Joint Multi-Conference on Human-Level Artificial Intelligence HLAI 2016

Pre-Proceedings of the 11th International Workshop on Neural-Symbolic Learning and Reasoning NeSy'16

Tarek R. Besold, Luis C. Lamb, Luciano Serafini, Whitney Tabor
(eds.)

New York City, USA, 16th & 17th of July 2016

We, as workshop organizers, want to thank the following members of the NeSy'16 program committee for their time and efforts in reviewing the submissions to the workshop and providing valuable feedback to accepted and rejected papers and abstracts alike:

- Antoine Bordes, Facebook AI Research, USA
- Artur d'Avila Garcez, City University London, UK
- James Davidson, Google Inc., USA
- Robert Frank, Yale University, USA
- Ross Gayler, Melbourne, Australia
- Ramanathan V. Guha, Google Inc., USA
- Steffen Hoelldobler, Technical University of Dresden, Germany
- Thomas Icard, Stanford University, USA
- Kristian Kersting, Technical University of Dortmund, Germany
- Kai-Uwe Kuehnberger, University of Osnabrueck, Germany
- Simon Levy, Washington and Lee University, USA
- Stephen Muggleton, Imperial College London, UK
- Isaac Noble, Google Inc., USA
- Andrea Passerini, University of Trento, Italy
- Christopher Potts, Stanford University, USA
- Daniel L. Silver, Acadia University, Canada
- Ron Sun, Rensselaer Polytechnic Insitute, USA
- Jakub Szymanik, University of Amsterdam, The Netherlands
- Serge Thill, University of Skovde, Sweden
- Michael Witbrock, IBM, USA
- Frank van der Velde, University of Twente, The Netherlands

These workshop pre-proceedings are available online from the workshop webpage under http://www.neural-symbolic.org/NeSy16/.

Bozen-Bolzano, 10th of July 2016
Tarek R. Besold, Luis C. Lamb, Luciano Serafini, and Whitney Tabor.[1]

[1] Tarek R. Besold is a postdoctoral researcher at the KRDB Research Centre of the Free University of Bozen-Bolzano, Italy; Luis C. Lamb is Professor of Computer Science at UFRGS, Porto Alegre, Brazil; Luciano Serafini is the head of the Data and Knowledge Management Research Unit at Fondazione Bruno Kessler, Trento, Italy; Whitney Tabor is Associate Professor at the Department of Psychological Sciences at the University of Connecticut, USA.

Contents: Contributed Papers

Contents: Contributed Abstracts

Inducing Symbolic Rules from Entity Embeddings using Auto-encoders

Thomas Ager, Ondřej Kuželka, Steven Schockaert

School of Computer Science and Informatics, Cardiff University
{AgerT,KuzelkaO,SchockaertS1}@cardiff.ac.uk

Abstract. Vector space embeddings can be used as a tool for learning semantic relationships from unstructured text documents. Among others, earlier work has shown how in a vector space of entities (e.g. different movies) fine-grained semantic relationships can be identified with directions (e.g. more violent than). In this paper, we use stacked denoising auto-encoders to obtain a sequence of entity embeddings that model increasingly abstract relationships. After identifying directions that model salient properties of entities in each of these vector spaces, we induce symbolic rules that relate specific properties to more general ones. We provide illustrative examples to demonstrate the potential of this approach.

1 Introduction

In this paper, we consider the problem of how we can learn symbolic rules from unstructured text documents that describe entities of interest, e.g. how we can learn that thrillers tend to be violent from a collection of movie reviews. Obtaining meaningful and interpretable symbolic rules is important in fields like exploratory data analysis, or explaining classifier decisions, as they can be interpreted easily by human users.

A straightforward approach might be to directly learn rules from bag-of-words representations of documents. However, such an approach would typically lead to a large number of rules of little interest, e.g. rules pertaining more to which words are used together rather than capturing capturing meaningful semantic relationships. Our approach instead builds on the method from [6], which induces an entity embedding from unstructured text documents. Their method finds directions which correspond to interpretable properties in a vector space, labelled using adjectives and nouns that appear in the text collection. In particular, these directions induce a ranking of the entities that reflects how much they have the corresponding property. For example, in a space of wines, a direction may be found that corresponds to the property of being "Tannic", allowing us to rank wines based on the number of tannins.

In order to obtain symbolic rules, we first derive a series of increasingly general entity embeddings using auto-encoders (see Section 3). To induce rules from embeddings, we link properties derived from those embeddings together.

As an example, below is one of the rules we have derived using this method:

```
IF Emotions AND Journey THEN Adventure        (1)
```

Using a set of symbolic rules that qualitatively describe domain knowledge is a promising approach to generate supporting explanations. Explanations of classification decisions can give valuable insight into why a system produces a result. For example, in fields such as medicine it is important for experts to verify the predictions of a system and justify its classification decisions [7, 9]. In the domain of movies, we may have a situation where the synopsis or reviews mention the words "Emotions" and "Journey", from which the system could derive that it is probably an "Adventure" movie and use rule (1) as a supporting explanation. We note that the ideas presented in this paper may also be directly useful for explaining predictions of some kinds of deep neural networks.

The rest of the paper explains how we use unsupervised methods to learn rules such as (1). In Section 2, we recall the method from [6] for identifying interpretable directions in entity embeddings. Subsequently in Section 3 we detail how we build on this method using stacked denoising auto-encoders, and how we induce rules that explain the semantic relationships between the properties that we discover. In Section 4 we qualitatively examine these properties and rules, and in Section 5 we place our work in the context of related work. Finally, in Section 6 we provide our conclusions.

2 Learning Interpretable Directions

In this section, we recall the method from [6] that learns a vector space representation for the entities of a given domain of interest, such that salient properties of the domain correspond to directions in the vector space. The method proceeds in several steps, detailed next.

From bags-of-words to vectors. We use a text collection where each document describes an entity. For example, if the entities are movies, a collection of movie reviews. We first learn a vector space of entities using classical multidimensional scaling (MDS), which takes a dissimilarity matrix as input. MDS is commonly used in cognitive science to generate semantic spaces from similarity judgements that are provided by human annotators. It outputs a space where entities are represented as points and the Euclidean distance between entities reflects the given dissimilarity matrix as closely as possible. It was empirically found to lead to representations that are easier to interpret than the more commonly used singular value decomposition method [5]. To obtain a suitable dissimilarity matrix, we quantify how relevant each term is to an entity using Positive Pointwise Mutual Information (PPMI). PPMI scores terms highly if they are frequently associated with an entity but relatively infrequent over the entire text collection. We create PPMI vectors for each entity using the PPMI values for each word as the components of its vector, and calculate the dissimilarity between those vectors using the normalized angular difference. These dissimilarity values are then used as the input to MDS.

Identifying directions for frequent terms. To discover terms that correspond to interpretable properties in the MDS space, the nouns and adjectives that occur in sufficiently many reviews are used as the input to a linear Support Vector Machine (SVM). The SVM is trained to find the hyperplane that best separates the entities that contain the term at least once in their associated textual description. To accommodate class imbalance, we increase the cost of positive instances such that their weight is inversely proportional to how many times the term has occurred. To assess the quality of the hyperplane found by the SVM, we use Cohen's Kappa score [4] which evaluates how well the hyperplane separates positive/negative instances while taking class imbalance into account. We consider terms with a high Kappa score to be labels of properties that are modelled well by the MDS space. The direction corresponding to a given term/property is given by the vector perpendicular to the associated hyperplane. This vector in turn allows us to determine a ranking of the entities, according to how much they have the property being modelled. This ranking is obtained by determining the orthogonal projection of each entity on an oriented line with that direction. It is easy to see that if v is the vector modelling a given property, then entity e_1 is ranked before entity e_2 iff $e_1 \cdot v < e_2 \cdot v$. Another way to look at this is that entities are ranked according to their signed distance to the hyperplane.

Identifying saleint properties by clustering directions. It can sometimes be ambiguous as to what property each term is referring to. For example, it is unclear whether "mammoth" refers to the animal or an adjective meaning large. In this paper, we have chosen the number of clusters equal to the number of dimensions. To determine the cluster centers, we first select directions whose associated Kappa score is above some threshold T^{+}. We use the highest scoring direction as the center of the first cluster and find the most dissimilar direction to the first cluster's direction to get the centre of the second cluster. Continuing in this way, we repeatedly select the direction which is most dissimilar to all previously selected clusters. By doing so, we obtain a collection of cluster centres that capture a wide variety of different properties from the space. We then associate each remaining direction to its most similar cluster centre. In this step, we consider directions whose associated Kappa score is at least T^{-}, where typically $T^{-} < T^{+}$. Finally, we take the average of all directions in a cluster to be the overall direction for a cluster. The value of T^{+} should be chosen as large as possible (given that the terms with the highest Kappa scores are those which are best represented in the space), while still ensuring that we can avoid choosing cluster centers which are too similar. Choosing the value of T^{-} represents a trade-off. A cluster of terms is often easier to interpret than a single term, which means that we shouldn't choose T^{-} to be too high. On the other hand, choosing T^{-} to be too low would result in poorly modelled terms being added to clusters. For example, we would not want to term "Bee" to be added to the cluster for "Emotional", even though the direction for "Bee" is closest to that cluster.

Note that as each cluster produced by the above procedure is associated with a direction, it induces a ranking of the entities. This gives us two ways to disambiguate which properties a term is referring to: the first being examining which terms it shares its cluster with e.g. we know that “Mammoth” refers to the adjective because it is shared with “Epic”, “Stupendous”, and “Majestic”, and the second being examining which entities score highly in the rankings for a cluster direction e.g. “Monster” defines a ranking in which “Frankenstein” and “The Wolfman” appear among the top ranked movies.

3 Inducing Rules from Entity Embeddings

In this section, we explain how we obtain a series of increasingly general entity embeddings, and how we can learn symbolic rules that link properties from subsequent spaces together.

To construct more general embeddings from the initial embedding provided by the MDS method, we use stacked denoising auto-encoders [16]. Standard auto-encoders are composed of an “encoder” that maps the input representation into a hidden layer, and a “decoder” that aims to recreate the input from the hidden layer. Auto-encoders are normally trained using an objective function that minimizes information loss (e.g. Mean Squared Error) between the input and output layer [2]. The task of recreating the input is made non-trivial by constraining the size of the hidden layer to be smaller than the input layer, forcing the information to be represented using fewer dimensions, or in denoising auto-encoders by corrupting the input with random noise, forcing the auto-encoder to use more general commonalities between the input features. By repeatedly using the hidden layer as input to another auto-encoder, we can obtain increasingly general representations. To obtain the entity representations from our auto-encoders, we use the activations of the neurons in a hidden layer as the coordinates of entities in a new vector space.

The main novelty of our approach is that we characterize the salient properties (i.e. clusters of directions) modelled in one space in terms of salient properties that are modelled in another space. Specifically, we use the off-the-shelf rule learner JRip [7] to predict which entities will be highly ranked, according to a given cluster direction, using as features the rankings induced by the clusters of the preceding space. To improve the readability of the resulting rules, rather than using the precise ranks as input, we aggregate the ranks by percentile, i.e $1\%, 2\%, ..., 100\%$, where an entity has a 1% label if it is among the 1% highest ranked entities, for a given cluster direction. For the class labels, we define a movie as a positive instance if it is among the highest ranked entities (e.g. top 2%) of the considered cluster direction. Using the input features of each layer and the class labels from the subsequent layer, these rules can be used to explain the semantic relationships between properties modelled by different vector spaces. We note that one drawback of discretizing continuous attributes is that the accuracy of the rules extracted from the network may decrease [14]. However, in our setting, interpretability is more important than accuracy, as we do not aim to use these rules for making predictions, but use them only for generating explanations and getting insight into data.

4 Qualitative Evaluation

We base our experiments on the movie review text collection of the 15,000 top scoring movies on IMDB[1] made available by [6]. To collect the terms that are likely to correspond to property names, we collect adjectives and nouns that occur at least 200 times in the movie review data set, collecting 17,840 terms overall. We share terms used for the property names across all spaces.

4.1 Software, Architecture and Settings

To implement the denoising auto-encoders, we use the Keras [3] library. For our SVM implementation, we use scikit-learn [11]. We have made all of the code and data freely available on GitHub[2]. We use a 200 dimensional MDS space from [6] as the input to our stack of auto-encoders. The network is trained using stochastic gradient descent and the mean squared error loss function. For the encoders and decoders, we use the tanh activation function. For the first auto-encoder, we maintain the same size layer as the input. Afterwards, we halve the hidden representation size each time it is used as input to another auto-encoder, and repeat this process three times, giving us four new hidden representations $\{Input : 200, Hidden : 200, 100, 50, 25\}$. We corrupt the input space each time using Gaussian noise with a standard deviation of 0.6. As the lower layers are closer to the bag-of-words representation and are higher dimensional, the Kappa scores are higher in earlier spaces, as it is easier to separate entities. We address this in the clusters by setting the high Kappa score threshold T^+ such that the number of terms we choose from is twice the number of dimensions in the space. Similarly, we set T^- such that 12,000 directions are available to assign to the cluster centres in every space.

4.2 Qualitative Evaluation of Induced Clusters

In Table 1, we illustrate the differences between clusters obtained using standard auto-encoders and denoising auto-encoders. Layer 1 refers to the hidden representation of the first auto-encoder, and Layer 4 refers to the hidden representation of the final auto-encoder. As single labels can lead to ambiguity, in Table 1 we label clusters using the top three highest scoring terms in the cluster. Clusters are arranged from highest to lowest Kappa score.

Both auto-encoders model increasingly general properties, but the properties obtained when using denoising auto-encoder properties are more general. For example, the normal auto-encoder contains properties like “Horror” and “Thriller”, but does not contain more general properties like “Society” and “Relationship”. Further, “Gore” has the most similar properties “Zombie” and “Zombies” in Layer 1, and has the most similar properties of “Budget” and “Effects” in Layer 4. By representing a category of movie where “Budget” and “Effects” are important, the property is more general.

[1] http://www.cs.cf.ac.uk/semanticspaces/

[2] https://github.com/eygrr/RulesFromAuto-encoders

Table 1. A comparison between the first layers and the fourth layers of two different kinds of auto-encoders.

Standard Auto-encoder		**Denoising Auto-encoder**	
Layer 1	**Layer 4**	**Layer 1**	**Layer 4**
horror: terror, horrific	horror: victims, nudity	gore: zombie, zombies	society: view, understand
thriller: thrillers, noir	documentary: perspective, insight	jokes: chuckle, fart	emotional: insight, portrays
comedies: comedy, timing	blood: killing, effects	horror: terror, horrific	stupid: flick, silly
adults: disney, childrens	suspense: mysterious, tense	emotionally: tragic, strength	gore: budget, effects
husband: wife, husbands	thriller: thrillers, cop	gags: zany, parodies	military: war, ship
relationships: intimate, angst	gory: gruesome, zombie	hindi: bollywood, indian	romance: younger, handsome
nudity: naked, gratuitous	beautifully: satisfying, brilliantly	touching: teach, relate	ridiculous: awful, worse
political: politics, nation	emotional: complex, struggle	scary: frightening, terrifying	government: technology, footage
smart: slick, sophisticated	laughed: laughing, loud	documentary: document, narration	awesome: chick, looked
creepy: sinister, atmospheric	charming: delightful, loves	adults: disney, teaches	political: country, documentary
laughed: humorous, offensive	hilarious: funny, parody	laughed: brow, laughter	relationship: relationships, sensitive
adventure: adventures, ship	scares: halloween, slasher	thriller: thrillers, procedural	horror: genre, dark
actions: reaction, innocent	funniest: funnier, gags	cgi: animated, animation	waste: concept, plain
cute: adorable, rom	emotions: respect, relationships	suspense: clues, atmospheric	army: disc, studio
british: england, accent	laugh: mom, crazy	dumb: mindless, car	combat: enemy, weapons
horrible: worse, cheap	filmmaker: approach, artist	political: propaganda, citizens	supporting: office, married
narrative: filmmaker, structure	drama: portrayed, portrayal	witty: delightfully, sarcastic	amazon: bought, copy
digital: dolby, definition	interviews: included, showed	laughing: outrageous, mouthed	study: details, detail
gory: graphic, gruesome	comedic: comedies, humorous	relationships: ensemble, interactions	land: water, super
romantic: handsome, attractive	emotionally: central, relationships	creepy: mysterious, eerie	chemistry, comedies, comedic

4.3 Qualitative Evaluation of Induced Symbolic Rules

Our aim in this work is to derive symbolic rules that can be used to explain the semantic relationships between properties derived from increasingly general entity embeddings. We provide examples of such rules in this section. Since the number of all induced rules is large, here we only show high accuracy rules that cover 200 samples or more. Still, we naturally cannot list even all the accurate rules covering more than 200 samples. Therefore we focus here on the rules which are either interesting in their own right or exhibit interesting properties, strengths or limitations of the proposed approach. The complete list of induced rules is available online from our GitHub repository[3].

