 | 0.00 | 0.00 |

| | $S$ | $C$ | $F$ | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | $i$ | $j$ | $k$ | $l$ | $m$ | $n$ |
| $i$ | **1.00** | 0.00 | 0.00 | **1.00** | 0.00 | 0.00 | **1.00** | 0.00 |
| $j$ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| $k$ | 0.00 | 0.00 | 0.10 | **1.00** | 0.00 | **1.00** | 0.00 | 0.00 |
| $l$ | 0.00 | 0.00 | 0.00 | 0.02 | 0.00 | 0.00 | 0.00 | 0.00 |
| $m$ | 0.00 | 0.03 | **1.00** | **1.00** | 0.12 | **1.00** | 0.00 | **1.00** |
| $n$ | **1.00** | 0.01 | 0.00 | 0.98 | 0.00 | 0.01 | 0.02 | 0.00 |

Learning and reasoning on $\mathcal{K}_{exp1} = \mathcal{K}^{SFC}_{a\ldots h} \cup \mathcal{K}^{SF}_{i\ldots n}$

| | $S$ | $C$ | $F$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $g$ | $h$ |
| $a$ | **0.84** | **0.87** | 0.02 | **0.95** | 0.01 | 0.03 | **0.93** | **0.97** | **0.98** | 0.01 |
| $b$ | 0.13 | 0.16 | 0.45 | 0.01 | **0.97** | 0.04 | 0.02 | 0.03 | 0.06 | 0.03 |
| $c$ | 0.13 | 0.15 | 0.02 | **0.94** | 0.11 | **0.99** | 0.03 | 0.16 | 0.15 | 0.15 |
| $d$ | 0.14 | 0.15 | 0.01 | 0.06 | **0.88** | 0.08 | 0.01 | 0.03 | 0.07 | 0.02 |
| $e$ | **0.84** | **0.85** | 0.32 | 0.06 | 0.05 | 0.03 | 0.04 | **0.97** | 0.07 | 0.06 |
| $f$ | **0.81** | 0.19 | 0.34 | 0.11 | 0.08 | 0.04 | 0.42 | 0.08 | 0.06 | 0.05 |
| $g$ | **0.82** | 0.19 | **0.81** | 0.26 | 0.19 | 0.30 | 0.06 | 0.28 | 0.00 | **0.94** |
| $h$ | 0.14 | 0.17 | 0.05 | 0.25 | 0.26 | 0.16 | 0.20 | 0.14 | **0.72** | 0.01 |

| | $S$ | $C$ | $F$ | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | $i$ | $j$ | $k$ | $l$ | $m$ | $n$ |
| $i$ | **0.83** | **0.86** | 0.02 | **0.91** | 0.01 | 0.03 | **0.97** | 0.01 |
| $j$ | 0.19 | 0.22 | **0.73** | 0.03 | 0.00 | 0.04 | 0.02 | 0.05 |
| $k$ | 0.14 | 0.34 | 0.17 | 0.07 | 0.04 | **0.97** | 0.04 | 0.02 |
| $l$ | 0.16 | 0.19 | 0.11 | 0.12 | 0.15 | 0.06 | 0.05 | 0.03 |
| $m$ | 0.14 | 0.17 | **0.96** | 0.07 | 0.02 | 0.11 | 0.00 | **0.92** |
| $n$ | **0.84** | **0.86** | 0.13 | 0.28 | 0.01 | 0.24 | **0.69** | 0.02 |

| | $a,\ldots,h,i,\ldots,n$ |
|---|---|
| $\forall x \neg F(x,x)$ | **0.98** |
| $\forall xy(F(x,y) \rightarrow F(y,x))$ | **0.90** **0.90** |
| $\forall x(S(x) \rightarrow C(x))$ | **0.77** |
| $\forall x(S(x) \wedge F(x,y) \rightarrow S(y))$ | **0.96** **0.92** |
| $\forall x \exists y(F(x,y))$ | **1.0** |

Learning and reasoning on $\mathcal{K}_{exp2} = \mathcal{K}^{SFC}_{a\ldots h} \cup \mathcal{K}^{SF}_{i\ldots n} \cup \mathcal{K}^{SFC}$

$C(i)$ and $C(n)$ are true. Similarly, from the symmetry of the friendship relation, the LTN concludes that $m$ is a friend of $i$, as expected. In fact, all the axioms in the generic background knowledge $\mathcal{K}^{SFC}$ are satisfied with a degree of satisfiability higher than 90 %, apart from the *smoking causes cancer* axiom - which is responsible for the classical inconsistency since in the data $f$ and $g$ smoke and do not have cancer -, which has a degree of satisfiability of 77 %.

