Logic Acquisition and Inference by Neural Networks

Bowen Xu

Department of Computer and Information Sciences,
Temple University, Philadelphia, USA
bowen.xu@temple.edu

December 12, 2023

Abstract. Can ChatGPT or GPT-4 [1] do *logic* reasoning? Many people have doubts on this question. Instead of answering it, let us consider a more general question – Can a neural network do logic reasoning? My answer implied in the tittle is of course yes. A neural network in a general sense can do logic reasoning, though it depends on the architecture or structure of a neural network. I will defend my answer by proposing and testing a neural network which can acquire logic rules through learning.

Keywords: Logic Acquisition · Logic Reasoning · Neural Network

1 Introduction

There is an intense debate on the question: *Can ChatGPT do reasoning*? The major positive arguments are

P1) You can see a bunch of reasoning tasks in which ChatGPT performs *well*.

P2) You can tell it an inference rule, and it seems able to do something like logical reasoning (see Fig. 1).

A possible objection to **P2** is

N0) The instance you gave to ChatGPT involves some common words, including *robin*, *bird*, and *animal*. These special cases might be encountered in its training stage, and ChatGPT just somehow memorizes it, and it does not really do reasoning.

A response to **N0** is

P0) OK, let's use some randomly generated words, such as "qwelkke", "xadfasdf", and "lkjwerk". They are most probably not encountered in the training set. ChatGPT still works well (see Fig. 2). This provides a positive evidence that ChatGPT is doing something which is irrelevant to concrete contents but is relevant to the transformation of the abstract form.

Nevertheless, **P0** is not convincing enough. The major negative arguments are

N1) You can see a bunch of reasoning tasks in which ChatGPT performs *badly*.

N2) I agree that ChatGPT can do reasoning in some sense. What it can do is merely *statistical reasoning*, meaning that the conclusion is influenced by the specific contents of the premises. However, what it cannot do is *logical reasoning*, in which it performs purely formal transformations that are independent of specific contents. Though **P0** provides an evidence that it seems to do formal transformation in that case, but there are still a bunch of cases which provides negative evidences. *There is no such a principle or mechanism which makes ChatGPT do logical reasoning.*

I believe debating on **P1** and **N1** is boring, and the valuable part is on **P2** and **N2**. Consequently, the meaningful question is *whether it is possible for a neural network to learn something called logical reasoning*. In this project, I am trying to prove that a certain neural network can do logical reasoning, by looking for a mechanism based on which a neural network indeed does logical reasoning.

A by-product will be the (partial) answer to a more interesting question: *How does logic emerge from biological/artificial neural networks*? I believe there are two possible answers:

(a) There is something innate which is called a logic, which justifies that "$\{A, B\} \vdash Z$".
(b) There is nothing innate as a logic, but a logic is an acquired skill. "$\{A, B\} \vdash Z$" is something learned by a neural network.

The key difference between these two answers is the representations to be adopted. The former one adopts a logical representation, while the latter one adopts a neural representation. Although the result of this project would probably support answer (b), this does not mean that answer (a) is not reasonable.

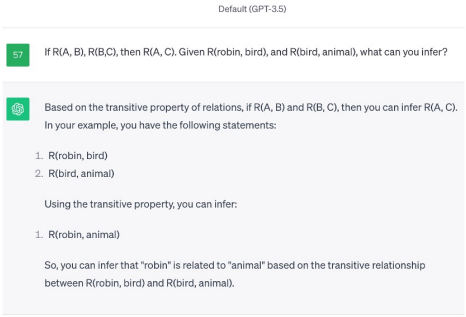


Fig. 1. ChatGPT learns the rule "$\{R(A, B), R(B, C)\} \vdash R(A, C)$".

The goal of this paper is trying to answer a theoretical question: *can neural networks do logical reasoning*?

To answer this question, I firstly considered the general from of logical rules; then I tried to generate a number of input-output instances for each of the rules; finally, neural networks learned from the instances, extracting abstract formal

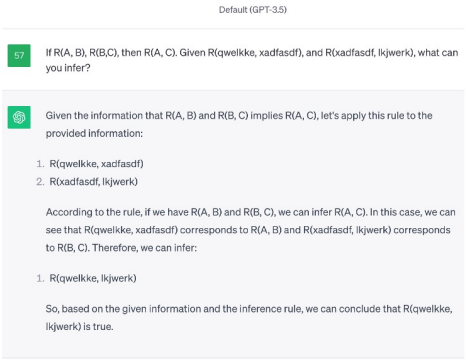


Fig. 2. A case to support that ChatGPT learns the rule "$\{R(A,B), R(B,C)\} \vdash R(A,C)$".

relations (*i.e.*, logical rule). The key point here is that how to examine the neural networks really know the relations, rather than *pretend* to know the relations.

