Conference on
Information Technology and
Data Science

Possible Neural Models to Support the Design of Prime Convo Assistant*

Norbert Bátfai, Máté Szabó

Faculty of Informatics, University of Debrecen, Hungary
batfai.norbert@inf.unideb.hu
szabo.mate@inf.unideb.hu

Proceedings of the 1st Conference on Information Technology and Data Science
Debrecen, Hungary, November 6–8, 2020
published at http://ceur-ws.org

Abstract

The Prime Convo Assistant initiative is a software development idea intended to examine how we could use the automatic and interactive theorem provers and machine learning methods to generate automatically new sentences in an artificial visual language. The name Prime Convo Assistant is a combination of the Prime Radiant and the internal conversation with ourselves. Isaac Asimov's psychohistorians used the Prime Radiant device to store psychohistorical equations. The internal conversations come from Julian Jaynes' theory of bicameral mind. Our idea is that the sentences of the visual language in question are initially given in the form of first-order logic formulas. In our previous work, Pasigraphy Rhapsody, we used first-order logic formulas to create visual objects. In the framework of the present work, we primarily conduct literature research and test existing models. On the one hand, in the field of what neural models exist whose input is a first-order logic corpus, and on the other hand, in the field of what deep learning-based solutions help the operation of automatic theorem provers. In addition, in a broader context, we examine the possible relationship between Society 5.0 and esport culture from a kind of robopsychological and robophilosophical point of view.

*This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

Keywords: Robopsychology, robophilosophy, machine learning, automated theorem proving, Society 5.0, esport

1. Introduction

The initiative called Pasigraphy Rhapsody [6] (PaRa) aims to create a first-order logic-based artificial graphical language. In our preliminary experiments, the logic formulas are represented by n-dimensional dotted hypercubes, the dotting of the hypercubes has been inspired by our previous work [7] as it can be seen in Fig 1. The idea of using yslant and xslant to achieve a 3D effect like appearance is based on Stefan Kottwitz's example [17], see [6]. Our motivation was to invent a Minecraft-like builder game based on such hypercubes.

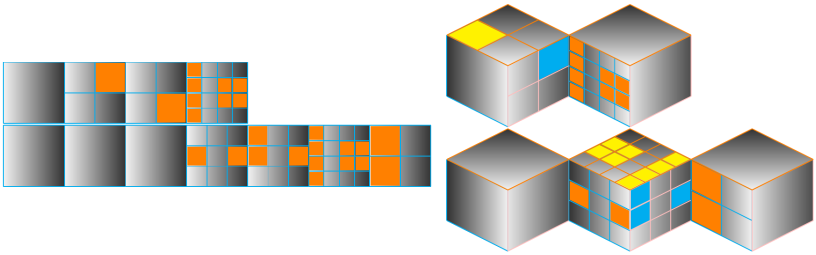


Figure 1. A detail of the PaRa formalization of Lord's Prayer in 2 and 3 dimensional LuaLaTeX visualization.
(Source: `https://gitlab.com/nbatfai/pasigraphy-rhapsody`)

2. Related Works

There are many languages based on first-order logic and these can be used in modern applications. The combination of logic and machine learning is not rare, we can see examples like extending decision trees to first-order logic [18], clustering and instance based learning [23]. There are many first-order logic based applications and languages like Col, a language for complex objects [1], DLP, a language for databases [22], S-TLA+ for computer forensic investigation [24], or knowledgebases using HiLog [21].

3. Visualization of First-Order Logic

To introduce Pasigraphy Rhapsody's first-order logic visualization, we will use the sentence s = Mice hate cats, which can be formalized as

$$\forall \text{Animal.x} \forall \text{Animal.y} (\text{P.Mouse(x)} \land \text{P.Cat(y)} \supset \text{P.Hate(x,y)}).$$

Numbers are easier to visualize than formulas, so the first step will be to convert these formulas to number using the Pasigraphy Rhapsody's numeration system, which is detailed in 4.2. The conversion between formalized objects and PaRa numerical codes can be seen in Table 1.