## 6 Related Work

In his recent note, Guha [6], advocates the need for a new model theory for distributed representations (such as those based on embeddings). The note sketches a proposal, where terms and (binary) predicates are all interpreted as points/vectors in an $n$-dimensional real space. The computation of the truth-value of the atomic formulae $P(t_1, \ldots, t_n)$ is obtained by comparing the projections of the vector associated to each $t_i$ with that associated to $P_i$. Real logic shares with [6] the idea that terms must be interpreted in a geometric space. It has, however, a different (and more general) interpretation of functions and predicate symbols. Real Logic is more general because the semantics proposed in [6] can be implemented within an `ltn` with a single layer ($k = 1$), since the operation of projection and comparison necessary to compute the truth-value of $P(t_1, \ldots, t_m)$ can be encoded within an $nm \times nm$ matrix $W$ with the constraint that $\langle \mathcal{G}(t_1), \ldots, \mathcal{G}(t_n) \rangle^T W \langle \mathcal{G}(t_1), \ldots, \mathcal{G}(t_n) \rangle \leq \delta$, which can be encoded easily in `ltn`.

Real Logic is orthogonal to the approach taken by (Hybrid) Markov Logic Networks (MLNs) and its variations [19,21,22]. In MLNs, the level of truth of a formula is determined by the number of models that satisfy the formula: the more models, the higher the degree of truth. Hybrid MLNs introduce a dependency on the real features associated to constants, which is given, and not learned. In Real Logic, instead, the level of truth of a complex formula is determined by (fuzzy) logical reasoning, and the relations between the features of different objects is learned through error minimization. Another difference is that MLNs work under the *closed world assumption*, while Real Logic is open domain. Much work has been done also on neuro-fuzzy approaches [23]. These are essentially propositional while real logic is first-order.

Bayesian logic (BLOG) [24] is open domain, and in this respect is similar to real logic and LTNs. But, instead of taking an explicit probabilistic approach, LTNs draw from the efficient approach used by tensor networks for knowledge graphs, as already discussed. LTNs can have a probabilistic interpretation but this is not a requirement. Other statistical AI and probabilistic approaches such as lifted inference fall in this same category, including probabilistic variations of inductive logic programming (ILP) [25], which are normally restricted to Horn clauses. Metainterpretive ILP [26], together with BLOG, seems closer to LTNs in what concerns the knowledge representation language, but without exploring tensor networks and its benefits in terms of computational efficiency.

Similarly to [9], LTN is a framework for learning in presence of logical constraints. The two approaches share the idea that logical constraints and training examples should be considered uniformly as supervisions of a learning algorithm. The learning process is an optimization that minimize the distance between the supervisions and the predictions. LTN introduces two novelties: First, in LTN existential quantifiers are not grounded into a finite disjunction, but it is skolemized. This means that we do not assume closed world assumption, and existential quantified formula can be satisfied by "new" individuals. Second, LTN allows to generate and predict data. For instance if a grounded theory contains the formula $\forall x \exists y R(x, y)$, LTN generates a real function (corresponding to the grounding of the skolem function intoduced in the formula) that, for every feature vector $\mathbf{v}$ returns the feature vector $f(\mathbf{v})$, which can be intuitively interpreted as the set of features of the *typical* $R$-related object of an object with features equal to $\mathbf{v}$.

Finally, related work in the domain of neural-symbolic computing and neural network fibring [8] has sought to combine neural networks with ILP to gain efficiency [27] and other forms of knowledge representation, such as propositional modal logic and logic programming. The above are more tightly-coupled approaches. In contrast, LTNs use a richer FOL language, exploit the benefits of knowledge compilation and tensor networks within a more loosely- coupled approach, and might even offer an adequate representation of equality in logic. Experimental evaluations and comparison with other neural-symbolic approaches are desirable though, including the latest developments in the field, a good snapshot of which can be found in [15].