To solve this issue, firstly, I *assume* each of elements in an embedding is in $[-1, 1]$. This is a justifiable assumption. Secondly, I propose an *hypothesis* (H1) that

> *if* the training data is highly biased (C1), while the test data is uniformly distributed in the whole space (C2), and a neural network can reach 100% accuracy on both training set and test set (C3), *then* the neural network successfully obtains the formal relations (D1).

Thirdly, a neural network is trained on a highly biased training set and tested on a uniformly distributed test set.

The preliminary results show that a Transformer [5] network gets 100% accuracy on both the data sets, implying that Transformer can learn formal relations (rather than merely statistical correlation). An extended conclusion is that neural networks have potential to learn formal relations.

2 Related Works

Many researchers tried to enable neural network to do reasoning. The field mainly on this issue is Neural Symbolic Reasoning [2], where most of the key ideas had been proposed more than 15 years ago. They interpreted some neurons or layers to be logical component (*e.g.*, AND gate) and trained networks given some input-output samples. The intuition in this project is the same, but the motivations are different. In the previous work, they aim to simulate logic in neural network. However, this project aims to address a theoretical issue, *i.e.*, whether neural networks can really do logical reasoning. Besides, the structures to be chosen is different from the previous ones. Some of them made some special designs or constraints on the neural network, as a result, some of their neural networks

are quite different from the commonly used neural networks, which sum up all inputs then pass the summation to an activation function and update weights by back-propagation and gradient descent algorithms. The structure adopted in this project is either the mainstream network (see Fig. 8) or a special network which is novel in the high-level structure (see Fig. 9), while keeping the fundamental mechanisms of deep neural networks unchanged. As a result, this project is more suitable for answering the questions raised above, since ChatGPT might do the similar things as the networks used in this project. There are also much recent work on neural symbolic reasoning, such as IBM's Logic NN [4], RNNLogic [3] and so on.

3 Methodology

The goal of this project is to prove that a neural network can do logical reasoning. To achieve that, the overall strategy is as the following: first, a logic rule is a formal transformation from premises to a conclusion; second, if a neural network is doing a formal transformation, it is doing logical reasoning; third, let a neural network learned the transformation, and prove that the transformation is independent of the concrete contents of its input. I aware that such a neural network, strictly speaking, can do logical inference, meaning a single step of a reasoning process, however, it is sufficient to prove the potential of the neural network to do (multi-steps) logical reasoning, by adding some addition parts.

3.1 General Form of Logic Rules

More concretely, the general form of a logic rule is as follows. Given some premises,

$$\begin{aligned} &Relation_1(Term_{1,1}, \ldots, Term_{1,n}) \quad \langle tv_1 \rangle \\ &Relation_2(Term_{2,1}, \ldots, Term_{2,n}) \quad \langle tv_2 \rangle \\ &\ldots \\ &Relation_m(Term_{m,1}, \ldots, Term_{m,n}) \quad \langle tv_m \rangle \end{aligned} \tag{1}$$

through a single step inference with a logic rule, a conclusion is made

$$Relation_{m+1}(Term_{m+1,1}, \ldots, Term_{m+1,n}) \langle tv_{m+1} \rangle \tag{2}$$

A special case in Non-Axiomatic Logic (Wang) is the syllogistic deduction rule , where the premises are

$$\begin{aligned} &isa(M, P) \langle f_1; c_1 \rangle \\ &isa(S, M) \langle f_2; c_2 \rangle \end{aligned} \tag{3}$$

and the conclusion is

$$isa(S, P) \langle f_1 f_2; f_1 f_2 c_1 c_2 \rangle \tag{4}$$

The hypothesis in this project is that it is possible for a neural network to learn the rule. There should be an operator to judge whether the two relations

of the premises are equal (both are "isa"), as well as whether the first term of the first premises is equal to the second term of the second premises (both are "M"). Then a conjunction operator is used to determined whether two premises can match the rule. If so, a copy operator transits the corresponding parts to the output. There can be a mapping to compute the truth-value. The schematic diagram is shown in Fig. 3.