Table 1. Conversion of formalized objects to PaRa numerical codes.

Formalized objects	PaRa numerical code
Animal.x	32
Animal.y	96
P.Cat	10
P.Mice	14
P.Hate	18

The next step is to create coordinates from these PaRa numerical codes, so they can be placed on a face of a hypercube. For this, we will use the following notation: l × l is the size of the image, k is the number of dots, p is the assigned PaRa numerical code, c denotes the coords of the dots in the form SMNIST(p) = {c}. We enumerate all k-combinations of l^2 pixels. These combinations can be seen in Figure 2.

```
n: 4 k: 1 p: 1, c: 1. (0,0)
n: 4 k: 1 p: 2, c: 1. (1,0)
n: 4 k: 1 p: 3, c: 1. (0,1)
n: 4 k: 1 p: 4, c: 1. (1,1)
*****
n: 4 k: 2 p: 5, c: 1. (0,0) 2. (1,0)
n: 4 k: 2 p: 6, c: 1. (0,0) 2. (0,1)
n: 4 k: 2 p: 7, c: 1. (0,0) 2. (1,1)
n: 4 k: 2 p: 8, c: 1. (1,0) 2. (0,1)
n: 4 k: 2 p: 9, c: 1. (1,0) 2. (1,1)
n: 4 k: 2 p: 10, c: 1. (0,1) 2. (1,1)
*****
n: 4 k: 3 p: 11, c: 1. (0,0) 2. (1,0) 3. (0,1)
n: 4 k: 3 p: 12, c: 1. (0,0) 2. (1,0) 3. (1,1)
n: 4 k: 3 p: 13, c: 1. (0,0) 2. (0,1) 3. (1,1)
n: 4 k: 3 p: 14, c: 1. (1,0) 2. (0,1) 3. (1,1)
*****
n: 9 k: 1 p: 15, c: 1. (0,0)
...
```

Figure 2. The enumeration of relevant k-combinations of 2^2. All relevant k-combinations of l^2 is based on $\sum_{l=2}^{\infty}\sum_{k=1}^{l^2-1}\binom{l^2}{k}$.

Now that we have the coordinates, we can use the Pasigraphy Rhapsody's tiling language to create cube faces from these. The transformation from formalized object to cube face can be seen in Figure 3.

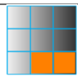
Formalized object	PaRa num code	SMNIST code
Animal.x	32	
Animal.y	96	
P.Mouse	10	

Figure 3. Transformation from formalized object to SMNIST code.

After that we replace the formula's parentheses with indentation, so we can determine the location of the formalized objects on the hypercubes and the order of these cubes. This step can be seen in Figure 4.

$\forall$	x	$\forall$	y		
	M	x	$\wedge$	C	y
	$\supset$	H	x	y	

Figure 4. Indentation of formalized objects.

The formalized objects will be replaced with numerical codes as it can be seen in Figure 5.

2	32	2	96		
	10	32	4	14	96
	6	18	32	96	

Figure 5. Numerical form of formalized objects.

With the location of formalized objects and their numerical codes, we can create the dotted faces of the hypercubes as it can be seen in Figure 6.

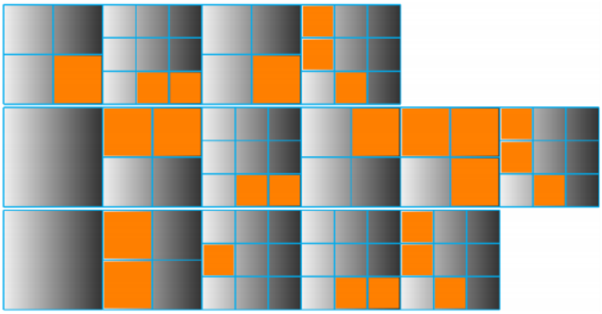


Figure 6. Visualized formalized objects.