For easier readability, we post-process the induced rules. For instance, the following is a rule obtained for the property "Gore" in the third layer of the network shown in the original format produced by JRip:

```
IF scares-L2 <= 6 AND blood-L2 <= 8 AND funniest-L2 >= 22
 => classification=+ (391.0/61.0)
```

In this rule, `scares-L2 <= 6` denotes the condition that the movie is in the top 6% of rankings for the property "`scares`" derived from the hidden representation of the second auto-encoder. We will write such conditions simply as "$\texttt{Scares}_2$". Similarly, a condition such as `funniest-L2 >= 22`, which indicates that the property is not in the top 22%, will be written as $\texttt{NOT Funniest}_2$. In this simpler notation the above rule will look as follows:

$\texttt{IF Scares}_2\texttt{ AND Blood}_2\texttt{ AND NOT Funniest}_2\texttt{ THEN Gore}_3$

This rule demonstrates an interpretable relationship. However, we have observed that the meaning of a rule may not be clear from the property labels that are automatically selected. In such cases, it is beneficial to label them by including the most similar cluster terms. For example, using the cluster terms below we can see that "Flick" relates to "chick-flicks" and that "Amazon" relates to old movies:

$\texttt{IF Flick}_2\texttt{ AND Sexual}_2\texttt{ AND Cheesy}_2\texttt{ AND NOT Amazon}_2\texttt{ THEN Nudity}_3$

$\texttt{Flick}_2\texttt{: \{Flicks, Chick, Hot\}}$
$\texttt{Amazon}_2\texttt{: \{Vhs, Copy, Ago\}}$

Rules derived from later layers use properties described by rules from previous layers. By seeing rules from earlier layers that contain properties in later layers, we can better understand what the components of later rules mean. Below, we have provided rules to explain the origins of components in a later rule:

$\texttt{IF Emotions}_2\texttt{ AND Actions}_2\texttt{ THEN Emotions}_3$
$\texttt{IF Emotions}_2\texttt{ AND Emotion}_2\texttt{ AND Impact}_2\texttt{ THEN Journey}_3$
$\texttt{IF Emotions}_3\texttt{ AND Journey}_3\texttt{ THEN Adventure}_4$

[3] https://github.com/eygrr/RulesFromAuto-encoders

We observe a general trend that as the size of the representations decreases and the entity embeddings become smaller, rules have fewer conditions, resulting in overall higher scoring and more interpretable rules. To illustrate this, we compare rules from an earlier layer to similar rules in a later layer:

```
IF Romance$_1$ AND Poignant$_1$ AND NOT English$_1$ AND NOT French$_1$
 AND NOT Gags$_1$  AND NOT Disc$_1$ THEN Relationships$_2$

IF Relationships$_2$ AND Emotions$_2$ AND Chemistry$_2$ THEN Romantic$_3$
IF Emotions$_2$ AND Compelling$_2$ THEN Beautifully$_3$
IF Warm$_2$ AND Emotions$_2$ THEN Charming$_3$
IF Emotions$_2$ AND Compelling$_2$ THEN Emotional$_3$
```

Rules in later layers also made effective use of a NOT component. Below, we demonstrate some of those rules:

```
IF Touching$_3$ AND Emotions$_3$ AND NOT Unfunny$_3$ THEN  Relationship$_4$
IF Laughs$_3$ AND Laugh$_3$ AND NOT Compelling$_3$ THEN  Stupid$_4$
IF Touching$_3$ AND Social$_3$ AND NOT Slasher$_3$ THEN  Touching$_4$
```

As the same terms were used to find new properties for each space, the obtained rules sometimes use duplicate property names in their components. As the properties from later layers are a combination of properties from earlier layers, the properties in later layers are refinements of the earlier properties, despite having the same term. Below, we provide some examples to illustrate this:

```
IF Emotions$_2$ AND Actions$_2$ THEN Emotions$_3$

Emotions$_2$: {Acted, Feelings, Mature}
Actions$_2$: {Control, Crime, Force}
Emotions$_3$: {Emotion, Issue, Choices}

IF Horror$_2$ AND Creepy$_2$ AND Scares$_2$ THEN Horror$_3$

Horror$_2$: {Terror, Horrific, Exploitation}
Creepy$_2$: {Mysterious, Twisted, Psycho}
Scares$_2$: {Slasher, Supernatural, Halloween}
Horror$_3$: {Creepy, Dark, Chilling}

IF Touching$_2$ AND Chemistry$_2$ THEN Touching$_3$
IF Touching$_2$ AND Emotions$_2$ THEN Touching$_3$
IF Compelling$_2$ AND Emotional$_2$ AND Suspense$_2$ THEN Compelling$_3$
If Romance$_2$ AND Touching$_2$ AND Chemistry$_2$ THEN Romance $_3$
IF Emotionally$_2$ AND Emotions$_2$ AND Compelling$_2$ THEN Emotionally$_3$
```

5 Related Work

The work presented in this paper differs from existing works in that it focuses on inducing rules which involve salient and interpretable features from unstructured text documents.

The existing neural network rule extraction algorithms can be categorized as either decompositional, pedagogical or eclectic [1]. Decompositional approaches derive rules by analysing the units of the network, while pedagogical approaches treat the network as a black box, and examine the global relationships between inputs and outputs. Eclectic approaches use elements of both decompositional and pedagogical approaches. Our method could be classified as decompositional, as we make use of the hidden layer of an auto-encoder. We will now describe some similar approaches and explain how our methods differs.

The algorithm in [10] is a decompositional approach that applies to a neural network with two hidden layers. It uses hyperplanes based on the weight parameters of the first layer, and then combines them into a decision tree. NeuroLinear [15] is a decompositional approach applied to a neural network with a single hidden layer that discretizes hidden unit activation values and uses a hyperplane rule to represent the relationship between the discretized values and the first layer's weights. HYPINV [13] is a pedagogical approach that calculates changes to the input of the network to find hyperplane rules that explain how the network functions.

The main difference in our work is that our method induces rules from properties derived from the layers of a network, rather than learning rules that describe the relationships between units in the network itself. Additionally, we focus on learning increasingly general entity embeddings from hidden representations rather than tuning network parameters such that weights directly relate to good rules.

Another recent topic that relates to our work is improving neural networks and entity embeddings using symbolic rules [8]. In [12] a combination of first-order logic formulae and matrix factorization is used to capture semantic relationships between concepts that were not in the original text. This results in relations that are able to generalize well from input data.

This is essentially the opposite of the task we consider in this paper: using embeddings to learn better rules. The rules that we derive are not intended to explain how the network functions but rather to describe the semantic relationships that hold in the considered domain. In other words, our aim is to use the neural network representations in the hidden layer as a tool for learning logical domain theories, where the focus is on producing rules that capture meaningful semantic relationships.

6 Conclusions

In this paper, we have shown how we can obtain increasingly general entity embeddings from stacked denoising auto-encoders, and how we can obtain rules from those embeddings that capture domain knowledge. We have qualitatively evaluated the obtained rules to demonstrate the semantic relationships that they capture. The results show the potential of the method for exploratory analysis of collections of unstructured text documents and explaining decisions of classifiers.

Acknowledgement. This work was supported by ERC Starting Grant 637277.

References

1. R. Andrews, J. Diederich, and A. B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6):373–389, 1995.
2. Y. Bengio. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009.
3. F. Chollet. Keras. https://github.com/fchollet/keras, 2015.
4. J. Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37, 1960.
5. J. Derrac and S. Schockaert. Enriching taxonomies of place types using Flickr. *Lecture Notes in Computer Science*, 8367:174–192, 2014.
6. J. Derrac and S. Schockaert. Inducing semantic relations from conceptual spaces: A data-driven approach to plausible reasoning. *Artificial Intelligence*, 228:66–94, 2015.
7. J. L. Herlocker, J. a. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. *Proceedings of the ACM conference on Computer supported cooperative work*, pages 241–250, 2000.
8. Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing. Harnessing Deep Neural Networks with Logic Rules. *arXiv preprint*, pages 1–18, 2016.
9. W. B. Kheder, D. Matrouf, P.-M. Bousquet, J.-F. Bonastre, and M. Ajili. Statistical Language and Speech Processing. *Statistical Language and Speech Processing*, 8791:97–107, 2014.
10. D. Kim and J. Lee. Handling continuous-valued attributes in decision tree with neural network modeling. 1810:211–219, 2000.
11. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
12. T. Rocktäschel, S. Singh, and S. Riedel. Injecting logical background knowledge into embeddings for relation extraction. *Proceedings of the 2015 Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, 2015.
13. E. W. Saad and D. C. Wunsch. Neural network explanation using inversion. *Neural Networks*, 20(1):78–93, 2007.
14. R. Setiono, B. Baesens, and C. Mues. Recursive neural network rule extraction for data with mixed attributes. *IEEE Transactions on Neural Networks*, 19(2):299–307, 2008.
15. R. Setiono and H. Liu. Neurolinear: From neural networks to oblique decision rules. *Neurocomputing*, 17(1):1–24, 1997.

16. P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

Shared Multi-Space Representation for Neural-Symbolic Reasoning

Edjard de S. Mota and Yan B. Diniz

Federal University of Amazonas
Institute of Computing
Av. Rodrigo Octávio, 6200 CEP 69077-000 Manaus, Brasil
{edjard,ybd}@icomp.ufam.edu.br

Abstract. This paper presents a new neural-symbolic reasoning approach based on a sharing of neural multi-space representation for coded fractions of first-order logic. A multi-space is the union of spaces with different dimensions, each one for a different set of distinct features. In our case, we model the distinct aspects of logical formulae as separated spaces attached with vectors of importance weights of distinct sizes. This representation is our approach to tackle the neural network's propositional fixation that has defied the community to obtain robust and sound neural-symbolic learning and reasoning, but presenting practical useful performance. Expecting to achieve better results, we innovated the neuron structure by allowing one neuron to have more than one output, making it possible to share influences while propagating them across many neural spaces. Similarity measure between symbol code indexes defines the neighborhood of a neuron, and learning happens through unification which propagates the weights. Such propagation represents the substitution of variables across the clauses involved, reflecting the resolution principle. In this way, the network will learn about patterns of refutation, reducing the search space by identifying a region containing ground clauses with the same logical importance.

1 Introduction

This paper presents a new neural-symbolic reasoning approach based on neural sharing of multi-space representation for a coded portion of first-order formulae suitable for machine learning and neural network methods. The Smarandache multi-space [9] is a union of spaces with different dimensions, each one representing a different set of distinct features. We distribute across such a structure the different aspects of logical expressions along with vectors of weights of distinct sizes. With such a representation one can compute the *degree of importance*, that is *induced* by the resolution principle and unification across distinct dimensions of the logical structure during a deduction [12], taking such spaces into account.

There have been some efforts to deal with the neural network's propositional fixation [10], since it was argued in [4] that for some fragments of first-order logics such a limitation can be overcome, for instance [1, 7]. However, their attempt to provide robust and sound neural-symbolic learning and reasoning were unsuccessful, as they all lack practical useful performance [3], defying us to tackle this issue from a different perspective. Looking at Amao[1] structure sharing-based

[1] A cognitive agent we are developing at the Intelligent and Autonomous Computing group at IComp in UFAM

implementation [13, 2], as in most Prolog engines, we felt like transforming them into a structure sharing of code indexes and use it for neural learning computation.

Automated deduction based on Resolution Principle [12], reduces the search space by transforming the task of proving the validity of a formula to prove that its negation is inconsistent. The main struggle with doing first-order logic reasoning in connectionist approaches is that the variable binding of terms may lead to a huge, if not infinite, number of neurons for all elements of the Herbrand Base. We realized that, instead of doing this, neural reasoning could actually *points* to "neural regions" where the negation of a given formula were most likely to be inconsistent. The difference would be the use of a structured neural network trained to learn about regions of potential refutations before one is even requested. This is only possible if the network learns from the initial set of formulae and self-organize in regions of refutation.

In this paper, we introduce the *Shared Neural Multi-Space* (Shared NeMuS) of coded first-order expressions (CFOE), a weighted multi-space of CFOEs. The idea is to give a relative *degree of importance* for each element within it according to the element attributes and similarity with others structurally equivalent. Similarity defines the neighborhood of an element and neural learning is performed by the propagation of weights through unification. Such propagation represents the substitution of variables across the clauses involved, reflecting resolution principle for first-order logic[12]. In this way the network will learn about patterns of refutation to reduce the search space when queries are proposed.

Before describing the formalities of our approach, section 2 shows the fundamental aspects of the neural shared multi-spaces of CFOEs. In a Shared NeMuS one neuron represents logical expression and it may have many inputs of importance as well as outputs that influence others. We formally present the shared NeMuS for CFOEs in section 3 to capture the fundamentals described. In section 4 we detail the mechanisms to train such a structured neural net based on an adapted best-match similarity measure for learning patterns of resolution-based deduction. This innovative way of creating a structured neural network, shared NeMuS, may not fit in the standards of the machine learning field as discussed in section 5. Nonetheless, such a perspective can bring new light to the way neural-symbolic learning and reasoning is performed for first-order logic as we discuss in section 6.

2 Fundamentals of Neural Sharing of Multi-Space

We use Smarandache multi-space [9], which is a union of n spaces $A_1, \ldots, A_n$ in which each A_i is the space of a distinct observed characteristic of the overall space. For each A_i there is a different metric to describe a different side (or objetive), of the "major" side (or objective). In this perspective, first-order language has *atomic constants* (of the Herbrand universe), *function*, *predicate* with its literal instances, and *clause* spaces. Variables are used to refer sets of atomic terms via quantification, and they belong to the same space of atoms. Figure 1.(a) depicts a multi-space representation of first-order expressions with n clauses, at space 3, each one defined by a (possibly different) number of literals at space 2. Each literal is composed of terms either from function space 1 or constant space 0, or both. Lines from one element covers its compound terms at the space below.

The neural network embedded within such a multi-space is based on a chain of importance weights, having constant space as the basic level of importance. In their turn, weights of the constant space induce the importance weights of functions space, and both (constant and function) spaces induces weights of the predicate space according to literal instances within it. Finally, weights of predicate space induces clauses importance weights. Figure 1.(b) depicts the neural multi-space of FOEs, in which weights are the (blue) arrows representing the influence of attributes from one space on objects at one or two space above them.

Different from traditional Artificial Neural Networks (ANN), one single neuron may have, along with its inputs (weights of influence), more than one output representing its influence upon more than one element at a space level above. From Figure 1.(b) constant a_3 affects literals l_1 and l_2 of clause C_1, and it affects l_1 of clause C_n. Note that there are two l_1 logical objects, but if both are positive/negative instances of the same predicate, then there should be just one neuron representation in this case rather then replicating information.

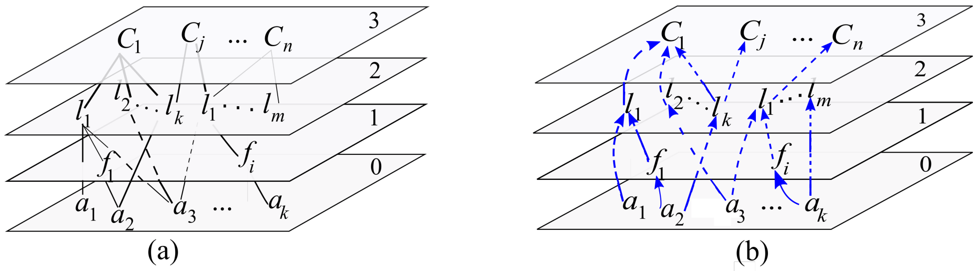


Fig. 1. (a) A general sharing multi-space of FOEs. (b) A neural sharing multi-space of FOEs

To avoid such repetition, we adopted the sharing of structure idea [13]: every logical neural element is a pair. The first component is the neuron symbol–attribute pair, formed by symbol code and a vector of indexes with the space each one belongs to. The second component is a vector of structured weights pointing to the elements the neuron exerts influence. A structured weight neuron is, in the case of constant neural space, a triple: the space index (0 up to 3), the code index of the symbol upon which it influences, and the value of the influence. A triple is used because atoms at level 0 can be attributes of functions (at level 1) or of a literal (at level 2), e.g. constant a_2, and most import is to tell the influence of a term on a function from a literal, like a_2 does on function f_1 as well as on literal l_k. For all other spaces, a structured weight is pair because, from space 1, every neuron will exert influence only on neurons at one level above.

When a shared NeMuS of FOE is generated all weight vectors of its components, at all levels, are set to zero to represent no previous learning. Then training is divided into two phases. In the first, every ground clause (clause with no variables), has its weights updated according to the code of its symbol components. This will create the importance of clauses, expressed in weights. Then, clauses that had their weights updated will propagate them via similarity of the predicate space, yielding regions associated to such similarity measure.

In the second phase, every clause with more than one literal and at least one variable, called *deduction rule*, is divided into two parts: conclusion and assumptions. For each assumption p of a deduction rule, with an index code i_p, a sort of neural unification is applied between p and its complementary literal with same index, if there is any, at the negative region of predicate space. The premisses with successful unification will update their weights from their components, and the weights of the conclusion will be updated by the weights of the shared variables. In the case of functions, not only variable weights are updated, but the composition of predicate and variable weights will update the weights of functions.

3 NeMuS Framework for Coded First-order Expressions

3.1 Amao Logical Language

Amao[2] symbolic representation and reasoning component is a clausal-based formal system [14], in which clauses are divided into two categories. 1) *Initial Clauses*, say $\mathtt{B}$, are those belonging to the set of axioms plus the negation of the query; 2) *structured Clauses* are the ones derived by a sort of Linear Resolution [12]. Roughly, if $\mathtt{S}$ is a sentence or query, in clausal form, and $\mathtt{B}$ is the set of initial clauses, then a deduction of $\mathtt{S}$ from $\mathtt{B}$ corresponds to derive an empty clause, $\sqcup$, from $\{\sim \mathtt{S}\} \cup \mathtt{B}$, or according to Herbrand theorem, to prove that $\{\sim \mathtt{S}\} \cup \mathtt{B}$ is *unsatisfiable* and it yields the most general unifier for $\mathtt{S}$.

A set of logical formulae is represented by clauses of literals according to the following terminology. Predicates and constant or atomic symbols start with lowercase letters like $p, q, r, \dots$ and $a, b, c, \dots$, respectively. Variables start with capital letters, like $X, Y, \dots$. A term is either a variable, a constant symbol or a function $f(t_1, \dots, t_k)$ in which f represents a mapping from terms $t_1, \dots, t_k$ to an "unknown" individual. If p is a symbol representing a predicate relation over the terms $t_1, \dots t_n$, then $p(t_1, \dots, t_n)$ is a valid atomic formula. Predicates and functions are compound symbols with similar structure, but with different logical meaning. A *literal* is either an atomic formula, L, or its negation $\sim L$, and both are said to be complementary to each other. A *Deduction Rule* is a disjunction of literals $\mathtt{L}_1$, ..., $\mathtt{L}_n$, written as $\mathtt{L}_1; \dots; \mathtt{L}_1$. There may exist more than one positive literal, an so any Horn clause is represented by $Head; \sim body$, in which literals of the body are called assumptions.

Example 1. The following is a valid sequence of clauses, each with its unique index code.

$$\begin{array}{lll} 1.\ p(a). & 3.\ r(a). & 5.\ q(X, f(X))\ ;\ \sim p(X) \\ 2.\ p(b). & 4.\ r(c). & 6.\ s(X, f(Y))\ ;\ \sim r(X); \sim p(Y) \end{array}$$

3.2 First-Order Expressions as Multi-Spaces

Amao symbolic reasoning component parses and translates a sequence of clauses into an internal structure of shared of data connected via memory address pointers. This representation is very efficient for dealing with symbols, and the idea of sharing data could be used to create computational efficient neural representations of clauses. Formal logic languages are structurally well defined, and such a structure can be thought as a structure of indexes. Instead of training a neural network with bare data like other approaches, e.g. [7], we decided to use an efficient encoding of shared structures, and turn them into spaces of index to build up a first-order neuronal multi-space.