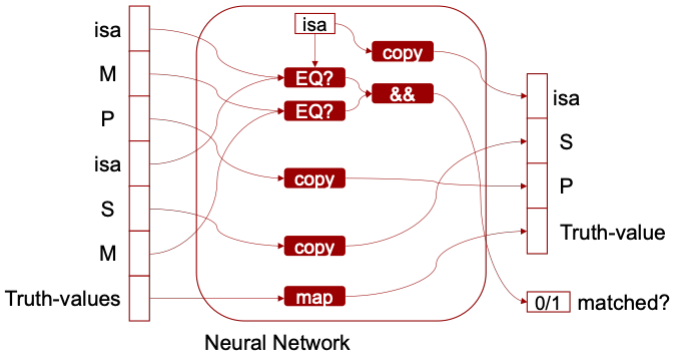


Fig. 3. The formal transformation from premises to a conclusion

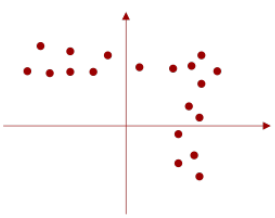
3.2 Dataset

The tricky part is how to test whether a rule is irrelevant to special contents of input or not. We have a hypothesis that if the neural network can generalize to the test data whose distribution is different from that of the training data, then it learns the formal transformation.

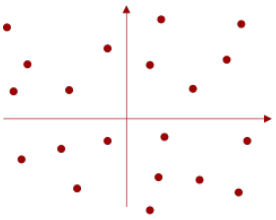
I suggest to use some synthetic dataset, so that the distributions can be well controlled. Let us assume each element of an embedding (i.e., an input content of the neural network) is in [-1, 1]. The data in training set should be highly biased (see Fig. 4(a)), while the data in test set should be uniformly in the whole space (see Fig. 4(b)). If a neural network can generalize to this test dataset, then it learns the abstract transformation, i.e., the logic rule.

3.3 Logic Rules

I consider a list of logic rules in Non-Axiomatic Logic [6]:



(a) Distribution of training data (highly biased)



(b) Distribution of test data (uniformly distributed in the whole space)

Fig. 4. Data distribution

Rules in NAL-1:

$$
\begin{aligned}
&\{\langle M \to P\rangle, \langle S \to M\rangle\} \vdash \langle S \to P\rangle (F_{ded}) \\
&\{\langle P \to M\rangle, \langle M \to S\rangle\} \vdash \langle P \to S\rangle (F'_{ded}) \\
&\{\langle M \to P\rangle, \langle M \to S\rangle\} \vdash \langle S \to P\rangle (F_{ind}) \\
&\{\langle M \to P\rangle, \langle M \to S\rangle\} \vdash \langle P \to S\rangle (F'_{ind}) \\
&\{\langle P \to M\rangle, \langle S \to M\rangle\} \vdash \langle S \to P\rangle (F_{abd}) \\
&\{\langle P \to M\rangle, \langle S \to M\rangle\} \vdash \langle P \to S\rangle (F'_{abd}) \\
&\{\langle P \to M\rangle, \langle M \to S\rangle\} \vdash \langle S \to P\rangle (F_{exe}) \\
&\{\langle M \to P\rangle, \langle S \to M\rangle\} \vdash \langle P \to S\rangle (F'_{exe})
\end{aligned}
\tag{5}
$$

Rules in NAL-2:

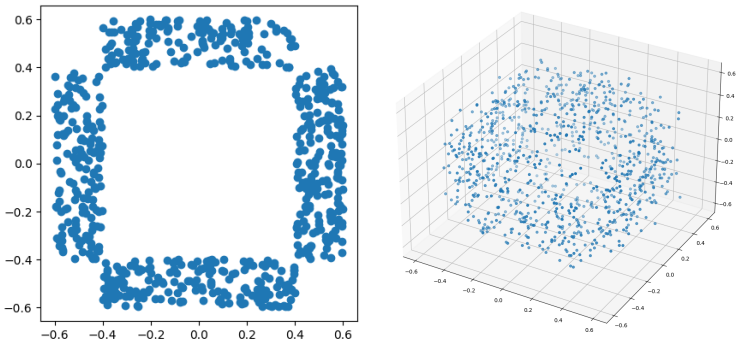
$$
\begin{aligned}
&\{\langle M \leftrightarrow P\rangle, \langle S \leftrightarrow M\rangle\} \vdash \langle S \leftrightarrow P\rangle (F_{res}) \\
&\{\langle M \leftrightarrow P\rangle, \langle M \leftrightarrow S\rangle\} \vdash \langle S \leftrightarrow P\rangle (F_{res}) \\
&\{\langle P \leftrightarrow M\rangle, \langle S \leftrightarrow M\rangle\} \vdash \langle S \leftrightarrow P\rangle (F_{res}) \\
&\{\langle P \leftrightarrow M\rangle, \langle M \leftrightarrow S\rangle\} \vdash \langle S \leftrightarrow P\rangle (F_{res}) \\
&\{\langle M \to P\rangle, \langle M \to S\rangle\} \vdash \langle S \leftrightarrow P\rangle (F_{com}) \\
&\{\langle P \to M\rangle, \langle S \to M\rangle\} \vdash \langle S \leftrightarrow P\rangle (F'_{com}) \\
&\{\langle M \to P\rangle, \langle S \leftrightarrow M\rangle\} \vdash \langle S \to P\rangle (F_{ana}) \\
&\{\langle M \to P\rangle, \langle M \leftrightarrow S\rangle\} \vdash \langle S \to P\rangle (F_{ana}) \\
&\{\langle P \to M\rangle, \langle S \leftrightarrow M\rangle\} \vdash \langle P \to S\rangle (F_{ana}) \\
&\{\langle P \to M\rangle, \langle M \leftrightarrow S\rangle\} \vdash \langle P \to S\rangle (F_{ana}) \\
&\{\langle M \leftrightarrow P\rangle, \langle S \to M\rangle\} \vdash \langle S \to P\rangle (F'_{ana}) \\
&\{\langle P \leftrightarrow M\rangle, \langle S \to M\rangle\} \vdash \langle S \to P\rangle (F'_{ana}) \\
&\{\langle M \leftrightarrow P\rangle, \langle M \to S\rangle\} \vdash \langle P \to S\rangle (F'_{ana}) \\
&\{\langle P \leftrightarrow M\rangle, \langle M \to S\rangle\} \vdash \langle P \to S\rangle (F'_{ana})
\end{aligned}
\tag{6}
$$