The last step is to form hypercubes from these faces. Only the first three face will form a hypercube, so the s = Mice hate cats sentence will generate 3 cubes which can be seen in Figure 7.

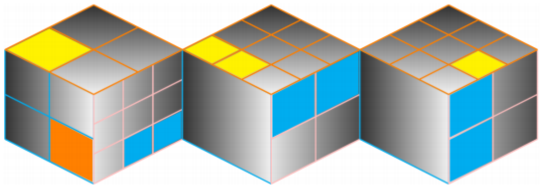


Figure 7. Generated hypercubes from "Mice hate cats" sentence.

4. Prime Convo Assistant

To develop Pasigraphy Rhapsody, in our previous work [5] we proposed to formalize standard artificial intelligence tasks (like monkey and banana [14], where there is a monkey in a room and the bananas are suspended from the ceiling, and the task is to knock bananas down using a stick and a chair) or to create a game where players formalize their everyday activities. But these are very slow processes if the latter is possible at all. Therefore, in this work, we raise the issue of examining the possibility of automatic conversion of large corpora to Pasigraphy Rhapsody such as converting the Lean Mathlib library [2, 12]. The goal of using a game to formalize sentences is to reach the widest possible user community. That is important from two main aspects. On the one hand, an esport game can shape the thinking of players widely, namely to teach modern mathematical logic to them, but it can also be interesting to compare Society 5.0 [15] and esport culture. On the other hand, we could build large corpus in this way. In this work we move away from looking for possible games but we try to fulfill the latter sub-goal with a different method. In an axiomatic system in principle we can automatically generate new sentences easily. The Lean Mathlib library a such axiomatic system. So it is worth trying to formalize it in PaRa. The first step in the PaRa formalization is to create the first-order logic form of the investigated sentences. However, the following steps has already been fully automated. Therefore, in principle, such a conversion may be possible. With PaRa conversion, our goal is to study the possible neural models that receive the same input in parallel, but one model gets its input in visual (as PaRa images) form and the other in textual (as first-order logic formulas) form. In this sense the conception of Prime Convo Assistant, introduced in [5], would be transformed to a such system that has the following use cases. 1. Translating the Lean corpus to PaRa 2. Creating new sentences from the Lean logical corpus on the one hand and from the converted visual (PaRa) images on the other hand.

4.1. Possible Neural Models

First-order Logic Input Our first question that what neural models exist whose input is a first-order logic corpus. Are there such BERT [13] based systems? It is pre-trained using masked language modeling, and there is a fine-tuning step where it is trained to specific tasks. BERT was tested on the General Language Understanding Evaluation benchmark, where the base version and the large version scored 79.6 and 82.1 points. On the Stanford Question Answering Dataset, BERT outperformed the nlnet and QANet ensemble systems in the terms of F1 score. Another question is whether a soft theorem prover, such as ROBERTa [11] could be applied to first-order logic input. Its ability to classify a statement with a context true or false was tested on generated datasets. The results showed that its accuracy was 99%. The first trained model with 0 depth was unable to answer reasoning questions. However, models with larger depth performed 97.6%, but they required more training data. Although logic input is rather rare in machine learning, there are some models that support it. Because of this rare kind, most machine learning libraries do not support logic, which means that we have to extend existing models. There are many solutions that would fit Prime Convo Assistant's like natural language understanding with Rasa [8] or multi-task deep neural networks [20], which performed a bit better on GLUE test than BERT with 82.7 score, or logic reasoning with logic-integrated neural networks (LINN) [26], which can solve logical equations with 94.4% accuracy, while recurrent or convolutional networks achieve only 64%, or deep reasoning networks (DRN) [10]. The accuracy comparison of DRNets and CapsuleNet on Multi-MNIST-SUDOKU dataset shows that DRNets with reasoning modules outperformed simple DRNets and CapsuleNet as their accuracy was 100% and 99.99%, while the capsule network's was 88.46%.