For this purpose, Amao makes use of a symbolic hash mapping [8] (SHM), that maps symbolic objects of the language to a hash key within a finite range. Such a key is not the one used for

[2] Amao is the name of a deity that taught people of Camanaos tribe, who lived on the margins of the Negro River in the Brazilian part of the Amazon rainforest, the process of making *mandioca powder* and *beiju* biscuit for their diet.

learning because there may occur collisions. For this reason a separate chaining is used to place keys that collide in a list associated with index, in which every node contains the kind of occurred symbol. Counters were added so that to every new symbol parsed and "hashed", a code hash mapping (CHM) function generates the next natural number, starting from 1. In this way, every single symbol has a unique index, and such an index shall be the one used for neural learning mechanism. All codes compose what we call coded corpus defined as follows.

Definition 1 (First-order Coded Corpus (FOCC)) *Let $\mathcal{C}$, $\mathcal{F}$ and $\mathcal{P}$ be a finite sets of constants, functions and predicates, respectively. The* First-order Coded Corpus *is a triple of associative hash mappings $\langle f_C, f_F, f_P \rangle$, such that $f_C : C \to \mathbb{N}$, $f_F : F \to \mathbb{N}$ and $f_P : P \to \mathbb{N}$. The mappings f_F and f_P take into account the arity of each function and predicate, to generate their indexes $n \in \mathbb{N}$. Each element of a FOCC triple $\mathcal{C}$ shall be identified as $\mathcal{C}_C$, $\mathcal{C}_F$ and $\mathcal{C}_P$.*

Note that the uniqueness of a mapping is only within a corpus space, i.e. the code "1" will be the index of the first predicate found, as well as the first atom four in the case of formula $p(a)$ be the first clause parsed. Figure 2 depicts a possible FOCC generated from clauses of example 1.

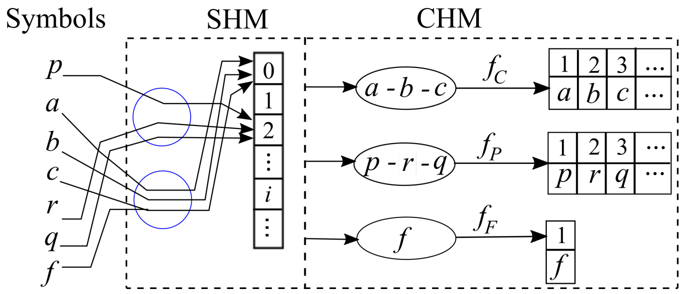


Fig. 2. First-order coded corpus of logical symbols from Example 1.

The result of parsing of any logical formula is passed to the corpus generation, which is also fed with variable indexes according to their clause scope. From a reasoning perspective, variables can be interpreted as an abstract way to talk about sets of atomic constants. For efficiency sake it is assumed that both belong to two different regions of the same space: positive region for constant symbols and negative for variable appearing in all clauses. The scope of each variable is bound via weights. The following two definitions capture these idea, in which $\mathbb{Z}_0$ means $\mathbb{Z} \setminus \{0\}$.

Definition 2 (Subject Binding) *Let $k \in \mathbb{N}$ be an index, $h \in \{1, 2\}$, $i \in \mathbb{Z}_0$ and $w \in \mathbb{R}$. The* Subject Binding *of k is the triple (h, i, w), and it represents that subject with index k* influences *object with index i at space h with measure w.*

The spaces a subject may influence are the function space (1) and the predicate space (2) (see Figure 1). As said above variables can be seen as a way to refer to sets of constants, either atoms or (mostly ground) functions. To be identified outside the subject space a variable shall always be a negative number, but its influence or subject binding will be accessed by its absolute value from the variable region of the subject space.

Definition 3 (Neural Subject Multi-Space (NeSuMS)) *Let* $C_\beta = [x_1, \ldots, x_m]$ *and* $V_\beta = [y_1, \ldots, y_n]$*, where each* $x_i(y_i)$ *is a vector of subject binding, be two subject binding spaces for constants and variables, respectively. A* Neural Subject Multi-Space *is the pair* (C_β, V_β)*.*

Functions and predicates have a different sort of binding, or importance, along with the information about their attributes. As they are both structurally alike, they are treated in the same way regarding their composition. Their binding is simpler than subject binding because they just need the logical element index at the space above and the value of such influence. In both cases, their attributes are uniquely identified by space, either zero (for subject space) or one (for function space), and the attribute index. In the case of space 0, if attribute index is less than zero this means that it refers to the variable region.

Definition 4 (Neural Compound Multi-Space (NeComMS)) *Let* $h_1, \ldots, h_m \in \{0, 1\}$ *be space indexes (for variable and function, i.e. 0 or 1),* $a_1, \ldots, a_m \in \mathbb{Z}_0$*,* $\overrightarrow{x_a^i} = [(h_1, a_1), \ldots, (h_m, a_m)]$ *a vector of pairs space-index of compound i,* $w_1, \ldots, w_n$ *are vectors of* Compound Binds $w \in \mathbb{Z}_0 \times \mathbb{R}$*. Then a* Neural Compound Multi-Space*, with k compounds, will be* $[(\overrightarrow{x_a^1}, \overrightarrow{\omega_1}), \ldots, (\overrightarrow{x_a^k}, \overrightarrow{\omega_k})]$*, in which every* $\overrightarrow{x_a^i}$ *may have a different size m as well as every* $\overrightarrow{\omega_i}$*, and* $i = 1 \ldots k$ *.*

Predicates, a part from the symbols uniquely indexed in the corpus, have their positive and negative occurrences, and so there will be two regions for predicate space too. This is one of the difference between predicates and functions, the other is their logical value. So, the spaces for them are defined as follows

Definition 5 (Function and Predicate Neural Multi-Spaces) *Let* C_f *,* C_p^+ *and* C_p^- *be NeComMS, such that* C_f *has index space one (1), and* C_p^+ *and* C_p^- *have both index space two(2). Then* C_f *is called a* Function Neural Multi-Space*, and every* $\overrightarrow{\omega}$ *appearing in* C_f *represents a vector of influences upon elements of space two (2). The pair* (C_p^+, C_p^-) *is called a* Predicate Neural Multi-Space *(LMS), in which every* $\overrightarrow{\omega}$ *appearing on both represent a vector of influences upon elements of space three (3) of clause.*

Clause spaces are simpler than compound spaces (functions and predicates) because clauses have "attributes" (their literals), but exert no influence upon spaces above, at least for the scope out our current research. One may think in terms of non-classical logics as adding other spaces composed of clauses that influence them. A clause is just an special case of a compound MS in which every weight vector has just one dimension pair $(\_, 0)$, where the _ symbol represents an anonymous logical object, and 0 represents no known influence to above spaces. The attributes will represent the literals that compose the clause.

Definition 6 (Neural Multi-Space of Clauses) *Let* $k_1, \ldots, k_m \in \mathbb{Z}_0$ *be predicate index codes, a* Neural Clause *at clause multi-space is* $C = ([(2, k_1), \ldots, (2, k_m)], [(\_, 0)])$*. A* Neural Multi-Space of Clauses *is simply* $[C_1, \ldots, C_n]$ *in which every* $C_i, i = 1..n$*, is a neural clause.*

Definition 7 (Shared NeMuS of CFOE) *Let* $\mathcal{S}$*,* $\mathcal{F}$*,* $\mathcal{P}$ *and* $\mathcal{C}$ *be a subject, function, predicate and clause neuronal multi-spaces. Then, we call a* Shared Neural Multi-Space of CFOE *to the ordered space* $\langle \mathcal{S}, \mathcal{F}, \mathcal{P}, \mathcal{C} \rangle$

4 Amao Learning Mechanism

In this section we present shared NeMuS learning process that is based on Kohonen [6] Self-Organizing Maps (SOM), although any learning mechanism could be used. Because shared NeMuS is not a standard matrix as in vector-spaces, distance measures are performed in different ways as it shall be clear in the sequel. The SOM training phase calculates the euclidean distance from the input vector to every neuron on the map. After that, it searches for the best match unity and updates the weights of every neuron in the neighborhood. The neighborhood of a single clause is defined by the index of a predicate. The following equation is used to update the weight vector:

$$\overrightarrow{\omega}_{(t+1)} = \overrightarrow{\omega}_{(t)} + \eta(\overrightarrow{\omega}_I - \overrightarrow{\omega}_{(t)}) \tag{1}$$

in which η is the learning rate, $\overrightarrow{\omega}_{(t)}$ and $\overrightarrow{\omega}_I$ are a multi-space vectors of weights, and t represents the epoch of interaction. We adapted the best match unity $\overrightarrow{\omega}_{bm}$ for our purposes, making it possible to apply resolution on clauses with complementary literals. In NeMuS this is easily obtained because the representation of any literal is its predicate index code in the positive region of the predicate space, and its complementary literal should have same index in the negative region, so the access is of complexity $O(1)$.

Training steps

This phase starts after the shared NeMuS structure had just been created from the compilation of the symbolic KB.The input for training is the KB itself and the steps are divided into two parts: one to deal with ground atomic formulae and the other deals with formulae with variables, henceforth called *deduction rules* (defined in section 3.1). Let $\overrightarrow{\omega}_I$ be an arbitrary input where its weights are represented by a CFOE, and $N_{KB} = \langle \mathcal{S}, \mathcal{F}, \mathcal{P}, \mathcal{C} \rangle$ a shared NeMuS.

Algorithm 1 Chain training

```
 1: for every clause $C \in \mathcal{C}$ do
 2:     if $C$ has just one literal then                                  ▷ (Process of ground atoms.)
 3:         for $k \in C$ a index for predicate codes do
 4:             $\overrightarrow{\omega}_{(t+1)} = [\omega_C, \omega_k, \omega_1 ..., \omega_m] + \eta(\overrightarrow{w_I} - [\omega_C, \omega_k, \omega_1 ..., \omega_m])$
 5:     if $C$ has more than one literal then                             ▷ (Process of deduction rules.)
 6:         for $k$ a index for predicate codes $\in C$ do
 7:             if $k > 0$ then
 8:                 for $f$ a function attribute of predicate with code $k$ do
 9:                     for $v$ a variable $\in f$ do
10:                         $\{\omega_C, \omega_k\} = \{\omega_k, \omega_f\} = \{\omega_{bm}, \omega_{bm}\}$, for every function, or literal in clause $C$.
11:             else
12:                 for a variable attribute $v$ from predicate with code $k$ do
13:                     $\{\omega_C, \omega_k\} = \{\omega_{bm}, \omega_{bm}\}$, for every literal $\in C$.
```

Refutation Pattern Learning Mechanism

The refutation pattern learning mechanism of Amao, called *NeMuS NeuraLogic Reasoner*, will try to find one refutation pattern for the input vector, has two important tasks that defines what was learned.

1. to recognize the refutation for a query (deduction rule inference with no premisses), it just needs to identify the region within its trained shared NeMuS for which all variable can be assigned a value.
2. to recognize the refutation for a ground formulae with more than one literal, it just need to compare the weight values of the input with the region indicated in the training phase, and if it is different, the answer is $false$, otherwise $true$.

Running Experiment on Refutation Pattern

The first test shows classical *Modus Ponens* reasoning with deduction rules having no restriction on the number of variables, and also when we have one level function. Using the NeMuS training on knowledge base presented in Example 1, we obtained these weights:

Symbolic Representation	Neural Representation
1. p(a).	(1.44, 0.84, 1.44)
2. p(b).	(1.44 0.84, 1.44)
3. r(a).	(3.12, 1.68, 2.04)
4. r(c).	(3.12, 1.68, 2.04)
5. q(X, f(X)); ∼p(X).	(1.04, 0.84, 1.44, 0.84, 1.44)
6. s(X, fY));∼r(X); ∼p(X).	(1.52, 1.68, 2.04, 0.84, 1.44, 1.68, 2.04, 0.84, 1.44)

Table 1. A NeMuS net trained

There are two important things to consider regarding the test results. The first is the translation of the symbolic input (query) into a NeMuS format with its vector of input weights $\overrightarrow{\omega}_I$. Second, identify the region this NeMuS object is most likely to belong. Furthermore, there must be a "kind of relation" between the input and the region which best matches it. For this Amao NeuraLogic reasoner creates a relation between region and the input. On the following table we present a best match selection from a single proof:

Proof $p(X)$:

1. Converting to $\overrightarrow{w_I}$: $\{1.44, 0.84, 0\}$
   The conversion of X is 0, because it is not in p.
2. Search for best match:

Distance $p(X) \leftrightarrow p(a)$	1.44
Distance $p(X) \leftrightarrow p(b)$	1.44
Distance $p(X) \leftrightarrow r(a)$	2.20617
Distance $p(X) \leftrightarrow r(c)$	2.20617
Distance $p(X) \leftrightarrow q(X, f(X))$	2.76261
Distance $p(X) \leftrightarrow q(X, f(Y))$	4.17248

With the information about the distance, NeuraLogic reasoner can define a relation between input and the best match solution. With the shortest distance 1.44, X can assume two values, $\{X/a\}$ and $\{X/b\}$. Now we are going to force a true and false for proposition, asking for:

Proof $s(a, f(c))$:

1. Converting to $\overrightarrow{\omega}_I : \{1.68,\ 0,\ 0.84,\ 0, 1.68,\ 2.04,\ 0.84,\ 1.44\}$
   From the translation we know that exist $r(a)$ and $p(c)$, and so their values are not 0. However, as it is not known whether there is a $s(a, f(c))$, their values for that positions are 0.
2. Search for best match give us: 2.49704.
   So now the shared NeMuS learn that 2.49704 is true.

Proof $s(c, f(b))$:

1. Converting to $\overrightarrow{\omega}_I : \{1.68,\ 0,\ 0.84,\ 0, 1.68,\ 0,\ 0.84,\ 0\}$
2. Search for best match give us: 3.53135
   With this shared NeMuS knows the maximum distance for true is 2.49704 and the answer is so far, that it's false.

Example 2. This example shows how first-order inductive learning can be easily dealt when recursive deduction rules are defined. For instance, to find a path on a graph can be simply defined with this knowledge base.

$$\begin{array}{lll} 1.\ link(a,b). & 3.\ link(c,d). & 5.\ path(XY)\ ;\sim link(X,Y) \\ 2.\ link(b,c). & 4.\ link(d,e). & 6.\ path(X,Y)\ ;\sim link(X,Z);\sim path(Z,Y). \end{array}$$

After knowledge base be represented it's possible to do training process:

Symbolic Representation	Neural Representation
1. link(a,b).	(3.35, 0.974, 1.44, 1.44)
2. link(b,c).	(3.35, 0.974, 2.28, 2.28)
3. link(c,d).	(3.35, 0.974, 3.12, 3.12)
4. link(d,e).	(3.35, 0.974, 3.96, 3.96)
5. path(X, Y);∼link(X, Y).	(4.89, 0.974, 3.39, 3.39, 0.974, 3.39, 3.39)
6. path(X,Y);∼link(X, Z);∼path(Z, Y).	(4.89, 0.974, 3.39, 3.39, 0.974, 3.39, 3.39, 0.974, 3.39, 3.39)

Table 2. Trained base of path between links problem.

Notice that there is a recursive rule, when $X \neq Y$ on clause 5, a value for Z is necessary on 6. So this search goes on until $link(X, Y)$ is true, or no path from X *to* Y is found. Our proposition is to give such a responsibility to NeuraLogic reasoner to perform an iterative process to verify if there is a path from X *to* Y by checking region weights. This is described in the following process to deal with **path(X, Y)**.

```
For $i$ a weight $\in \mathcal{P}$
  - If there's a index CFOE with $\overrightarrow{\omega}_i$ and $\overrightarrow{\omega}_Y$ with the same weight, answer true.
  - Else
```

- If there's a index CFOE with weight $\overrightarrow{\omega}_i$ and $\overrightarrow{\omega}_X$ with the same weight
  $\overrightarrow{\omega}_X \leftarrow \overrightarrow{\omega}_i$
- Else the answer is false.

For now we can not avoid this iterative process to express a recursive execution, so NeuraLogic reasoner have only to give the right answer when it is asked for a link.

5 Related Work

Developing robust and sound, yet efficient, neural-symbolic learning and reasoning is the ultimate goal of the marriage between neural networks and symbolic (logical) reasoning[3]. The approach presented in this paper falls in the category of the ones pursuing for a feasible representation to overcome John McCarthy's claim that connectionist systems have propositional fixation[10], but which provides a feasible implementation to achieve useful performance.

Some recent approaches that sought to overcome this issue have proposed frameworks to allow expressive representation of complex nesting of symbols in first-order formulae. Komendantskaya proposed unification neural network [7], to allow first-order connectionist deduction. Practical results were not proven to be easily achieved for arbitrary first-order formulae having a (potential) infinite number of symbols. The proposed CFOE representation (section 3.2) has no such limit, and the sharing of neural CFOE makes the access of any neuron of $O(1)$ complexity in any case, while saving storage space. This is also an advantage when compared to Pinkas, Lima and Cohen, [11], who designed pools (tables) for symbols to allow the nesting of bindings and to keep track of unification. Despite the claimed efficiency when compared to the former, the pools are actually matrices representing directed acyclic graph. Sets of formulae with different numbers of terms and literal would generate sparse matrices compromising the complexity of the algorithms for learning and reasoning.

Guillame-Bert, Broda and Garcez, [5], encoded first-order formulae as vectors of real number from Cantor set aiming to provide neural-symbolic inductive learning about first-order rules. The type restriction on terms, but not on sub-terms, weakened the claimed expressive power. The generation of codes for large sets of first-order sentences may have an impact on the efficiency of the training process. Besides, our approach does not suffer the type restriction since it is already based on a multi-space concept where every logical symbol e well placed in its appropriate space.

6 Concluding Remarks

In this paper we presented a novel approach for neural-symbolic learning and reasoning of first-order logic. Our main purpose was to create a neural model that we could characterize *patterns of proof by refutation*, based on the resolution principle with unification for first order inference. There were two well known challenges to be tackled in or der to achieve this general and ambitious goal: to overcome the propositional fixation and a neural network architecture that could allow efficient computations. This means, Amao should perform reasoning faster than symbolic approaches as it should take advantage of having learned something about the domain.

These challenge were dealt with a little ingenuity of the shared NeMuS (Neural Multi-Space approach), which combines Smarandache multi-space modeling technique with sharing of structure concept from Boyer-Moore efficient implementation of Prolog engines. By separating in spaces constants and variables, functions, predicates (literals) and clauses, we treated each of this logical objects as a type since each has specific computations for the overall neural computation of learning and reasoning.