## 7 Conclusion and Future Work

We have proposed *Real Logic*: a uniform framework for learning and reasoning. Approximate satisfiability is defined as a learning task with both knowledge and data being mapped onto real-valued vectors. With an inference-as-learning approach, relational knowledge constraints and state-of-the-art data-driven approaches can be integrated. We showed how Real Logic can be implemented in deep tensor networks, which we call Logic Tensor Networks (LTNs), and applied efficiently to knowledge completion and data prediction tasks. As future work, we will make the implementation of LTN available in TensorFlow™ and apply it to large-scale experiments and relational learning benchmarks for comparison with statistical relational learning, neural-symbolic computing, and (probabilistic) inductive logic programming approaches.

## References

1. Bengio, Y.: Learning deep architectures for AI. Found. Trends Mach. Learn. **2**, 1–127 (2009)
2. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of go with deep neural networks and tree search. Nature **529**, 484–503 (2016)

3. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**, 41–50 (2003)
4. Kiela, D., Bottou, L.: Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In: Proceedings of EMNLP 2014 (2014)
5. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in Neural Information Processing Systems, pp. 926–934 (2013)
6. Guha, R.: Towards a model theory for distributed representations. In: 2015 AAAI Spring Symposium Series (2015)
7. Bottou, L.: From machine learning to machine reasoning. Technical report, arXiv.1102.1808 (2011)
8. d'Avila Garcez, A.S., Lamb, L.C., Gabbay, D.M.: Neural-Symbolic Cognitive Reasoning. Cognitive Technologies. Springer, Heidelberg (2009)
9. Diligenti, M., Gori, M., Maggini, M., Rigutini, L.: Bridging logic and kernel machines. Mach. Learn. **86**, 57–88 (2012)
10. Barrett, L., Feldman, J., MacDermed, L.: A (somewhat) new solution to the variable binding problem. Neural Comput. **20**, 2361–2378 (2008)
11. Bergmann, M.: An Introduction to Many-Valued and Fuzzy Logic: Semantics, Algebras, and Derivation Systems. Cambridge University Press, New York (2008)
12. TensorFlow: Large-scale machine learning on heterogeneous systems (2016). tensorflow.org
13. d'Avila Garcez, A.S., Gori, M., Hitzler, P., Lamb, L.C.: Neural-symbolic learning and reasoning (dagstuhl seminar 14381). Dagstuhl Rep. **4**, 50–84 (2014)
14. McCallum, A., Gabrilovich, E., Guha, R., Murphy, K. (eds.): Knowledge representation and reasoning: integrating symbolic and neural approaches. In: AAAI Spring Symposium, Stanford University, CA, USA (2015)
15. Besold, T.R., d'Avila Garcez, A., Marcus, G.F., Miikulainen, R. (eds.): Cognitive computation: integrating neural and symbolic approaches. In: Workshop at NIPS 2015, Montreal, Canada, CEUR-WS 1583, April 2016
16. Huth, M., Ryan, M.: Logic in Computer Science: Modelling and Reasoning About Systems. Cambridge University Press, New York (2004)
17. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
18. Brys, T., Drugan, M.M., Bosman, P.A., De Cock, M., Nowé, A.: Solving satisfiability in fuzzy logics by mixing CMA-ES. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, GECCO 2013, pp. 1125–1132. ACM, New York (2013)
19. Richardson, M., Domingos, P.: Markov logic networks. Mach. Learn. **62**, 107–136 (2006)
20. Tieleman, T., Hinton, G.: Lecture 6.5 - RMSProp, COURSERA: Neural networks for machine learning. Technical report (2012)
21. Wang, J., Domingos, P.M.: Hybrid markov logic networks. In: AAAI, pp. 1106–1111 (2008)
22. Nath, A., Domingos, P.M.: Learning relational sum-product networks. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25–30, 2015, Austin, Texas, USA, pp. 2878–2886 (2015)
23. Kosko, B.: Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence. Prentice-Hall Inc., Upper Saddle River (1992)
24. Milch, B., Marthi, B., Russell, S.J., Sontag, D., Ong, D.L., Kolobov, A.: BLOG: probabilistic models with unknown objects. In: IJCAI 2005, pp. 1352–1359 (2005)

25. Raedt, L.D., Kersting, K., Natarajan, S., Poole, D.: Statistical Relational Artificial Intelligence: Logic, Probability, and Computation. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, San Rafael (2016)
26. Muggleton, S.H., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. Mach. Learn. **100**, 49–73 (2015)
27. França, M.V.M., Zaverucha, G., d'Avila Garcez, A.S.: Fast relational learning using bottom clause propositionalization with artificial neural networks. Mach. Learn. **94**, 81–104 (2014)