Pasigraphy Rhapsody image input Beside first-order logic, Prime Convo would use models with Pasigraphy Rhapsody image inputs. These are special kind of images because it contains 3D cubes with a part of first-order logic formula on it. These formulas were transformed to a numerical form, and we form coordinates based on this representation. Each number will be placed on a cube's side. Our task was to find suitable models to recognize these images. There are many models we can use, but they must interpret the relative positions of the coordinates.

Most libraries support convolutional neural network [19], which can recognize images. The biggest downside of it is that it will look for smaller image parts like the squares on a side of the cube. If these squares appear in an image CNN can classify it, but the relative position if squares will be ignored. Nonetheless CNN can be a possible model to recognize Pasigraphy Rhapsody image input as it is a popular and well supported neural model for image recognition. Another viable alternative is the Capsule Network [25] model. It is a rather new model with the advantage of recognizing the relative positions of objects, even if they are rotated. The disadvantage of capsules is that it is poorly supported by machine learning libraries.

Symbiosis of Theorem Provers and Machine Learning The DeepMind work [3] provides some answers to our questions, as it uses formulas formally derived from axioms as a training set. Therefore, a natural question is whether we can repeat their results using Lean. Can machine learning provide a Jaynesian inner voice that, for example, can say hints to the interactive theorem prover? Machine learning can support theorem provers and with it, Prime Convo assistant too. We can use it to select a good heuristic [9], or we can use reinforced learning for theorem proving [16], like DeepHol [4].

4.2. Translation between Lean corpus and Pasigraphy Rhapsody

One of Prime Convo Assistant's main feature is that it should translate the Lean corpus to Pasigraphy Rhapsody. The translation can rely on the numeration system of Pasigraphy Rhapsody. This numeration system is about converting elements of first-order logic to natural numbers. The conversion is the following: 1. Logical connectives and quantifiers: $\exists = 1$, $\forall = 2$, $\neg = 3$, $\wedge = 4$, $\vee = 5$, $\supset = 6$. 2. The odd numbers that are larger than 6 representing sentences that are already formalised (7, 9, 11, 13, 15, 17, ...). 3. The numbers that are divisible only by the first power of 2 represent the predicate names (10, 14, 18, 22, 26, ...). 4. The numbers that are divisible only by the second power of 2 represent the function names (12, 20, 28, 36, 44, ...). 5. The numbers that are divisible only by the third power of 2 represent the type names (8, 24, 40, 56, 72, ...). 6. The numbers that are divisible only by the fourth power of 2 represent the constants of the first type (16, 48, 80, 112, 144, ...). 7. The numbers that are divisible only by the fifth power of 2 represent the variables of the first type (32, 96, 160, 224, 288, ...). 8. The numbers that are divisible only by the 2n + 2-th power of 2 represent the constants of the n-th type (2^{2n+2}, $2^{2n+2} + 2^{2n+3}$, $2^{2n+2} + 2\times\ 2^{2n+3}$, $2^{2n+2} + 3\times\ 2^{2n+3}$, ...). 9. The numbers that are divisible only by the 2n + 3-th power of 2 represent the variables of the n-th type (2^2n+3, 2^2n+3 + 2^2n+4, 2^2n+3 + $2\times$ 2^2n+4, 2^2n+3 + $3\times$ 2^2n+4, ...). These numerical codes can be converted into coordinates with the SMNIST, and they can be visualized on a Pasigraphy Rhapsody cube.

4.3. Creation of New Sentences

The Prime Convo Assistant is a constantly expanding system, because new sentences can be added anytime. These sentences can come from Lean corpus or Pasigraphy Rhapsody images. In the previous section we introduced the translation from Lean corpus to PaRa. From images we can recreate the first-order logic formula if we use our own PaRa language. There can be countably infinite number of PaRa languages because each user will have different predicates, functions, types, constants and variables. The challenge is to interpret other PaRa languages, and beside traditional methods, this is where machine learning can be useful. With the available translation, every language can be expanded with other PaRa language, what will result rather large languages.