Our main contribution was to show, like in Example 2 (in the end of section 4), that first-order neural-symbolic reasoning does not need to compute the entire Herbrand base (i.e. the set of ground atomic formulae). Amao used its trained shared NeMuS to iterate over the regions of similar ground atomic formulae and efficiently find a refutation or say the query does not follow from what it has learned. However, some interesting challenges remain to be tackled and we point some here.

– recursive deduction rules generating a potentially infinite number of ground terms, e.g. $s(s(s(\ldots)))$, were not tested. Although Amao is not likely to deal with it, another space orthogonal to all others seem to be one solution to deal with recursive loops on functions.
– a part from induction inference by recursive rules, which other kinds of deduction pattern can a self-trained NeMuS recognize?

References

1. Bader, S., Hitzler, P., Hölldlber, S.: Connectionist model generation: A first-order approach. Neurocomputing 1(71), 2420–2432 (2008)
2. van Emden, M.H.: An interpreting algorithm for Prolog programs, Ellis Horwood Series Artificial Intelligence, vol. 1, chap. 2, pp. 93–110. Ellis Horwood (1984)
3. d'A. Garcez, A., Besold, T.R., de Raedt, L., Földiak, P., Hitzler, P., Icard, T., Kühnberger, K.U., Lamb, L.C., Miikkkulainen, R., Silver, D.L.: Neural-symbolic learning and reasoning: Contributins and challenges. In: AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches - Dagstuhl (2014)
4. d'Avila Garcez, A.S., Broda, K., Gabbay, D.: Neural-Symbolic Learning Systems: Foundations and Applications, Perspectives in Neural Computing. Springer-Verlag (2002)
5. Guillame-Bert, M., Broda, K., d'Avila Garcez, A.: First-order logic learning in artificial neural networks. In: International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2010)
6. Kohonen, T.: Self-Organizing Maps. Springer, 3rd edn. (2001)
7. Komendantskaya, E.: Unification neural networks: unification by error-correction learning. Logic Journal of the IGPL 19(6), 821–847 (May 2010)
8. Konheim, A.G.: Hashing in Computer Science: Fifty Years of Slicing and Dicing. John Wiley & Sons (2010)
9. Mao, L.: An introduction to smarandache multi-spaces and mathematical combinatorics. Scientia Magna 3(1), 54–80 (2007)
10. McMCarthy, J.: Epistemological challenges for connectionism. Behavioral and Brain Sciences 11(1), 11–44 (1988)
11. Pinkas, G., Lima, P., Cohen, S.: Representing, binding, retrieving and unifying relatinal knowledge using pools of neural binders. Elsevier Biologically Inspired Cognitve Architectures 1(6), 87–95 (2013)
12. Robinson, A.: A machine-oriented logic based on the resolution principle. Journal of the ACM 12(1), 23–42 (1965)
13. R.S. Boyer, J.M.: The sharing of structure in theorem-proving programs. In: Bernadrd Meltzer, D.M. (ed.) Annual Machine Intelligence. vol. 7, pp. 101–116. Edinburgh University Press (1972)
14. Vieira, N.: Máquinas de Inferência para Sistemas Baseados em Conhecimento. Ph.D. thesis, Pontifícia Universidade Católica do Rio de Janeiro (1987), phD Thesis

Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge[⋆]

Luciano Serafini[1] and Artur d'Avila Garcez[2]

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

Abstract. We propose *Logic Tensor Networks*: a uniform framework for integrating automatic learning and reasoning. A logic formalism called Real Logic is defined on a first-order language whereby formulas have truth-value in the interval [0,1] and semantics defined concretely on the domain of real numbers. Logical constants are interpreted as feature vectors of real numbers. Real Logic promotes a well-founded integration of deductive reasoning on a knowledge-base and efficient data-driven relational machine learning. We show how Real Logic can be implemented in deep Tensor Neural Networks with the use of Google's TENSORFLOW™ primitives. The paper concludes with experiments applying Logic Tensor Networks 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 recent availability of large-scale data combining multiple data modalities, such as image, text, audio and sensor data, has opened up various research and commercial opportunities, underpinned by machine learning methods and techniques [5, 12, 17, 18]. 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. We argue that there are great possibilities for improving the current state of the art in machine learning and artificial intelligence (AI) thought the principled combination of knowledge representation, reasoning and learning. Guha's recent position paper [15] is a case in point, as 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 [8, 10, 11] and in semantic attachment and symbol grounding [4] to achieve a vector-based representation which can be shown adequate for integrating machine learning and reasoning in a principled way.

[⋆] 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.

This paper proposes a framework called *Logic Tensor Networks (LTN)* which integrates learning based on tensor networks [26] with reasoning using first-order many-valued logic [6], all implemented in TENSORFLOW™ [13]. This enables, for the first time, a range of knowledge-based tasks using rich knowledge representation in first-order logic (FOL) to be combined with efficient data-driven machine learning based on the manipulation of real-valued vectors[1]. 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 specified compactly 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 reasoning, as detailed in what follows.

We are interested in providing a computationally adequate approach to implementing learning and reasoning [28] 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, a person may have a grounding into a 4-tuple containing some numerical representation of the person's name, her 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 on the objects of O; (ii) predict the numerical properties or the class of the objects in O.

Classes and relations are not normally independent. For example, it may be the case that if an object x is of class C, $C(x)$, 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 depend 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 examples to the contrary 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 [9, 2, 1], the framework introduced in this paper will be capable of combining in an effective way 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 repre-

[1] In practice, FOL reasoning including function symbols is approximated through the usual iterative deepening of clause depth.

sentational power of (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.

The paper is organized as follows. In Section 2, we define Real Logic. In Section 3, we propose the Learning-as-Inference framework. In Section 4, we instantiate the framework by showing how Real Logic can be implemented in deep Tensor Neural Networks leading to Logic Tensor Networks (LTN). Section 5 contains an example of how LTN handles knowledge completion using (possibly inconsistent) data and knowledge from the well-known *smokers and friends* experiment. Section 6 concludes the paper and discusses directions for future work.

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 binary relation R; $\forall xy.(R(x,y) \rightarrow R(y,x))$ states that R is a symmetric 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. For simplicity, without loss of generality, we assume that all logical sentences of $\mathcal{L}$ are in prenex conjunctive, skolemised normal form [16], e.g. a sentence $\forall x(A(x) \rightarrow \exists y R(x,y))$ is transformed into an equivalent clause $\neg A(x) \vee R(x, f(x))$, where f is a new 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*[2].

- $\mathcal{G}$ 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 clause ϕ of $\mathcal{L}$. Intuitively, $\mathcal{G}(\phi)$ represents one's confidence in the truth of ϕ; 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.

Definition 1. *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. *$\mathcal{G}(f) \in \mathbb{R}^{n \cdot \alpha(f)} \longrightarrow \mathbb{R}^n$ for every $f \in \mathcal{F}$;*

[2] In logic, the term "grounding" indicates the operation of replacing the variables of a term/formula with constants. To avoid confusion, we use the term "instantiation" for this.

3. $\mathcal{G}(P) \in \mathbb{R}^{n \cdot \alpha(R)} \longrightarrow [0,1]$ *for every* $P \in \mathcal{P}$*;*

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 μ is an s-norm operator, also known as a t-co-norm operator (i.e. the dual of some t-norm operator). [3]

Example 1. Suppose that $O = \{o_1, o_2, o_3\}$ is a set of documents defined on a finite dictionary $D = \{w_1, ..., w_n\}$ of n words. Let $\mathcal{L}$ be the language that contains the binary function symbol $concat(x,y)$ denoting the document resulting from the concatenation of documents x with y. 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 to each document its bag-of-words vector [7]. 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 $\mathbf{v}, \mathbf{u} \in \mathbb{R}^n$, $\mathcal{G}(Sim)(\mathbf{u},\mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}||||\mathbf{v}||}$.

For instance, if the three documents are 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"*, and W=$\{$*John, Mary, and, football, game, logic, play, study, together*$\}$ then the following are examples of the grounding of terms, atomic formulas and clauses.

$$\begin{aligned}
\mathcal{G}(o_1) &= \langle 1,0,1,1,0,1,1,1,0 \rangle \\
\mathcal{G}(o_2) &= \langle 0,1,1,1,1,1,1,0,0 \rangle \\
\mathcal{G}(o_3) &= \langle 1,1,2,1,0,1,1,1,1 \rangle \\
\mathcal{G}(concat(o_1,o_2)) &= \mathcal{G}(o_1) + \mathcal{G}(o_2) = \langle 1,1,2,2,1,2,2,1,0 \rangle \\
\mathcal{G}(Sim(concat(o_1,o_2),o_3) &= \frac{\mathcal{G}(concat(o_1,o_2)) \cdot \mathcal{G}(o_3)}{||\mathcal{G}(concat(o_1,o_2))|| \cdot ||\mathcal{G}(o_3)||} \approx \frac{13}{14.83} \approx 0.88 \\
\mathcal{G}(Sim(o_1,o_3) \vee Sim(o_2,o_3)) &= \mu_{max}(\mathcal{G}(Sim(o_1,o_3),\mathcal{G}(Sim(o_2,o_3)) \\
&\approx \max(0.86, 0.73) = 0.86
\end{aligned}$$

[3] Examples of t-norms which can be chosen here 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 $\mu_{max}(x,y) = \max(x,y)$.

3 Learning as approximate satisfiability

We start by defining ground theory and their satisfiability.

Definition 2 (Satisfiability). *Let ϕ be a closed clause in $\mathcal{L}$, $\mathcal{G}$ a grounding, and $v \leq w \in [0,1]$. We say that $\mathcal{G}$ satisfies ϕ in the confidence interval $[v,w]$, written $\mathcal{G} \models_v^w \phi$, if $v \leq \mathcal{G}(\phi) \leq w$.*

A partial grounding, denoted by $\hat{\mathcal{G}}$, is a grounding that is defined on a subset of the signature of $\mathcal{L}$. A grounded theory is a set of clauses in the language of $\mathcal{L}$ and partial grounding $\hat{\mathcal{G}}$.

Definition 3 (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 $[v,w] \subseteq [0,1]$ is an interval contained in $[0,1]$, and $\hat{\mathcal{G}}$ is a partial grounding.*

Definition 4 (Satisfiability of a Grounded Theory). *A GT $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ is satisfiabile if there exists a grounding $\mathcal{G}$, which extends $\hat{\mathcal{G}}$ such that for all $\langle [v,w], \phi(\mathbf{x}) \rangle \in \mathcal{K}$ and any tuple $\mathbf{t}$ of closed terms, $\mathcal{G} \models_v^w \phi(\mathbf{t})$.*

From the previous definiiton it follows that checking if a GT $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ is satisfiable amounts to seaching for an extension of the partial grounding $\hat{\mathcal{G}}$ in the space of *all possible groundings*, such that *all* the instantiations of the clauses in $\mathcal{K}$ are satisfied w.r.t. the specified interval. Clearly this is unfeasible from a practical point of view. As is usual, we must restrict both the space of grounding and clause instantiations. Let us consider each in turn: To check satisfiability on a subset of all the functions on real numbers, recall that a grounding should capture a latent correlation between the quantitative attributes of an object and its relational properties[4]. In particular, we are interested in searching within a specific class of functions, in this paper based on tensor networks, although other family of functions can be considered. To limit the number of clause instantiations, which in general might be infinite since $\mathcal{L}$ admits function symbols, the usual approach is to consider the instantiations of each clause up to a certain depth [3].

When a grounded theory $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ is inconsitent, that is, there is no grounding $\mathcal{G}$ that satisfies it, we are interested in finding a grounding which satisfies *as much as possible* of $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$. For any $\langle [v,w], \phi \rangle \in \mathcal{K}$ we want to find a grounding $\mathcal{G}$ that minimizes the *satisfiability error*. An error occurs when a grounding $\mathcal{G}$ assigns a value $\mathcal{G}(\phi)$ to a clause ϕ 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 $\mathcal{G}(\phi)$:

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

[4] For example, whether a document is classified as from the field of Artificial Intelligence (AI) 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 AI" then the grounding of $AI(x)$, which is a function from bag-of-words vectors to [0,1], should assign values close to 1 to the vectors which are close semantically to AI. Furthermore, if two vectors are similar (e.g. according to the cosine similarity measure) then their grounding should be similar.

Notice that if $\mathcal{G}(\phi) \in [v, w]$, $\text{Loss}(\mathcal{G}, \phi) = 0$.

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 5 (Approximate satisfiability). *Let $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ be a grounded theory and $\mathcal{K}_0$ a finite subset of the instantiations of the clauses in $\mathcal{K}$, i.e.*

$$K_0 \subseteq \{\langle [v, w], \phi(\mathbf{t}) \rangle\} \mid \langle [v, w], \phi(\mathbf{x}) \rangle \in \mathcal{K} \textit{ and } \mathbf{t} \textit{ is any } n\textit{-tuple of closed terms.}\}$$

Let $\mathbb{G}$ be a family of grounding functions. 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 set $\mathcal{K}_0$, that is:

$$\mathcal{G}^* = \underset{\hat{\mathcal{G}} \subseteq \mathcal{G} \in \mathbb{G}}{\text{argmin}} \sum_{\langle [v,w], \phi(\mathbf{t}) \rangle \in \mathcal{K}_0} \textit{Loss}(\mathcal{G}, \langle [v, w], \phi(\mathbf{t}) \rangle)$$

4 Implementing Real Logic in Tensor Networks

Specific instances of Real Logic can be obtained by selectiong the space $\mathbb{G}$ of groundings and the specific s-norm for the interpretation of disjunction. 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 $\mathcal{G}(f)(\mathbf{v}_1, \ldots, \mathbf{v}_m)$ can be written as:

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

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 m-ary predicate P, $\mathcal{G}(P)$, is defined as a generalization of the neural tensor network [26] (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:

$$\mathcal{G}(P) = \sigma\left(u_P^T \tanh\left(\mathbf{v}^T W_P^{[1:k]} \mathbf{v} + V_P \mathbf{v} + B_P\right)\right) \quad (2)$$

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 σ 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 $\neg P(x, y) \to A(y)$ is shown in Figure 1. This architecture is a generalization of the structure proposed in [26], that has been shown rather effective for the task of knowledge compilation, also in presence of simple logical constraints. 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 $P(x, y) \to A(y)$.

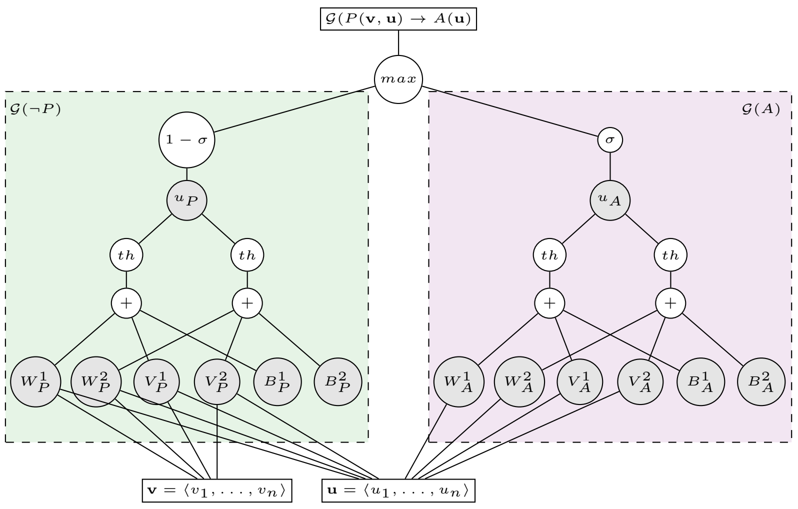

$\mathcal{G}(P(\mathbf{v},\mathbf{u}) \to A(\mathbf{u}))$
max
$\mathcal{G}(\neg P)$
$\mathcal{G}(A)$
$1-\sigma$
σ
u_P
u_A
th
$+$
W_P^1 W_P^2 V_P^1 V_P^2 B_P^1 B_P^2
W_A^1 W_A^2 V_A^1 V_A^2 B_A^1 B_A^2
$\mathbf{v} = \langle v_1, \ldots, v_n \rangle$
$\mathbf{u} = \langle u_1, \ldots, u_n \rangle$

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

5 An Example of Knowledge Completion

Logic Tensor Networks have been implemented as a Python library called `ltn` using Google's TENSORFLOW™ . To test our idea, in this section we use the well-known *friends and smokers*[5] example [24] 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 imcomplete 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 Figure 2.

The facts contained in the knowledge-bases should have different degrees of truth, and this is not known. Otherwise, the combined knowledge-base would be inconsistent (it would deduce e.g. $S(b)$ and $\neg S(b)$). Our main task is to complete the knowledge-base (KB), that is: (i) find the degree of truth of the facts contained in KB, (ii) find a truth-value for all the missing facts, e.g. $C(i)$, (iii) find the grounding of each constant symbol $a, ..., n$.[6] To answer (i)-(iii), we use `ltn` to find a grounding that best

[5] 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.

[6] Notice how no grounding is provided about the signature of the knowledge-base.

$\mathcal{K}^{SFC}_{a\ldots h}$

$$\begin{array}{r} S(a),\ S(e),\ S(f),\ S(g), \\ \neg S(b),\ \neg S(c),\ \neg S(d),\ \neg S(g),\ \neg S(h), \\ F(a,b),\ F(a,e),\ F(a,f),\ F(a,g),\ F(b,c), \\ F(c,d),\ F(e,f),\ F(g,h), \\ \neg F(a,c),\ \neg F(a,d),\ \neg F(a,h),\ \neg F(b,d),\ \neg F(b,e), \\ \neg F(b,f),\ \neg F(b,g),\ \neg F(b,h),\ \neg F(c,e),\ \neg F(c,f), \\ \neg F(c,g),\ \neg F(c,h),\ \neg F(d,e),\ \neg F(d,f),\ \neg F(d,g), \\ \neg F(d,h),\ \neg F(e,g),\ \neg F(e,h),\ \neg F(f,g),\ \neg F(f,h), \\ C(a),\ C(e), \\ \neg C(b),\ \neg C(c),\ \neg C(d),\ \neg C(f),\ \neg C(g),\ \neg C(h) \end{array}$$

$\mathcal{K}^{SF}_{i\ldots n}$

$$\begin{array}{r} S(i),\ S(n), \\ \neg S(j),\ \neg S(k), \\ \neg S(l),\ \neg S(m), \\ F(i,j),\ F(i,m), \\ F(k,l),\ F(m,n), \\ \neg F(i,k),\ \neg F(i,l), \\ \neg F(i,n),\ \neg F(j,k), \\ \neg F(j,l),\ \neg F(j,m), \\ \neg F(j,n),\ \neg F(l,n), \\ \neg F(k,m),\ \neg F(l,m) \end{array}$$

$\mathcal{K}^{SFC}$

$$\begin{array}{rr} \forall x \neg F(x,x), & \forall xy (S(x) \wedge F(x,y) \rightarrow S(y)), \\ \forall xy (F(x,y) \rightarrow F(y,x)), & \forall x (S(x) \rightarrow C(x)) \\ \forall x \exists y F(x,y), & \end{array}$$

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

approximates the complete KB. We start by assuming that all the facts contained in the knowledge-base are true (i.e. have degree of truth 1). To show the role of background knolwedge in the learning-inference process, we run two experiments. In the first ($exp1$), we seek to complete a KB consisting of only factual knowledge: $\mathcal{K}_{exp1} = \mathcal{K}^{SFC}_{a\ldots h} \cup \mathcal{K}^{SF}_{i\ldots n}$. In the second ($exp1$), we also include background knowledge, that is: $\mathcal{K}_{exp2} = \mathcal{K}_{exp1} \cup \mathcal{K}^{SFC}$.