where the symbol "→" intuitively means "is a", and "↔" intuitively means "is similar to".

3.4 Data Generation

For each of the rules, some corresponding instance are generated. For example, with the deduction rule $\{\langle M \to P\rangle, \langle S \to M\rangle\} \vdash \langle S \to P\rangle$ [1], an instance could be $\{\langle bird \to animal\rangle, \langle robin \to bird\rangle\} \vdash \langle robin \to animal\rangle$, meaning that if "bird is an animal" and "robin is a bird" are true, then "robin is an animal" is true.

To generate more instances, I define a dictionary, in which there are around $20,000$ words, marked by 'obj(1)', 'obj(2)', ..., 'obj(20000)'. Word 'obj(i)' where $i \in \{1, 2, ..., 10000\}$ are only used in training set, and 'obj(j)' where $j \in \{10000, 10001, ..., 20000\}$ only in test set. The word embeddings in the training set are distributed a "cube shell". For example, if the dimension of an embedding is 2, then the data points are shown in Fig. 5. However, in the test set, data points corresponding to their word embeddings are uniformly distributed.



(a) Word embeddings of training set in 2-D space

(b) Word embeddings of training set in 3-D space

Fig. 5. The distribution of training data is highly biased.

Given the dictionary as well as a single rule, some words are randomly picked up to instantiate the rule. For example, one instance looks like

$$\begin{aligned}(((\text{`}\to\text{'}, \text{`}obj4827\text{'}, \text{`}obj1255\text{'}), (\text{`}\to\text{'}, \text{`}obj1732\text{'}, \text{`}obj4827\text{'})),\\ (\text{`}\to\text{'}, \text{`}obj1732\text{'}, \text{`}obj1255\text{'}))\end{aligned} \tag{7}$$

where the first line contains the input words, and the second line contains the output words.

This instance is then converted to vectors, as shown in

[1] Here, the truth-values are omitted.

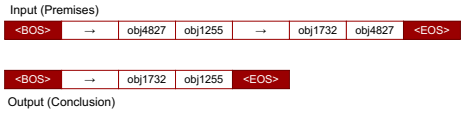


Fig. 6. Vector representation of the instance in Eq. 7

Taking all the input words as a whole embedding, it is checked whether the distributions in the training set and the test set are significantly different (see Fig. 7). We can see that the experimental setting satisfies the conditions (C1) and (C2) in the hypothesis (H1).

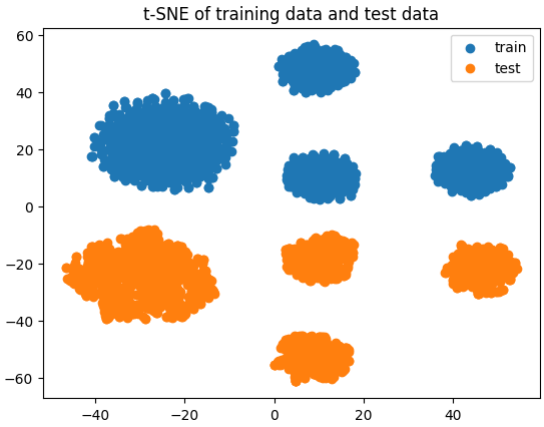


Fig. 7. Data distributions of training set and test set

3.5 Neural Networks

I use a Transformer network (see Fig. 8) to induce the logic rules from the instances.