5. Results

This paper focuses on the Prime Convo Assistant's conceptional functionalities and position them in the introduction of Pasigraphy Rhapsody and the game based on this artificial graphical language. As Pasigraphy Rhapsody is a first-order logic based language, and every user will have a different version of it, the main question was the translation between them. For that, we did literature research about neural models that would fit for this task. For first-order logic input, deep reasoning networks seem the most appropriate. Because of the structure of image inputs, convolutional neural network or capsule network model would be the best. Extension is important in PaRa, because of it, these languages can merge with others. The extension could be made with Lean corpus or Pasigraphy Rhapsody images.

6. Conclusions and Future

As Prime Convo Assistant is in a very early, conceptional phase, we can only talk about the role of it in the Pasigraphy Rhapsody based game, and the functionalities needed to support this game. Changing traditional first-order logic processing to a neural model based was required, because of the possible size of the different Pasigraphy Rhapsody languages. Even though we have the conceptional design for Prime Convo Assistant, and functionalities like creating new sentences, translation between PaRa languages, translation between Lean corpus and PaRa, it is a long way before Prime Convo Assistant could be implemented and further to be used in a builder game.

References

[1] S. Abiteboul, S. Grumbach: *COL: A logic-based language for complex objects*, in: International Conference on Extending Database Technology, Springer, 1988, pp. 271–293, doi: https://doi.org/10.1007/3-540-19074-0_58.

[2] J. Avigad, A. Baanen, R. Barton, M. Carneiro, B. G.-g. Chen, J. Commelin, F. van Doorn, G. Ebner, S. Gouëzel, S. Hudon, C. Hughes, Y. G. Kudryashov, R. Y. Lewis, H. Macbeth, P. Massot, S. Morrison: *Lean mathlib*, `https://github.com/leanprover-community/mathlib`, 2020.

[3] E. Aygün, Z. Ahmed, A. Anand, V. Firoiu, X. Glorot, L. Orseau, D. Precup, S. Mourad: *Learning to Prove from Synthetic Theorems*, 2020, arXiv: `2006.11259 [cs.LO]`, URL: `https://arxiv.org/abs/2006.11259`.

[4] K. Bansal, S. Loos, M. Rabe, C. Szegedy, S. Wilcox: *HOList: An Environment for Machine Learning of Higher Order Logic Theorem Proving*, in: Proceedings of the 36th International Conference on Machine Learning, ed. by K. Chaudhuri, R. Salakhutdinov, vol. 97, Proceedings of Machine Learning Research, Long Beach, California, USA: PMLR, June 2019, pp. 454–463,
URL: `http://proceedings.mlr.press/v97/bansal19a.html`.

[5] N. Bátfai: *Hacking with God: a Common Programming Language of Robopsychology and Robophilosophy*, 2020, arXiv: `2009.09068 [cs.CY]`,
URL: `https://arxiv.org/abs/2009.09068`.

[6] N. Bátfai: *Pasigraphy Rhapsody*, `https://gitlab.com/nbatfai/pasigraphy-rhapsody`, 2019.

[7] N. Bátfai, D. Papp, G. Bogacsovics, M. Szabó, V. S. Simkó, M. Bersenszki, G. Szabo, L. Kovács, F. Kovács, E. S. Varga: *Object file system software experiments about the notion of number in humans and machines*, Cognition, Brain, Behavior. An Interdisciplinary Journal 23.4 (2019), pp. 257–280,
DOI: `https://doi.org/10.24193/cbb.2019.23.15`.

[8] T. Bocklisch, J. Faulkner, N. Pawlowski, A. Nichol: *Rasa: Open source language understanding and dialogue management*, arXiv preprint arXiv:1712.05181 (2017).