We confgure the network as follows: each constant (i.e. person) can have up to 30 real-valued features. We set the number of layers k in the tensor network to 10, and the regularization parameter[7] $\lambda = 1^{-10}$. For the purpose of illustration, we use the Lukasiewicz t-norm with s-norm $\mu(a,b) = \min(1, a+b)$, and use the harmonic mean as aggregation operator. An estimation of the optimal grounding is obtained after 5,000 runs of the RMSProp learning algorithm [27] 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 listed in a knowledge-base are highlighted with the same background color of the knowledge-base in Figure 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 F and C 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

[7] A smoothing factor $\lambda ||\mathbf{\Omega}||^2_2$ is added to the loss function to create a preference for learned parameters with a lower absolute value.

			F							
	S	C	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

			F					
	S	C	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 $\mathcal{K}_{exp1} = \mathcal{K}^{SFC}_{a\ldots h} \cup \mathcal{K}^{SF}_{i\ldots n}$

			F							
	S	C	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

			F					
	S	C	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 $\mathcal{K}_{exp2} = \mathcal{K}^{SFC}_{a\ldots h} \cup \mathcal{K}^{SF}_{i\ldots n} \cup \mathcal{K}^{SFC}$

Table 1.

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 $C(i)$ 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, [15], Guha 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 [15] 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 [15] 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 [24, 29, 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 from 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 [19]. These are essentially propositional while real logic is first-order.

Bayesian logic (BLOG) [20] is open domain, and in this respect 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 into this category, including probabilistic variations of inductive logic programming (ILP) [23], which are normally restricted to Horn clauses. Metainterpretive ILP [21], together with BLOG, seem closer to LTNs in what concerns the knowledge representation language, but do not explore the benefits of tensor networks for computational efficiency.

An approach for embedding logical knowledge onto data for the purpose of relational learning, similar to Real Logic, is presented in [25]. Real Logic and [25] share the idea of interpreting a logical alphabet in an n-dimensional real space. Terminologically, the term “grounding” in Real Logic corresponds to “embeddings” in [25]. However, there are several differences. First, [25] uses function-free langauges, while we provide also groundings for functional symbols. Second, the model used to compute the truth-values of atomic formulas adopted in [25] is a special case of the more general model proposed in this paper (as described in Eq. (2)). Finally, the semantics of the universal and existential quantifiers adopted in [25] is based on the closed-world assumption (CWA), i.e. universally (respectively, existentially) quantified formulas are reduced to the finite conjunctions (respectively, disjunctions) of all of their possible instantiations; Real Logic does not make the CWA. Furthermore, Real Logic does not assume a specific t-norm.

As in [11], LTN is a framework for learning in the presence of logical constraints. LTNs share with [11] the idea that logical constraints and training examples can be treated uniformly as supervisions of a learning algorithm. LTN introduces two novelties: first, in LTN existential quantifiers are not grounded into a finite disjunction, but are *scolemized*. In other words, CWA is not required, and existentially quantified formulas can be satisfied by "new individuals". Second, LTN allows one to generate data for prediction. 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 introduced by the formula) which for every vector $\mathbf{v}$ returns the feature vector $f(\mathbf{v})$, which can be intuitively interpreted as being the set of features of a *typical* object which takes part in relation R with the object having features equal to $\mathbf{v}$.

Finally, related work in the domain of neural-symbolic computing and neural network fibring [10] has sought to combine neural networks with ILP to gain efficiency [14] 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 [1].

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. *Cognitive Computation: Integrating Neural and Symbolic Approaches*, Workshop at NIPS 2015, Montreal, Canada, April 2016. CEUR-WS 1583.
2. *Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches*, AAAI Spring Symposium, Stanford University, CA, USA, March 2015.
3. Dimitris Achlioptas. Random satisfiability. In *Handbook of Satisfiability*, pages 245–270. 2009.
4. Leon Barrett, Jerome Feldman, and Liam MacDermed. A (somewhat) new solution to the variable binding problem. *Neural Computation*, 20(9):2361–2378, 2008.
5. Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.
6. M. Bergmann. *An Introduction to Many-Valued and Fuzzy Logic: Semantics, Algebras, and Derivation Systems*. Cambridge University Press, 2008.
7. David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.
8. Léon Bottou. From machine learning to machine reasoning. Technical report, arXiv.1102.1808, February 2011.
9. Artur S. d'Avila Garcez, Marco Gori, Pascal Hitzler, and Luís C. Lamb. Neural-symbolic learning and reasoning (dagstuhl seminar 14381). *Dagstuhl Reports*, 4(9):50–84, 2014.

10. Artur S. d'Avila Garcez, Luís C. Lamb, and Dov M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Cognitive Technologies. Springer, 2009.
11. Michelangelo Diligenti, Marco Gori, Marco Maggini, and Leonardo Rigutini. Bridging logic and kernel machines. *Machine Learning*, 86(1):57–88, 2012.
12. David Silver et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.
13. Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
14. Manoel V. M. França, Gerson Zaverucha, and Artur S. d'Avila Garcez. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine Learning*, 94(1):81–104, 2014.
15. Ramanathan Guha. Towards a model theory for distributed representations. In *2015 AAAI Spring Symposium Series*, 2015.
16. Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, New York, NY, USA, 2004.
17. Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, January 2003.
18. Douwe Kiela and Léon Bottou. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of EMNLP 2014*, Doha, Qatar, 2014.
19. Bart Kosko. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.
20. Brian Milch, Bhaskara Marthi, Stuart J. Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: probabilistic models with unknown objects. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 1352–1359, 2005.
21. Stephen H. Muggleton, Dianhuan Lin, and Alireza Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. *Machine Learning*, 100(1):49–73, 2015.
22. Aniruddh Nath and Pedro M. Domingos. Learning relational sum-product networks. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2878–2886, 2015.
23. Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2016.
24. Matthew Richardson and Pedro Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, February 2006.
25. Tim Rocktaschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, June 2015.
26. Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*. 2013.
27. T. Tieleman and G. Hinton. Lecture 6.5 - RMSProp, COURSERA: Neural networks for machine learning. Technical report, 2012.
28. Leslie G. Valiant. Robust logics. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 642–651, New York, NY, USA, 1999. ACM.
29. Jue Wang and Pedro M. Domingos. Hybrid markov logic networks. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1106–1111, 2008.

Learning sequential control in a Neural Blackboard Architecture for in situ concept reasoning

Frank van der Velde
University of Twente, CPE-CTIT; IOP, Leiden University, The Netherlands
f.vandervelde@utwente.nl

Abstract. Simulations are presented and discussed of learning sequential control in a Neural Blackboard Architecture (NBA) for in situ concept-based reasoning. Sequential control is learned in a reservoir network, consisting of columns with neural circuits. This allows the reservoir to control the dynamics of processing by responding to information given by questions and the activations in the NBA. The in situ nature of concept representation directly influences the reasoning process and learning in the architecture.

Keywords. Learning • Neural blackboard architecture • In situ concepts • Reasoning • Reservoir • Wilson-Cowan dynamics

1 Introduction

Neural representation and processing of symbol-like structures as presented here takes its inspiration from the observation that concept representations in the brain are 'in situ', in line with the neural assemblies as proposed by Hebb ([1]). Neural assemblies, as Hebb argued, will develop over time when neurons that process information about, for example, a concept become interconnected. Such concept representations could be distributed, but parts of the assembly could also consist of more local representations. They will generally consist of neurons involved in processing information but also of neurons involved in actions. In this way, in situ concepts representations are always grounded in perception or action, so that the neural connection structure underlying a concept is determined by both its 'incoming' (perception-based) connections and its 'outgoing' (action-generating) connections [2].

The in situ nature of neuronal concept representations imposes constraints on the way they can be combined to represent and process more complex forms of information. However, complex conceptual structures (e.g., sentences with hierarchical structures) can be represented and processed when the neural assemblies underlying in situ concept representation are embedded in a 'Neural Blackboard Architecture' or NBA [3].

In general, "in-situ concept-based computing" would be achieved by embedding in situ concepts in several NBAs, each needed for a specific form of processing. Blackboards are also used in computer domains, e.g., to store arbitrary forms of (symbolic) information. NBAs as intended here, however, are fundamentally different. They possess structural information, (e.g., related to sentence structures as in [3]), and are implemented with dedicated structures (e.g., neural circuits, as in the brain). In this way they cannot store arbitrary information, but they can process specific forms of (high-level cognitive) information, e.g., by the interactions between the structured representations in the blackboards. Fig 1. illustrates that there will be NBAs for sentence structures, phonological structures (e.g., forming new words), sequential structures based on in situ concept representations, relation structures as used in reasoning, and potentially other NBAs (blackboards) as well. The interaction between these NBAs derives from the in situ concept representations they share. For example, a word (concept) would be shared by the sentence, phonology, sequential and relation blackboards (and potentially more). In turn, concepts are also related to each other in several "feature spaces", which can influence processing in the NBAs as well.

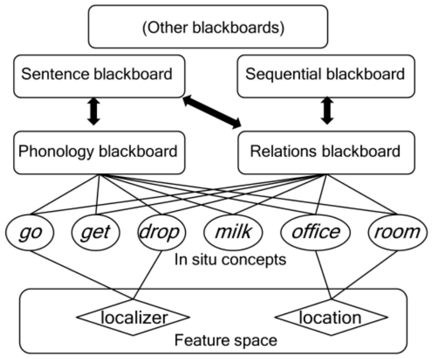
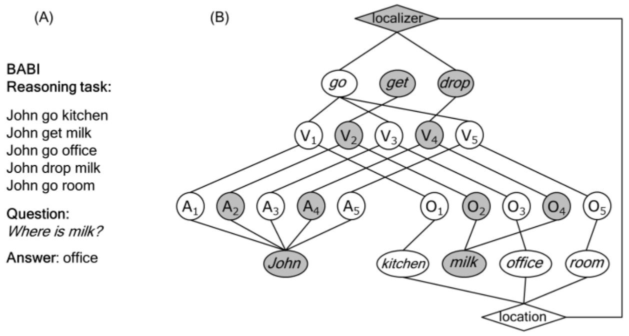


Figure 1. Overview of in situ concept-based computing with Neural Blackboard Architectures (NBAs).

The NBA in [3] can account for sentence structure and processing, including sentence learning [4] and examples of ambiguity resolution and garden path modelling [5]. A recent extension of the NBA approach to symbol-like processing is the NBA for (basic forms of) reasoning presented in [6], which can be used in basic reasoning processes, such as BABI reasoning tasks as presented in [7].


(A)

BABI
Reasoning task:

John go kitchen
John get milk
John go office
John drop milk
John go room

Question:
Where is milk?

Answer: office

Figure 2. (A) BABI task for reasoning (after [7]). (B) Propositions in a relation blackboard (A = agent, O = object). Grey nodes activated by the question Where is milk?. <localizer> and <location> are concept features, belonging to feature space.

Fig. 2A presents an example of a BABI reasoning task. A set of relations is given and a question has to be answered on the basis of these relations (propositions). So, the question *Where is milk?* can be answered by first retrieving *John drop milk* (providing a location for *milk*) and then retrieving *John go office* as the last location of *John* before *John drop milk*. This would provide *office* as the location of *milk*. Fig. 2B presents the representations of these relations in the relation NBA, and the representations activated by the question *Where is milk?* are illustrated in grey.

Here, a first set of simulations will be presented and discussed that address the way NBAs can learn to perform reasoning processes of this kind. This paper focusses on the fundamental issue of whether questions can retrieve information needed for reasoning in NBAs. Simulations of other aspects of the NBAs and reasoning process are outside the scope of this paper, or under development (e.g., the selection of the more recent activated relation in the sequence blackboard, which is assumed here). A further description of how BABI tasks can be solved in NBAs is presented in [6].

2 Learning control of reasoning in an NBA

A key element of in situ concept processing in NBAs is that the information provided by a question is directly used in the activation of the concepts involved. In Fig. 2B, for example, the question *Where is milk*? directly activates the in situ concepts *is* and *milk*. In turn, they will activate their features (e.g. <localizer> for *is*) in feature space. Due to the activation of the in situ concept *milk,* all relations in Fig. 2A in which *milk* occurs can be activated as well, because they share the same in situ concept (*milk*). So, *John drop milk* and *John get milk* can be directly activated in this way.

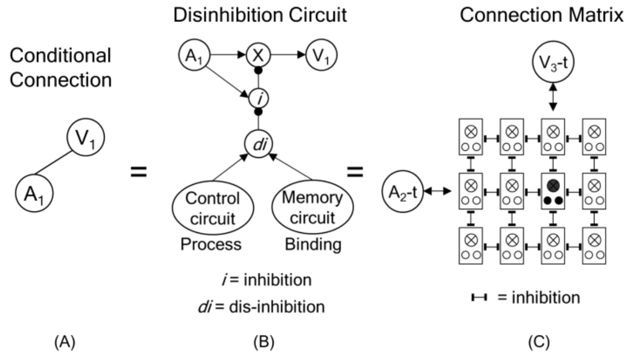


Figure 3. (A). Conditional connection in Fig. 2B. (B) Conditional connections with disinhibition circuits . (C) A connection matrix of conditional connections for binding.

Activation of concept structures (here relations) in an NBA depends on the nature of the structure representations, and the control and binding circuits in the NBA. In [3] an extensive discussion of these is given for the sentence NBA, but they are the same in the relation NBA of Fig. 2. Fig. 3 illustrates a basic overview. Each connection in Fig. 2B represents a conditional connection. These connections can operate only when they are activated. In this way, relations can be represented and processed in NBAs (instead of just associations as with unconditional connections). A conditional connection can be implemented with a disinhibition circuit, as illustrated in Fig. 3B. The circuit can be activated by a control circuit or by a memory circuit. The latter produces (temporal) bindings in the NBA. The process of binding and (re)activation is determined by the control circuits.

For example, the binding and (re)activation of *John drop milk* in Fig. 2B proceeds as follows. First, *John* is activated. To achieve the representation of *John* as agent, *John* and an (arbitrary) Agent node in the NBA (e.g., A4) are bound by activating the memory circuit between them ([3]). This results in the binding of *John* and A4. In the same way, *drop* binds to V4 and *milk* as object to O4. Then, a control circuit activates the conditional connections between A4 and V4 to represent *John drop*. To achieve a binding between arbitrary A and O nodes, all A and O nodes are connected to each other in a connection matrix, as illustrated in Fig. 3C. A connection matrix is a matrix of circuits ('columns') that regulate the binding process. To bind A4 with V4, the control circuit activates the conditional connections between A4 and V4 and their corresponding column in the connection matrix. This, in turn results in the activation of the memory circuit in that column. As long as this circuit remains active (with sustained or 'delay' activation), A4 and V4 are bound, even when they themselves are deactivated again. This binding process is used for all bindings in the NBA.

The relation *John drop milk* can be reactivated by activating one of the in situ concepts involved and the required conditional connections. For example, the question *Where is milk?* activates *milk*. Because of their binding with *milk*, O4 (and O2) are activated as well. By activating the conditional connections for Object between Object and Verb nodes, O4 activates V4, which activates *drop*. A4 can be activated by activating (enabling) the Agent conditional connections between Verb nodes and Agent nodes. This results in the reactivation of *John drop milk*. In Fig. 2B, this process also results in the reactivation of *John get milk*, because these binding are active as well (i.e., this relation is also stored in the NBA). A distinction between these two relations can be made by a sequential ordering in a sequence blackboard (assumed here, e.g., see [6]).

The reactivation of stored relations is crucial for a reasoning process as illustrated in Fig. 2A. For example, *Where is milk?* can be answered by first activating *John drop milk* and *John get milk*, using the activation of *milk* by the question. Then by selecting *John drop milk* as the more recent relation. In this case, *drop* indicates a location for *milk*, given by the <localizer> feature of *drop* in Fig. 2. This would initiate a second question *Where is agent-drop?*, i.e., *Where is John?* here. This question would activate *John go office*, selected as the most recent location of *John*. This produces *office* as the location of *milk*. In other words, new questions in the reasoning process derive from activations in the NBA initiated by previous questions.

Hence, the activations initiated by the (first) question and the resulting interactions with the blackboard determine the process of answering questions like *Where is milk*? Here, this interaction is simulated by using a control network to recognize the (first) question and initiate the required interaction (e.g., further questions) with the blackboard. The control network consists of a form of reservoir computing (e.g., [8]).

2.1 Reservoir for control

A reservoir is a set of neurons or 'nodes' that are sparsely interconnected in a random (fixed) fashion. Also, the nodes are connected (randomly) to input neurons providing external information. A reservoir can learn to activate specific output neurons in response to a sequence presented to it. In this way, they can learn to process and recognize sequential information [8].

Hinaut and Dominey [9] used a reservoir to recognize sets of sentences. However, in a reservoir the nodes activate each other based on their (node) activation dynamics. When a sequence with a specific sequential dynamics is presented to the reservoir, it can learn to 'resonate' to the external dynamics because that is predictable [8]. This is typically more difficult for language, because timing differences between words can vary. In [9] this was solved by adjusting the dynamics of the reservoir nodes to regular word presentation timing. Here, however, the sequence to be learned is not only determined by the presented question but also by the interactions with and within the neural blackboard, which could vary given the amount of information stored and processed in it. So, a direct adjustment of timing of node activation is not possible. Therefore, the reservoir presented and simulated here is more complex.

Fig. 4 illustrates that the reservoir consists of columns, which in turn consist of neural circuits. The sequential activation as produced by the reservoir is given by the 'sequence' (S) nodes. They are randomly and sparsely connected to each other, in a fixed manner. However, S nodes do not directly activate each other. Instead, an active node Si will activate a 'delay' population in the column of a node Sj to which it is connected. The delay population remains active (unless inhibited). It activates Sj but also an inhibitory node i, which inhibits Sj. In this way, the timing of the sequence in the reservoir is under control. Sj can be activated only when node i is inhibited. As indicated, this will happen when an 'Item' node activates another inhibitory node that inhibits i. When this happens, Sj will be activated and it will in turn activate other S nodes in the sequence in the same manner.

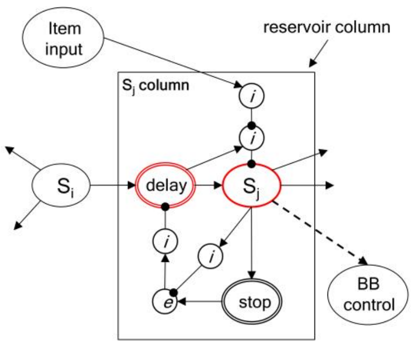


Figure 4. Reservoir of columns. Circles and ovals represent neural populations. Double lined ovals remain active (sustained or delay activity). I = inhibition, e = excitation. S = sequence. Dashed connections are modifiable by learning (e.g. LTP).

Item nodes represent the external inputs to the reservoir. Here, they consist of sentence information and/or information derived from the blackboard. Hence, the sequential dynamics produced by the reservoir is under control by the information given by the question and the interactions produced in the blackboard.

The aim of the reasoning NBA is to simulate and learn reasoning in this way. That is, the reservoir will learn to recognize sequential information given by the question and by activations in the blackboard to initiate new activations in the blackboard, until the question can be answered. Learning can be achieved by the adaptive connections between S nodes and nodes that control the binding and (re)activation process in the blackboard. So, Sj could learn to activate a specific control in the blackboard, such as the control to activate the Agent or Object conditional connections.

Here, basic aspects of this process are simulated for answering the questions *Where is John* and *Where is milk?* in the task illustrated in Fig. 2A.

3 Simulation of reservoir activity

All the populations in the NBA are modelled with Wilson Cowan population dynamics [10]. Each population consist of groups of interacting excitatory (E) in inhibitory (I) neurons. The behavior of the E and I groups are each modeled with an ODE at population level. Both ODEs interact and they receive input from outside. A working memory (or delay) population consists of two interacting populations, say A and B. The output results from A. The role of B is to sustain the activity by its interaction with A. It is assumed that B has a lower activation maximum than other populations. This results in a reduced activity of a working memory population when it relies on delay activity only (i.e., does not receive input). The E neurons are used for output to other populations. Populations are excitatory when their output connection has a positive weight. They are inhibitory when their output connection has a negative weight. All populations operate with the same parameters and all weights are the same (as in [5]). The behavior of the populations is simulated with a fourth order Runge Kutta numerical integration (with h = 0.1).

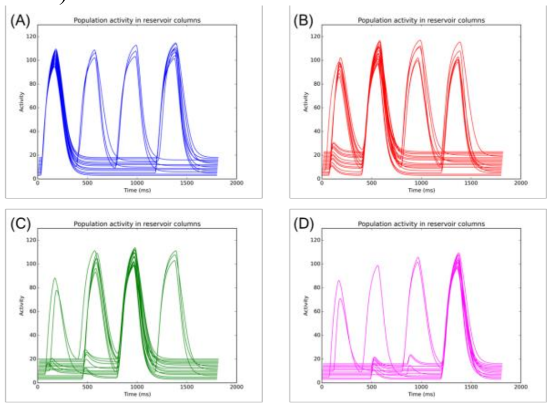


Figure 5. Activations of reservoir S nodes. (A) Where. (B) <localizer>. (C) noun. (D) Agent.

The question *Where is John?* is presented to the reservoir word by word. However, for the reservoir word type and feature information is used. Words like *is* and *go* are represented as <localizer>, words like *John* and *milk* are presented as nouns. The specific words in the questions are used to activate their (in situ) representations in the backboard. So, the active concept *John* activates the nodes A4 and A2 in the blackboard. This provides information that *John* is the agent in the relations stored in the blackboard. In turn, this information can be used to learn that the blackboard should provide the object information related to (bound to) *John is*.

The reservoir can learn to do this by recognizing the item sequence *Where - <localizer> - noun – Agent* and producing the activation of the Object conditional connections in the blackboard. This will produce the activations of *John go kitchen*, *John go office*, and *John go room*, from which *John go room* can be selected as the most recent, using the sequential blackboard in Fig. 1 (see [6]).

Fig. 5 presents the activations of sets of S nodes in a reservoir of 750 columns with sparse connectivity in response to the item sequence *Where - <localizer> - noun – Agent*. The first three items (*Where - <localizer> - noun*) are based on the question, the fourth item (*Agent*) is derived from the blackboard. Each color represents a different set of S nodes in the reservoir. The blue set are S nodes that are initially activated by start nodes (not shown), that respond to the start of a question. They also respond to the item *Where* (specifically). However, as the figure shows, some of these nodes also respond to the other items in the item sequence presented to the reservoir.

The red S nodes are activated by the blue S nodes (columns) and also specifically respond to the second item *<localizer>*. But some of them also respond to the other items at other steps in the sequence. Likewise, the green S nodes specifically respond to the third item because they are activated by the combination of the active red S nodes and the item *noun*. Again, however, some of them respond to other items at other sequence steps as well. The magenta S nodes specifically respond to the fourth item, because they are activated by the green S nodes and the item *Agent*.

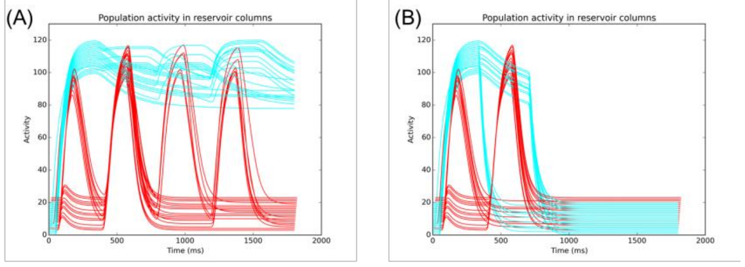


Figure 6. (A). Activation of the red nodes in Fig 5 and the delay populations in their colums (cyan). (B). The same activations after activation stop.

The active magenta S nodes could be used to learn that the blackboard should activate the Object conditional connection, because *John* (*noun*) is an Agent and the question asks for an object bound to that agent (and a localizer word). Here, that would be possible when the magenta S nodes dominate the activation in the reservoir at the fourth step. However, a substantial amount of other nodes are active as well. But reservoir nodes can learn specific responses based on their distributed activation ([9].

But when the activation of S nodes is more specific, such learning could be achieved by direct adjustments of neural weights (the dashed connections in Fig. 4), which would allow rapid forms of learning. To achieve more specific activation of S nodes it is important to look more closely at the activations produced in the columns.

Fig. 6A shows the activations of the red S nodes and the delay populations in their columns. The delay populations remain active. So, when a new item is presented, some of the red S nodes respond to that item because it is connected to their column. This accounts for the repeated activation of some of the S nodes of all color in Fig. 5.

Delay activity can be stopped, however, by a neural circuit illustrated in Fig. 4. To this end, the S node is connected to a ‘stop’ population (consisting of sustained activation). When Sj is active it will activate this population. But it will also activate an inhibitory neuron that prevents the effect of the stop population. The stop population can inhibit the delay population only when Sj is deactivated, and it continues to do so as long as it is active. The Sj node is deactivated when the Item node is deactivated, which will occur with the presentation of a new item in the sequence. It that case, the delay population ensures the deactivation of the Sj node, which in turn ensures the deactivation of the delay population.

Fig. 6B shows the effect of stopping the delay activation. After the red S nodes are deactivated, the delay activation is deactivated as well. This prevents further activation of the red S nodes. Yet, some of the red S nodes are active at the first step, even though they are not activated by the start nodes. This activation results from the rapid activation of their delay nodes by the blue S nodes (Figure 5) and the fact that these red S nodes also respond to the first item (*Where*). However, due to their activation at the first step, these red S nodes are no longer activated at the second step (unlike in Fig. 5) because their delay populations are deactivated. So, the S nodes in the second step are now specifically active for the second step in the sequence. Similarly, the first step in the sequence is now given by the blue S nodes and the red S nodes active at that step. In this way, specific sets of S nodes are activated at specific steps in the sequence. This allows for rapid learning by direct synaptic modification.

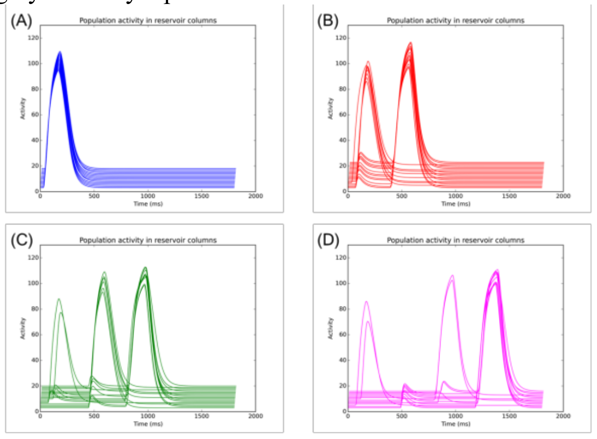


Figure 7. Activations of reservoir S nodes, with Stop of activation. (A) Where. (B) <localizer>. (C) noun. (D) Agent.

Fig. 7 shows the results for all S nodes presented in Fig. 5. The colored nodes are specifically active at the step in the sequence that is related to the item they represent.

Activation after that step is prevented. In some cases, some of the S nodes are active before that step. In that case, however, they will not be activated again. So, at each step a specific set of S nodes will be active that uniquely represents the item of that step. Specifically, the magenta S nodes can learn to produce the activation of Object conditional connections by direct synaptic modification.

3.1 Learning more complex control

The question *Where is John?* generates a direct answer by the reactivation of *John go room*. The question *Where is milk?*, however, does not directly produce an answer in this way. To answer this question, a second representation needs to be activated after *John drop milk*. In turn, this requires a longer and more complex sequential sequence to be learned by the reservoir and a longer interaction process with the blackboard. A reservoir can indeed learn such a process and interaction with the blackboard to produce the answer. Here, however, only a few aspects of that can be illustrated.

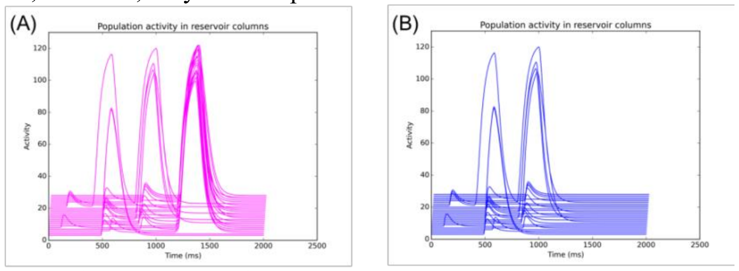


Figure 8. (A). Activation of S nodes in the fourth step in *Where is milk?* (B) Activation of the same S nodes with the question *Where is John*?

First, the question *Where is milk?* generates Object instead of Agent as response from the blackboard in the fourth step. Yet, the first three steps are the same as with *Where is John?*. Fig. 8A illustrates the activation of S nodes in the fourth step of *Where is milk?* These nodes are thus activated by the S nodes active at the third step and by the item *Object*, derived from the activation of O4 and O2 in the blackboard (Fig. 3). Fig. 8B illustrates the activation of the same S nodes when, at the fourth step, the item *Agent* is presented. It is clear that the S nodes selectively respond to the item *Object*, instead of *Agent*. This allows them to learn to activate the Agent conditional connections in the blackboard, to reactivate the relation *John drop milk*.

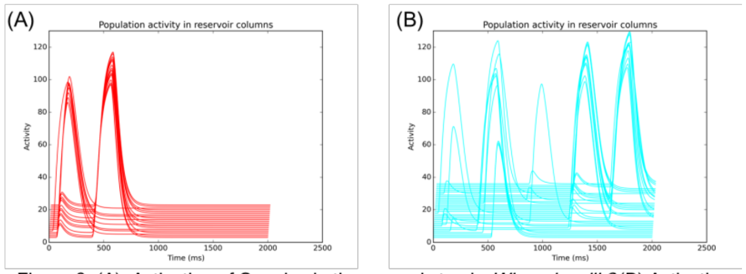


Figure 9. (A). Activation of S nodes in the second step in *Where is milk?* (B) Activation of the S nodes in the fifth step of that question.

Second, the question *Where is milk*? gives location information (*is*) at step two, but it also requires location information at step five in the process, to retrieve the location of *John* after *John drop milk* has been reactivated. Fig. 9 illustrates the activation in the reservoir related to the same information at different steps.

The red S nodes in Fig. 9A respond to the first activation of <*localizer*> in the process. The cyan S nodes in Fig. 9B respond to the second activation of <*localizer*> in the process. That is, these S nodes would all be activated by the active S nodes in the fourth step and by the item <*localizer*>. Some of them are already activated in some of the previous steps, however, which prevents their activation in the fifth step. Hence, the control (stop) of activation in the reservoir results in a set of S nodes that selectively respond to the item <*localizer*> in the fifth step, irrespective of the presence of that item in a previous step. Such a selective response to repeated activation of item information will be crucial for the success of learning reasoning in a neural reasoning architecture as presented here.

5 Conclusions

Simulations of the learning of sequential control in a neural blackboard architecture (NBA) for reasoning were presented. The NBA is based on in situ concept representation. This entails that concepts are always represented by the same underlying neural assemblies (although different parts of them might be activated at different occasions). The in situ nature of concepts imposes constraints on the ways they can be used to represent and process complex forms of conceptual information, as found in language or reasoning.

But it also provides distinctive benefits. First, in situ concepts are content addressable. Thus, as illustrated here, the concept and other information given by a question will directly select the related information stored in the neural blackboard by reactivating the in situ concept representations. This, in turn, can guide the reasoning process by interactions between the neural blackboard and a reservoir network that selectively responds to sequential information.

The interaction between neural blackboards and control networks (e.g., reservoirs) also offers new forms of learning, in which the distinction between structured neural blackboards, control circuits and content addressable activation by in situ concepts strongly reduces the number of contingencies that have to be learned.

Furthermore, in situ representations are not moved or copied. And, as noted, they are content addressable. Therefore, neural blackboard architectures of reasoning and other forms of (high-level) cognitive processing with in situ representations would be very suitable for implementation in (e.g., new) forms of parallel and power reduced hardware.

Acknowledgements

The work of the author was funded by the project ConCreTe. The project ConCreTe acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET grant number 611733.

References

1. Hebb, D. O. (1949). *The organisation of behaviour*. New York: Wiley.
2 van der Velde, F (2015). Communication, concepts and grounding. *Neural networks*, *62* , 112 - 117.
3 van der Velde, F. and de Kamps, M. (2006). Neural blackboard architectures of combinatorial structures in cognition. *Behavioral and Brain Sciences*, *29*, 37-70.
4. van der Velde, F & de Kamps, M. (2010). Learning of control in a neural architecture of grounded language processing. *Cognitive Systems Research*, *11*, 93–107.
5 van der Velde, F., and de Kamps, M. (2015). Combinatorial structures and processing in neural blackboard architectures. In *Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches (CoCo@NIPS 2015)*, eds T. R. Besold, A. d'Avila Garcez, G. F. Marcus, and R. Miikkulainen (Montreal). CEUR Workshop Proceedings (pp. 1-9).
6. van der Velde, F. (2016). Concepts and relations in neurally inspired in situ concept-based computing. *Frontiers in Neurorobotics*. *10:4*. doi: 10.3389/fnbot.2016.00004
7 Bordes, A., Weston, J., Chopra, S., Mikolov, T., Joulin, A., Rush, S., & Bottou, L. (2015). *Artificial Tasks for Artificial Intelligence*. Facebook AI Research. ICLR – San Diego – May 7, 2015. http://www.iclr.cc/lib/exe/fetch.php?media=iclr2015:abordes-iclr2015.pdf
8. Jaeger, H and Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, *304*, 78-80.
9. Hinaut X, Dominey PF (2013) Real-Time Parallel Processing of Grammatical Structure in the Fronto-Striatal System: A Recurrent Network Simulation Study Using Reservoir Computing. *PLoS ONE 8(2):* e52946. doi:10.1371/journal.pone.0052946
10 Wilson HR, Cowan JD (1972) Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical Journal*, *12*, 1–24

A Proposal for Common Dataset in Neural-Symbolic Reasoning Studies

Ozgur Yilmaz, Artur d'Avila Garcez, and Daniel Silver

Turgut Ozal University, Computer Science Department, Ankara Turkey
City University London, Department of Computer Science, London UK
Acadia University, Jodrey School of Computer Science, Nova Scotia Canada,
ozyilmaz@turgutozal.edu.tr,a.garcez@city.ac.uk
danny.silver@acadiau.ca

Abstract. We promote and analyze the needs of a common publicly available benchmark dataset to be used for neural-symbolic studies of learning and reasoning. The recently released Visual Genome repository is proposed as a suitable dataset to meet these needs. Along with the original tasks that were suggested by the Visual Genome creators, we propose neural-symbolic tasks that can be used as challenges to promote research in the field and competition between lab groups.

Keywords: Neural-symbolic computing, common dataset, relational learning, reasoning, visual entailment

1 Introduction

Research into neural-symbolic integration seeks to combine learning from sub-symbolic vector representations of data and concepts with symbolic reasoning and knowledge representation. [4–7]. In order to integrate the sub-symbolic neural representations of sensory data with the symbolic knowledge tools developed within AI over the last 60 years of research, a mathematical toolbox has to be designed that has the capability of translating between different levels of knowledge representation. In its infancy, by comparison, neural-symbolic studies are promising ventures towards an AI system which can recognize patterns in sensory data and reason about such commonsense patterns and knowledge.

The existence of a satisfactory dataset has been shown to be fruitful in many computer science fields. It enables a fair comparison of existing approaches and encourages competition. It should be mentioned also that benchmark datasets introduce a potential bias, as problems not covered by the benchmark receive less attention. Due to the growth of the web and abundance of data, ease of annotation by crowd-sourcing and the desire to build accurate applications, many large datasets have been developed within computer vision, such as ImageNet [1], Microsoft COCO [2] and VQA [3]. The size of these datasets is large to accommodate very complex models, specifically deep neural networks, with the promise of use as technological tools in everyday life such as image search and retrieval, or image captioning for the visually impaired.