[9] J. P. Bridge, S. B. Holden, L. C. Paulson: *Machine learning for first-order theorem proving*, Journal of automated reasoning 53.2 (2014), pp. 141–172,
DOI: `https://doi.org/10.1007/s10817-014-9301-5`.

[10] D. Chen, Y. Bai, W. Zhao, S. Ament, J. Gregoire, C. P. Gomes: *Deep Reasoning Networks: Thinking Fast and Slow*, ArXiv abs/1906.00855 (2019).

[11] P. Clark, O. Tafjord, K. Richardson: *Transformers as Soft Reasoners over Language*, 2020, arXiv: `2002.05867 [cs.CL]`,
URL: `https://arxiv.org/abs/2002.05867`.

[12] T. mathlib Community: *The lean mathematical library*, Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (Jan. 2020),
DOI: `http://dx.doi.org/10.1145/3372885.3373824`.

[13] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, CoRR abs/1810.04805 (2018), arXiv: `1810.04805 [cs.CL]`,
URL: `http://arxiv.org/abs/1810.04805`.

[14] J. A. Feldman, R. F. Sproull: *Decision theory and artificial intelligence II: The hungry monkey*, Cognitive Science 1.2 (1977), pp. 158–192,
DOI: `https://doi.org/10.1207/s15516709cog0102_2`.

[15] Y. Harayama: *Society 5.0: Aiming for a New Human-centered Society*, Hitachi Review 66.6 (2017), pp. 8–13,
URL: `https://www.hitachi.com/rev/archive/2017/r2017_06/pdf/p08-13_TRENDS.pdf`.

[16] C. Kaliszyk, J. Urban, H. Michalewski, M. Olšák: *Reinforcement learning of theorem proving*, in: Advances in Neural Information Processing Systems, 2018, pp. 8822–8833.

[17] S. Kottwitz: *Example: Sudoku 3D cube*, 2008,
URL: `http://www.texample.net/tikz/examples/sudoku-3d-cube`.

[18] S. Kramer, G. Widmer: *Inducing classification and regression trees in first order logic*, in: Relational Data Mining, Springer, 2001, pp. 140–159,
doi: https://doi.org/10.1007/978-3-662-04599-2_6.

[19] Y. LeCun, Y. Bengio: *Convolutional networks for images, speech, and time series*, in: The handbook of brain theory and neural networks, ed. by M. A. Arbib, 2nd ed., 2003, pp. 276–279.

[20] X. Liu, P. He, W. Chen, J. Gao: *Multi-Task Deep Neural Networks for Natural Language Understanding*, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy: Association for Computational Linguistics, July 2019, pp. 4487–4496,
doi: https://doi.org/10.18653/v1/P19-1441,
url: https://www.aclweb.org/anthology/P19-1441.

[21] A. Lovrenčić: *Knowledge Base Amalgamation Using the Higher-Order Logic-Based Language HiLog*, Journal of Information and Organizational Sciences 23.2 (1999), pp. 133–147.

[22] S. Manchanda, D. S. Warren: *A logic-based language for database updates*, in: Foundations of deductive databases and logic programming, Elsevier, 1988, pp. 363–394,
doi: https://doi.org/10.1016/B978-0-934613-40-8.50014-2.

[23] J. Ramon: *Clustering and instance based learning in first order logic*, AI Communications 15.4 (2002), pp. 217–218.

[24] S. Rekhis, N. Boudriga: *A formal logic-based language and an automated verification tool for computer forensic investigation*, in: Proceedings of the 2005 ACM symposium on Applied computing, 2005, pp. 287–291,
doi: https://doi.org/10.1145/1066677.1066745.

[25] S. Sabour, N. Frosst, G. E. Hinton: *Dynamic routing between capsules*, in: Advances in neural information processing systems, 2017, pp. 3856–3866.

[26] S. Shi, H. Chen, W. Ma, J. Mao, M. Zhang, Y. Zhang: *Neural Logic Reasoning*, 2020, arXiv: 2008.09514 [cs.LG].