There are valuable experimental studies in neural-symbolic reasoning, however there is a need for a common publicly-available benchmark dataset to encourage progress and communications in the field. Datasets exist in Statistical Relational Learning (SRL) and Inductive Logic Programming (ILP) which may be suitable for neural-symbolic integration. Recently developed datasets for vision-language tasks such as image caption generation and visual question answering seem attractive for neural-symbolic studies since they require complex pattern recognition over images and symbol manipulation of language. Yet, symbol manipulation and reasoning are limited to image description text that is unstructured, and not amenable to traditional natural language processing (NLP) tools. The ideal dataset for neural-symbolic studies should include a large and complex raw data set for sub-symbolic systems to learn effective and discriminative representations, as well as a formal representation of the raw data (a knowledge-base in first order logic) for symbolic systems to learn general rules and perform logical inference. Existence of both complex sub-symbolic data and its high level symbolic interpretation is essential for developing the above-mentioned translational methods between the two forms of representations which are at the heart of neural-symbolic integration.

In this paper, we propose the use of the Visual Genome dataset [13] as the best challenge benchmark dataset for neural-symbolic integration. The dataset is valuable "as is" towards the goals of neural-symbolic integration, however, we also suggest additional features and challenge tasks for the dataset to meet a wider range of research objectives within neural-symbolic computing.

In Section 2, we recall the goals of neural-symbolic integration (NSI). In Section 3, we describe the visual genome (VG) dataset. In Section 4, we list existing applications of VG to NSI. In Section 5, we propose the new applications and extensions, and in Section 6, we conclude the paper.

2 Neural-Symbolic Reasoning

Neural-symbolic systems [8] integrate logical reasoning and statistical learning by offering sound translation algorithms between network and logic models. They contain three main components: (1) knowledge encoding and reasoning in neural networks, (2) knowledge evolution and network learning, and (3) knowledge extraction from trained networks. In a neural-symbolic system, neural networks provide the machinery for efficient computation and robust learning, while logic provides high-level representations, reasoning and explanation capabilities to the network models, promoting modularity, facilitating validation and maintenance and enabling a better interaction with existing systems.

Neural-symbolic systems have had important applications in diverse areas such as bioinformatics, fraud prevention, assessment and training in simulators, cognitive robotics, general game playing, image, audio and video classification, software verification, and the semantic web. Nevertheless, a major challenge that remains is how to effectively benefit from both (i) robust statistical methods that work well on real-valued vectors and (ii) rich and interpretable represen-

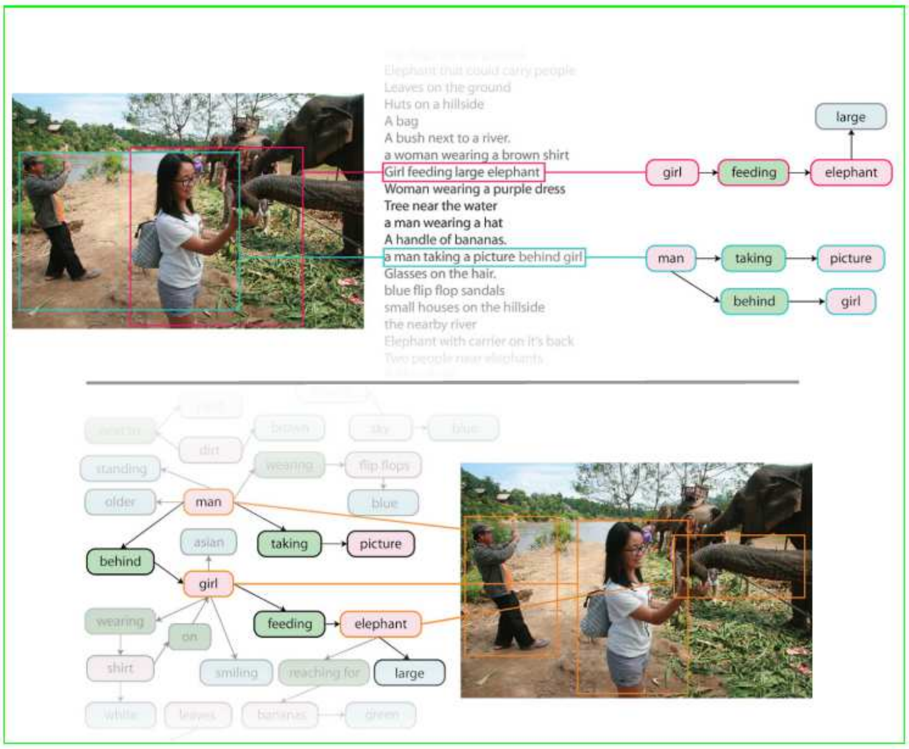


Fig. 1. From perceptual awareness to cognitive understanding of images [13]: images are annotated with numerous region descriptions, objects, attributes, and relationships, e.g.: "girl feeding large elephant" and "a man taking a picture behind girl" (top picture), with objects (e.g. elephant), attributes (e.g. large) and their relationships (e.g. feeding) described in the bottom picture.

tations which enable explanations to be reasoned about and transferred across applications. The above requires the effective translation of relational symbolic knowledge for use by statistical methods which work well with vectors (without the need for grounding all instances of the knowledge-base into the model of choice) and the effective extraction of compact and rich representations from vector-based models following neural network learning.

The emergence of symbolic representations is natural in any complex domain associated with large collections of data. In fact, symbolic representations seem critical to the solution of many interesting challenges involving big data. Consider, for example, the recent AlphaGo experiment[1] or the requirements of life-long learning[9] or intelligent agents who interact with the environment. The above is particularly relevant when neural-symbolic integration meets computer

[1] https://www.technologyreview.com/s/601072/five-lessons-from-alphagos-historic-victory/

vision. As pointed out at a recent Dagstuhl seminar on neural-symbolic computing [2], a serious challenge in the field is the lack of specifically relevant and systematic evaluation mechanisms. The benchmark-based approach, which is useful in some cases, is very limited in others, including the benchmarks used in Statistical Relational Learning (SRL) and Inductive Logic Programming (ILP) [11, 12]. In particular, when the goal is (i) to evaluate how well a system integrates learning and reasoning, or (ii) to evaluate how useful or interpretable the learned descriptions are, existing benchmarks fall short: SRL will tend to ground all representation without a focus on first-order reasoning; ILP tend not to handle real-valued vectors or provide for robust learning. Neural-symbolic systems seek to benefit from the knowledge representation and reasoning capacities of logical symbolic representations, and the robust learning capacities of neural networks, reconciling the logical nature of reasoning and the statistical nature of learning [10]. The provision of a data challenge as proposed here should promote the fair comparative evaluation of: (1) effective learning from noisy data and (2) reasoning about what has been learned.

3 Visual Genome

Visual understanding is suggested to be an AI-complete problem [17], therefore it is a challenging testbed for neural-symbolic studies. A genuine understanding of a visual scene requires detecting objects, recognizing attributes of objects and inferring their interactions and relationships. Understanding images thoroughly requires a grounding of visual concepts onto language and a formalized representation of the components of an image, as stated in [13]: "existing models would be able to detect discrete objects in a photo but would not be able to explain their interactions or the relationships between them. Such explanations tend to be cognitive in nature, integrating perceptual information into conclusions about the relationships between objects in a scene...". Going from **perceptual** to **cognitive**, from image to language, demands a range of operations that must lift the representation from subsymbolic to symbolic, which it is at the core of neural-symbolic computation studies.

Similar to previous attempts on visual knowledge bases [14–16], the Visual Genome provides a large set of images and annotations of image regions which is formalized as a scene graph of objects and their relations. Images in the dataset (see Figure 1) contain multiple image regions each having multiple object instances. The attributes of object instances and their relationship (predicate) with other objects are also recorded. Region graphs are combined to form a scene graph of an image, which can be translated into a knowledge base, as well as plain language using basic NLP tools. The concepts in the dataset can be linked to existing knowledge in other datasets or systems because all objects, attributes and relationships in each image in the Visual Genome can be mapped onto a corresponding WordNet ID, called a synset ID [18]. As described in the Visual

[2] http://www.dagstuhl.de/14381

Genome project main webpage[3], the dataset contains 108,249 images (with an average image size of 500 pixels), 4.2 million region descriptions (with around 75,000 unique image objects), 1.7 million visual question-answers, 2.1 million object instances, 1.8 million attributes (40,500 unique attributes), 1.8 million relationships (40,500 unique relationships), 1.5 million object-object relationship instances, 1.6 million attribute-object instances, 108,249 total scene graphs and 3,788,715 total region graphs.

Therefore, visual genome contains a dense formal knowledge representation of images suitable to be manipulated by symbolic computation approaches, as well as sensory image data ready to be recognized and analyzed by connectionist methods. For vision/language tasks, region descriptions and question-answer pairs related to images are also provided. Overall the dataset enables a wide range of scene understanding applications, which typically require high level symbol manipulation and language processing. Furthermore, the symbolic formalism contained in Visual Genome favors first order logic representations and relational learning. The scale of the dataset means that approaches which perform grounding will probably be less effective than truly relational approaches. In other words, Visual Genome targets a major, arguably the most important, open challenge in neural-symbolic integration: the effective handling of learning from real-valued vectors and reasoning from rich knowledge representations.

4 Existing Applications on the Visual Genome

The developers of the dataset have introduced some interesting tasks, two of which are explained below.

4.1 Attribute and Relationship Prediction

Object class prediction and object detection is at the center of computer vision studies, and successful deep learning algorithms [20, 19] dominate the field. The Visual Genome enables dense and accurate attribute/predicate estimation; bounding boxes that contain an object can be analyzed for predicting attribute/predicate dimensions.

Researchers have found that learning attribute-object class pairs for each bounding box dramatically improves attribute prediction performance possibly due to the unique association of some attributes with specific object classes. Similarly, learning *subject class - predicate - object class* triplets instead of *predicate* only, can improve performance. This is again due to the fact that some relationships occur only among a very small subset of objects classes (e.g. the *drive* predicate accepts the *person* subject exclusively). Such applications can be considered an instantiation of collective classification in relational learning [32].

[3] https://visualgenome.org/

4.2 Caption Generation and Visual Question Answering

The existence of region descriptions and question-answer pairs on images facilitate vision-language processing tasks. The visual representation of images and regions can be used in a generative architecture to produce syntactically and semantically correct text such as automated image caption generation. Recurrent neural network algorithms have been deployed successfully [21] for such vision-language applications. However, a major challenge has been judging performance accuaracy of automated image captioning, e.g. is "A cat is beside a dog under a parked car" the same as "A car is parked over a dog and a cat"?

5 Suggested Applications and Extensions

Visual Genome holds a very rich representation of the visual world, ready to be exploited by cognitive tasks. We envision that the dataset can be used for a wide set of experimental paradigms, or can be extended by additional crowd-sourced annotations as required. We provide a set of novel tasks, which is not meant to be exhaustive. Along with the task definitions, we provide a high level algorithmic description of how to tackle them in order to illustrate how neural-symbolic studies would benefit from the dataset.

Generally, neural-symbolic approaches would ground the sensory data onto symbols and manipulate those, or perform vector algebra on neural representations to form a hierarchy of concepts and rules on the vector space. The main questions are how to accurately and effectively ground the data or how to manipulate the vectors as done with symbols in AI, as well as how to use both mathematical tools simultaneously.

5.1 Visual Entailment

Comprehension of entailment and contradiction in sentences is an important part of language processing. In textual entailment tasks, two sentences need to be understood and the system has to decide whether they contradict each other, they are neutral (unrelated) or they entail each other. The scene graph in Visual Genome is already a valuable asset in the textual entailment task, as utilized in a study in [22], yet there is much more to be done. We propose a new task called visual entailment in which images, relationships and scene graphs are used to detect entailment and contradictions. This is a very natural use of the image representation for neural-symbolic tasks: inference can be performed at the symbolic level if images are grounded onto class and attribute predictions by a classifier, or inference can be partly done at the sub-symbolic level using the neural representations of images. Sub-symbolic computation requires an algebra on semantically meaningful vector representations [33].

We present two image bounding boxes, then ask whether there is entailment/contradiction/neutralism. The decision is very much related to the possible relationships between image boxes. If there is a relationship then the answer is entailment, if not, it can be neutral or contradiction, depending on the compatibility with commonsense. A *car* and a *tire* imply entailment, a *car* and a *house window* may be neutral but a *car* and a *kitchen sink* is probably a contradiction. The output can be set to a range between -1 (contradiction) and 1 (entailment), at which point the supervised learning may become a regression task instead of classification. It should be noted that visual entailment aims at finding relationships between two scenes thus the proposed task is closely related to link prediction in relational learning, where the goal is to learn the existence of a relationship. Therefore, the idea of contradiction in visual entailment means learning the lack of a relationship, which is not the case in textual entailment task.

The task becomes even more interesting and similar to textual entailment if we allow one or two of the image boxes to be a large region with multiple objects and relationships in it. Then the system needs to analyze the congruence of region graphs, hence knowledge bases. A subsymbolic approach would use neural embeddings of the image boxes to generate rules of entailment on the vector space possibly using a vector symbolic architecture [23, 24] and/or an attention-memory computation framework [25]. A symbolic approach would use the class/attribute/relationship predictors to go up to knowledge base level.

5.2 Scene Graph Estimation

Possibly the hardest task is generating the scene graph of an image because the graph holds the complete high level information regarding the image, we need to go from the sensory to the most complete cognitive level. It requires to focus on specific bounding boxes in the image, estimate object/attribute labels and jump to other image boxes while predicting relationships between them. Thus the graph can be built part by part possibly with multiple passes on the same image region. These multiple passes can possibly be hierarchical in nature, extracting graph structure from coarse to fine details. This workflow resembles the strategy of recurrent architectures with attention-memory mechanisms[26]. Another strategy more in the flavor of neural-symbolic computation would be training the system by encoding regions and scenes in the training dataset with fixed length vector representations and forming a "graph knowledge-base", then matching the test region with the knowledge base to obtain the most representative and similar region description in the training set. After this initial estimation, fine-tuning can optionally be done with the recurrent architectures with attention-memory mechanisms.

The main challenge in this task is related to the variable binding problem: multiple instances of the same object/concept/relationship as it appears in different times and context need to reuse a common function with possibly different values. One possible solution to this problem is transferring learned representation across different contexts [28].

5.3 Visual Rule Extraction and Analogy

Is it possible to mine the scene graphs for extracting logical clauses such as "If **Man** not(**Standing**) Then **Man SitsOn**(**Something**)"? This capability is essential for forming the visual commonsense knowledge mentioned earlier. In a similar flavor, visual analogies can be made such as "**Leg** is to **Man** as **Tire** is to **Car**". These are strictly in the domain of symbolic computation when images are grounded to class/attributes and predicate predictions are processed in the scene graph. However, what if we wanted to retrieve rules and analogies directly using image portions? Then, neural representations of images would need to be processed to harvest conditional and analogical "statements" at the sub-symbolic level [27, 29, 34]. The rules and analogies that form the commonsense knowledge and representations of the images are expected to live on the same space, which is essential for combining connectionist and symbolic capabilities. Visual rule extraction can also be tackled with inductive bias transfer of neural networks across different task domains [30]. More interesting approaches would be again hybrid ones that utilizes the symbolic mechanisms along with vector algebra.

5.4 Collective Classification

Another relevant relational learning task is collective classification: simultaneous prediction of the class of several object bounding boxes in a region given their attributes or relations. This is superficially similar to attribute and relation prediction tasks already examined in [13], yet the proposed task is not bounded by pairwise bounding box queries but all the objects in a region or even in a whole image can be considered for a more challenging collective classification. This is directly related with multiple task learning and inductive bias transfer between many tasks, as studied from a neural-symbolic perspective in [31].

5.5 Unsupervised co-training of a subject class - predicate - object class using images and symbols

Related to prior work discussed in Section 4.1 is the unsupervised co-training of subject class - predicate - object class triples using both image data as well as symbolic logic. The intention is to show that one can learn an unsupervised generative model (e.g. stacked Restricted Boltzmann Machines) that are capable of reconstructing the images given the symbols, and the symbols given the images. Here, symbols could be represented as combinations of textual inputs or as images themselves.

6 Conclusion

We have proposed Visual Genome as a challenge and benchmark dataset for neural-symbolic integration. Along with the original tasks that were suggested by the Visual Genome creators, we also identify tasks specific for neural-symbolic integration, in particular combining learning from real-valued vectors and reasoning from rich relational knowledge representations, to promote research in the field and competition between lab groups.

Acknowledgments. We would like to thank the reviewers for detailed and very beneficial comments on the paper. Ozgur Yilmaz is supported by The Scientific and Technological Research Council of Turkey (TUBİTAK) Career Grant, No: 114E554.

References

1. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.
2. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014*, pages 740–755. Springer, 2014.
3. Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433, 2015.
4. Tarek R Besold and Kai-Uwe Kühnberger. Towards integrated neural–symbolic systems for human-level AI: Two research programs helping to bridge the gaps. *Biologically Inspired Cognitive Architectures*, 14:97–110, 2015.
5. Artur S. d'Avila Garcez, Luis C Lamb, and Dov M Gabbay. *Neural-symbolic cognitive reasoning*. Springer Science & Business Media, 2008.
6. Sebastian Bader and Pascal Hitzler. Dimensions of neural-symbolic integration-a structured survey. *arXiv preprint cs/0511042*, 2005.
7. Artur S. d'Avila Garcez, Tarek R Besold, Luc de Raedt, Peter Földiak, Pascal Hitzler, Thomas Icard, Kai-Uwe Kühnberger, Luis C Lamb, Risto Miikkulainen, and Daniel L Silver. Neural-symbolic learning and reasoning: contributions and challenges. In *Proceedings of the AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches, Stanford*, 2015.
8. Artur S. d'Avila Garcez, Luís C. Lamb, and Dov M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Cognitive Technologies. Springer, 2009.
9. Daniel L Silver, Qiang Yang, and Lianghao Li. Lifelong machine learning systems: Beyond learning algorithms. In *in AAAI Spring Symposium Series*. Citeseer, 2013.
10. Leslie G. Valiant. Knowledge infusion. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 1546–1551, 2006.
11. Jue Wang and Pedro M. Domingos. Hybrid markov logic networks. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1106–1111, 2008.
12. Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*. Synthesis

Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2016.

13. Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalanditis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016.
14. Yuke Zhu, Ce Zhang, Christopher Ré, and Li Fei-Fei. Building a large-scale multimodal knowledge base system for answering visual queries. *arXiv preprint arXiv:1507.05670*, 2015.
15. Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Neil: Extracting visual knowledge from web data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1409–1416, 2013.
16. Fereshteh Sadeghi, Santosh K Divvala, and Ali Farhadi. Viske: Visual knowledge extraction and question answering by visual verification of relation phrases. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1456–1464. IEEE, 2015.
17. Dafna Shahaf and Eyal Amir. Towards a theory of ai completeness. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, pages 150–155, 2007.
18. George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
19. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision andpattern recognition*, pages 580–587, 2014.
20. Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 2009.
21. Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.
22. Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.
23. Simon D Levy and Ross Gayler. Vector symbolic architectures: A new building material for artificial general intelligence. In *Proceedings of the 2008 conference on Artificial General Intelligence 2008: Proceedings of the First AGI Conference*, pages 414–418. IOS Press, 2008.
24. Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *ACL*, pages 236–244, 2008.
25. Ivo Danihelka, Greg Wayne, Benigno Uria, Nal Kalchbrenner, and Alex Graves. Associative long short-term memory. *arXiv preprint arXiv:1602.03032*, 2016.
26. Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439, 2015.
27. Ozgur Yilmaz. Symbolic computation using cellular automata-based hyperdimensional computing. *Neural computation*, 2015.
28. Daniel L Silver. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connection Science*, 8(2):277–294, 1996.

29. Ozgur Yilmaz. Analogy making and logical inference on images using cellular automata based hyperdimensional computing. In *Advances in Neural Information Processing Systems, Cognitive Computation Workshop*, pages 1–9, 2015.
30. Daniel L Silver. Selective functional transfer: Inductive bias from related tasks. In *IASTED International Conference on Artificial Intelligence and Soft Computing (ASC2001)*. Citeseer, 2001.
31. Daniel L Silver and Liangliang Tu. Image transformation: inductive transfer between multiple tasks having multiple outputs. In *Advances in Artificial Intelligence*, pages 296–307. Springer, 2008.
32. Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher and Tina Eliassi-Rad. Collective Classification in Network Data. AI Magazine, 3(29):93–106. 2008.
33. Luciano Serafini and Artur S. d'Avila Garcez. Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge. *arXiv preprint arXiv:1606.04422*, 2016.
34. Tarek Richard Besold, Kai-Uwe Kühnberger, Artur S. d'Avila Garcez, Alessandro Saffiotti, Martin H. Fischer and Alan Bundy. Anchoring Knowledge in Interaction: Towards a Harmonic Subsymbolic/Symbolic Framework and Architecture of Computational Cognition. Artificial General Intelligence - 8th International Conference, AGI 2015, AGI 2015, Berlin, Germany, July 22-25, 2015.

High-Power Logical Representation via Rulelog, for Neural-Symbolic
(position paper, extended abstract)

Benjamin N. Grosof

Coherent Knowledge
Mercer Island, Washington, USA
`http://coherentknowledge.com`
email: `firstname dot lastname at coherentknowledge dot com`

1 Introduction

Combining neural networks (NN) methods with symbolic methods is an area that is exciting in terms both of basic research and practical application. We discuss here the opportunities and challenges involved in combining NN ("neural") with logical knowledge representation and reasoning (KRR) that has high expressive power and fundamental scalability ("high-power" KRR). We describe how Rulelog KRR fits pretty well these challenges. (Note "logical" here includes probabilistic.)

2 Why to Combine KRR with NN and ML, generally

The domain-independent core of artificial intelligence (AI) consists of both KRR and machine learning (ML), as has been widely recognized within the AI research community since at least the 1980s.

There are a number of ways in which it is useful, or even required, to combine KRR methods with ML methods. The *prediction* step of ML requires reasoning. The *target* of ML is a representation. Getting business value from ML usually requires reasoning for analysis and decisions. KRR is required to effectively *combine* the results of ML from multiple ML episodes, sources, or methods. KRR is required to *accumulate* knowledge *coherently*, as knowledge ongoingly originates from ML (as well as from non-ML origins). KRR is required to *explain* knowledge understandably. Explanations are often part of required or desired analysis functionality for their own sake; they are also needed for humans to trust an automated system, check and debug knowledge/reasoning, and to help ask drill-down follow-up questions. ML-based technology today often – arguably, usually – has weaknesses in regard to such effective combination, coherent accumulation, and understandable explaining. Reasoning is useful to *supply derived facts* for ML to chew on. Reasoning is useful to *focus* ML's tasks and conjecture schemas: e.g., to provide sets of relevant features (perhaps weightedly) and/or important questions (reasoning "(sub)goals"), so as as to drive ML. Last but not least, *humans know stuff beyond what is available via ML training data*, and such knowledge is often pretty complex to state. Programming is expensive, so KRR methods are often a more cost-effective approach to entry/capture of such knowledge.

3 Symbolic Side's Representational Challenges and Requirements

For the (KRR-based) symbolic side to combine most effectively with the neural side, there are a number of representational challenges and requirements. Highly flexible expressiveness is needed; ideally any kind of knowledge can be ingested by the KRR. Thus it should be equipped feature-wise with *higher-order* syntax, logically *quantified formulas*, and strong *meta* (statements about statements). The KRR should represent *numeric weighting and uncertainty*, including (but not limited to) probabilistic and fuzzy. The KRR should treat the *evolving* character of knowledge and of the world; technically, it should have the *defeasibility* feature. The KRR should provide reasoning that is *deep*, including (but not limited to) multi-step in its logical chaining. The KRR should be *scalable* despite being expressive: not only computationally scalable to large amounts of asserted and concluded knowledge ("volume" and "velocity") but also "socially scalable" in regard to the diverse multiplicity of ML/other info sources, algorithmic methods, and underlying data samples ("variety").

4 Rulelog KRR Technology, its Advantages and Limitations

Rulelog methods meet the above set of neural-driven representational challenges pretty well overall: not only pareto-optimally among the set of available KRR approaches, but arguably better than any other KRR approach. Rulelog methods are especially strong on the meta and higher-order features, while providing scalability. They are also well suited to orchestrating / federating multiple knowledge sources and components so as to assemble and compose multiple analysis results.

Rulelog is a leading approach to semantic rules KRR. It is expressively powerful, computationally affordable, and has capable efficient implementations. A large subset of Rulelog is in draft as an industry standard[1] to be submitted to RuleML[2] and W3C[3] as a dialect of Rule Interchange Format (RIF) [1, 2].

Rulelog [4] extends database logic and well-founded declarative logic programs (LP) with:

- strong meta-reasoning, including higher-order syntax (Hilog) and rule ids (within the logical language);
- explanations of inferences;
- efficient higher-order defaults, including "argumentation theories";
- flexible probabilistic reasoning — including distribution semantics and evidential probability;
- bounded rationality, including restraint — a "control knob" to ensure that the computational complexity of inference is worst-case polynomial time;
- "omni-directional" disjunction and existential quantifiers in the rule heads;
- object-orientation and frame syntax, which subsumes RDF triples;

[1] `http://ruleml.org/rif/rulelog/rif/RIF-Rulelog.html`
[2] `http://www.ruleml.org`
[3] `http://www.w3.org`

– sound tight integration of first-order-logic ontologies including OWL; and several other lesser features, including aggregation operators and integrity constraints.

Probabilistic reasoning and tight integration with (inductive) ML is a key area of recent technology progress and ongoing R&D on Rulelog.

Rulelog also combines closely with natural language processing (NLP), in the Textual Rulelog approach [5], so as to support: human authoring of knowledge; mapping between different info schemas and terminologies; and explaining conclusions. This is another key area of ongoing R&D on Rulelog.

Implementation techniques for Rulelog inferencing include transformational compilations and extensions of *tabling* algorithms from logic programming. “Tabling” here means smart caching of subgoals and conclusions together with incremental revision of the cached conclusions when facts or rules are dynamically added or deleted [7, 8]. “Tabling” is thus a mixture of backward-direction and forward-direction inferencing.

There are both open-source and commercial tools for Rulelog that vary in their range of expressive completeness and of user convenience. They are interoperable with databases and spreadsheets, and complement inductive machine learning and natural language processing techniques. The most complete system today for Rulelog is Ergo[4], a commercial platform suite from Coherent Knowledge[5]. Ergo Lite, a.k.a. Flora-2[6], is an open source system that implements a significant subset of Rulelog reasoning. Ergo’s ErgoText feature is a commercial realization of Textual Rulelog.

Rulelog has some important limitations. One is that it lacks “reasoning-by-cases”, a.k.a. it is “intuitionistic”; it only concludes a disjunction if it concludes one of the disjuncts. Another limitation is that Rulelog methods are not yet optimized for probabilistic reasoning.

As has been extensively discussed in the ML and KRR literature, classical logic (e.g., first-order logic but also higher-order logic) has reasoning-by-cases but is brittle in the face of conflicting/evolving knowledge, and lacks computational scalability. Markov Logic Networks [6] are attractively flexible and principled in their ability to represent probabilistic/weighted knowledge, but are much less computationally scalable than Rulelog. Answer Set Programs [3] have reasoning-by-cases and are much less brittle than classical logic, but lack computational scalability and many of Rulelog’s expressive features. For reasons of focus, we refrain from giving here additional comparison to other KRR approaches.

Applications to date of Rulelog technology include a wide range of tasks and domains in business, government, and science. Examples include: legal/policy compliance, particularly in financial services; personalized tutoring about science; and e-commerce marketing. Rulelog shines especially on representing, and deeply reasoning with, complex commonly-arising kinds of knowledge such as: mappings among terminologies, ontologies, and data schemas; policies, regulations, and contracts; and causal pathways (e.g., in science).

[4] `http://coherentknowledge.com/ergo-suite-platform-technology/`
[5] `http://coherentknowledge.com`
[6] `http://http://flora.sourceforge.net`

5 Future Research Directions, including Applications

An immediate direction for future R&D is to hook up Rulelog implementations to NN systems. There are many potential applications for this combination of Rulelog KRR with NN – and/or with other ML. One realm is compliance and fraud. Another realm is NL understanding in intelligence analysis and in search.

There are a number of other interesting future research directions in terms of core technology and experiments, per the overall discussion in section 1. Next, we highlight a few of these directions. One is to *feed derived data* from Rulelog to NN. Another is to combine the results of neural with other ML and structured info, including human-authored complex knowledge, e.g., that started life as English sentences. Terminology mappings and source trustworthiness are two interesting kinds of such complex knowledge represented in Rulelog. A third direction is to combine NN *word-vector* distributed representations with Textual Rulelog.

Related future directions for research on Rulelog KRR itself include to optimize its reasoning with probabilistic/weighted knowledge, and to extend Rulelog's expressiveness to *selective* reasoning-by-cases.

References

1. Boley, H., Kifer, M.: RIF Basic Logic Dialect (February 2013), `http://www.w3.org/TR/rif-bld/`, W3C Recommendation. `http://www.w3.org/TR/rif-bld/`
2. Boley, H., Kifer, M.: RIF Framework for Logic Dialects (February 2013), `http://www.w3.org/TR/rif-fld/`, W3C Recommendation. `http://www.w3.org/TR/rif-fld/`
3. Gelfond, M.: Answer Sets. In: van Harmelen, F., Lifschitz, V., , Porter, B. (eds.) Handbook of Knowledge Representation, pp. 285–316. Elsevier, Amsterdam (2008)
4. Grosof, B.N., Kifer, M., Fodor, P.: Powerful Practical Semantic Rules in Rulelog: Fundamentals and Recent Progress (July 2015), `http://2015.ruleml.org/tutorials.html`, Conference Tutorial (1.5 hours). Extended abstract is in the Symposium's Proceedings (Springer-Verlag), and contains numerous references. Also available at `http://coherentknowledge.com/publications/`.
5. Grosof, B.: Rapid Text-Based Authoring of Defeasible Higher-Order Logic Formulas, via Textual Logic and Rulelog. In: Morgenstern, L., Stefaneas, P., Lvy, F., Wyner, A., Paschke, A. (eds.) Theory, Practice, and Applications of Rules on the Web, Lecture Notes in Computer Science, vol. 8035, pp. 2–11. Springer Berlin Heidelberg (2013), `http://dx.doi.org/10.1007/978-3-642-39617-5_2`
6. Richardson, M., Domingos, P.: Markov Logic Networks. Machine Learning 62, 107–136 (2006)
7. Swift, T., Warren, D.: XSB: Extending the Power of Prolog using Tabling. Theory and Practice of Logic Programming 12(1-2), 157–187 (September 2012)
8. Swift, T.: Incremental Tabling in Support of Knowledge Representation and Reasoning. TPLP 14(4-5), 553–567 (2014), `http://dx.doi.org/10.1017/S1471068414000209`

Heterotic Continuous Time Real-valued/Boolean-valued Networks

Daniel R. Patten[1] and Howard A. Blair[2]

[1] USAF Research Laboratory, Rome NY USA
`drpatten@syr.edu`
[2] EECS, Syracuse University, Syracuse, NY USA
`blair@ecs.syr.edu`

Keywords: heterotic, differential, dynamical system, convergence space

1 Extended abstract

Heterotic models of computation were introduced in 2012 by Stepney et al. in [2011]. Heterotic models of computation *seamlessly* combine computational models such as classical/quantum, digital/analog, synchronous/asynchronous, imperative/functional/relational, etc. to obtain increased computational power, both practically and theoretically.

Although much greater generality is possible – we have previously reported on heterotic quantum/classical dynamical systems, [2014] – we here concentrate on heterotic dynamical systems that are given by continuous time real-valued/boolean-valued networks, which in the sequel we refer to as a heterotic Boolean network (HBN). A network of this kind is a finite directed graph with a reflexive edge relation where each vertex is of type **real** or of type **boolean**. Each vertex updates its value in continuous time according to a dynamics specified by a set of *autonomous* first-order differential equations

$$\left\{dx_i/dt \;=\; f_i(x_1,\ldots,x_n) \mid i=1,\ldots,n\right\} \tag{1}$$

where each variable x_i corresponds to a vertex in the network.

Nearly all continuous time dynamical systems can be expressed as a system of 1st-order differential equations provided a differential calculus satisfying a functorial chain rule is available for functions mapping between the spaces corresponding to the types of the variables involved in the system. The purpose of the proposed differential calculus in this application to HBN's is to allow for the uniform and seamless specification of an HBN via such systems of 1st-order differential equations. Since the calculus is mathematically rigorous it provides a formal semantics for such specifications and therefore a rigorous basis for verification and validation of an HBN with respect to those specifications. In the HBN applications we need to have a differential calculus that agrees with the familiar calculus on Euclidean spaces that extends to functions mapping reals to Booleans, and Booleans to Booleans. We use the apparatus of *convergence spaces*, [2016].

To conservatively extend the notion of differentiation to general convergence spaces, we note first that the set of continuous functions from X to Y which we denote here by Y^X has a convergence structure uniformly constructible from the convergence structures on X and Y such that the category of convergence spaces is Cartesian-closed; the details of the convergence structure on the exponential spaces Y^X need not concern us at present; it is enough that we have convergence structures available on the exponential spaces sufficient for obtaining a chain rule. Also for the present application of convergence spaces differences $x_0 - x$ are needed; they are constructible from an Abelian regular action on the convergence space, [1995], which renders the space as a module over a ring. Modules are nearly vector spaces where non-zero scalars are not guaranteed to have inverses.

Suppose convergence spaces Y and X are each equipped with regular actions and the sum and difference of pairs of points in X and Y have been determined. Then choose a subspace $\mathsf{Diff}(X,Y)$ of Y^X to serve as values of the derivative operation on these functions spaces. For example, we choose $\mathsf{Diff}(\mathbb{R}^m,\mathbb{R})$ to be the space of linear functionals on $\mathbb{R}^m$, and similarly for $\mathsf{Diff}(\mathbb{B}^m,\mathbb{B})$. Then $g \in \mathsf{Diff}(X,Y)$ is a *differential* of f at x_0 iff for every filter $F \downarrow_X x_0$ there is a filter $G \downarrow_{Y^X} g$ such for each $W \in G$ there is a $V \in F$ such that for each $x \in V$, there is $h \in W$ such that $h(x - x_0) = f(x) - f(x_0)$.

In this application we also need to choose $\mathsf{DIff}(\mathbb{R},\mathbb{B})$. We identify $\mathbb{B}$ with $\{0,1\}$ with the indiscrete convergence structure. We take $\mathsf{Diff}(\mathbb{R},\mathbb{B})$ to be the discrete space of the following four functions: (1) the constant function mapping all real number to 0, (2) the "step" function$f(x) = (x \le 0)\ ?\ 0 : 1$, (3) $f(x) = (x \ge 0)\ ?\ 0 : 1$ and (4) $f(x) = (x = 0)\ ?\ 0 : 1$. Under these definitions the system (1) is well-defined, but may or may not have a solution. After all, even with a rigorous denotational semantics, not every syntactically correct program in an ordinary programming language has a solution in the sense that it will produce a well-defined trajectory (i.e. no run-time errors.)

We conclude with a small example of an HBN:

$$dx/dt = ((b = 1)\ ?\ [-\sin x] : [\sin x]), \quad dy/dt = ((b = 1)\ ?\ [\cos x] : [-\cos x])$$
$$db/dt = (\mathsf{prime}(t)\ ?\ (\lambda t'.((t' \le t)\ ?\ 0 : 1) : \lambda t'.0)$$

The solutions to this equation have the boolean value of b changing whenever t is a prime nonnegative integer and have point given by cordinates (x, y) traversing a circle of radius 1 centered at the origin. The point reverses direction whenever t is prime.

References

2011. S. Stepney, V. Kendon, P. Hines and A. Sebald: "A framework for heterotic computing", *8th International Workshop on Quantum Physics and Logic (QP L 2011)*, 2012, 263–273

2014. D.R. Patten, H.A. Blair and P.M. Alsing: "Heterotic quantum dynamical systems: an application of differential calculus on digraphs", *14th Haifa Workshop on Interdisciplinary Applications of Graph Theory, Combinatorics and Algorithms*, Haifa, Israel

2009. S. Rudeanu. "What can be expected from a Boolean derivative," *Analele Stiintifice ale Universitatii Ovidius Constanta* Vol. 17(1), 2009, 177–186

2016. L. Nel. *Continuity Theory*. Springer, 2016.

2015. Patten et.al., "Differential Calculus on Cayley Graphs". arXiv 1504.08013.

1995. J. Rotman. *An Introduction to the Theory of Groups (Graduate Texts in Mathematics).*