Another structure of neural network is attempted in this paper (see Fig. 9). There is a list of sub-networks (*i.e.*,*RuleNet*s), each of which represents a logic rule. There is a selection network which produce masks for selecting rules. There is also a scalar (*indicator*) to indicate whether any rules are applicable. All the *RuleNet*s and the *Selection Network* are Fully Connected neural networks, each of which composed of 3 layers.

The full code is attached in the supplementary materials.

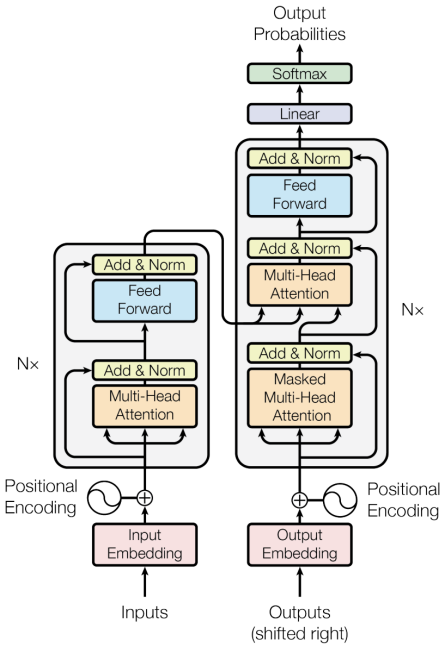


Fig. 8. The Transformer – model structure [5]

4 Results

The Transformer model is trained for 4 epochs, and the loss curve is shown in Fig. 10. The accuracy on the training set and the test set are both 100%. Consequently, the condition (C3) is satisfied. The LogicANN model is trained for 10 epochs, and the loss curve is shown in Fig. 11. The training accuracy is 97.1%, and the test accuracy is 72.7%.

	Transformer	**LogicANN**
Training Acc	100.0%	97.1%
Test Acc	100.0%	72.7%

Table 1. Accuracy

Without doubt, to display some internal states of the neural networks and explain how the formal transformation happens inside the network, it is valuable and may give us some deeper understandings. Why the merely Fully-Connected neural networks does not successfully learn the formal relation but the Transformer can learn it deserves further experiments and research. However, considering the aim of this paper, I do not continue the research since it is sufficient

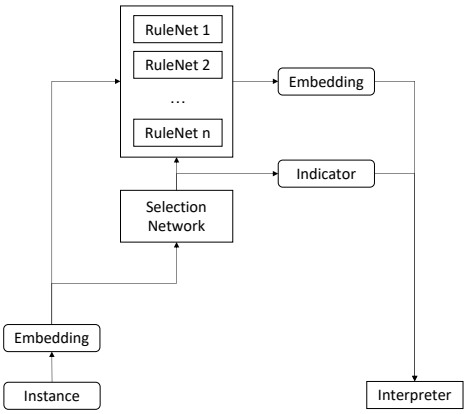


Fig. 9. The structure of *LogicANN*

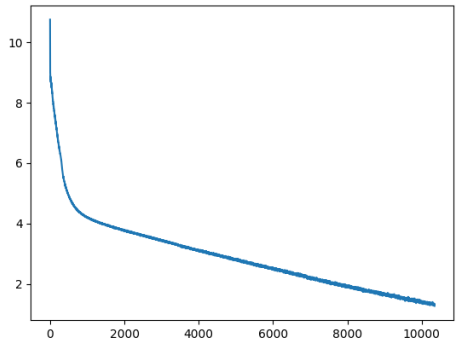


Fig. 10. Loss curve of Transformer in the training set

to get the answer we want. I have other issues to research so that I have to stop this work that is not so intriguing for me.

5 Conclusion

From the dataset setting, we know that the training set is highly biased, and the test set is uniformly distributed. From Tab. 1 we can see that the Transformer network obtains 100% accuracy in both training and test dataset. Therefore, according to the hypothesis (H1), we can draw a conclusion that *the Transformer network successfully obtains the formal relations*.

Since the formal relations in this paper are the logical rules, the Transformer network actually does logical inference in this experiment. Hence, the experimental result support that *a neural network, especially a Transformer network, can do logical reasoning*. By contrast, the *LogicANN* network does not satisfy the third condition (C3), thus it is not really able to do logical reasoning.

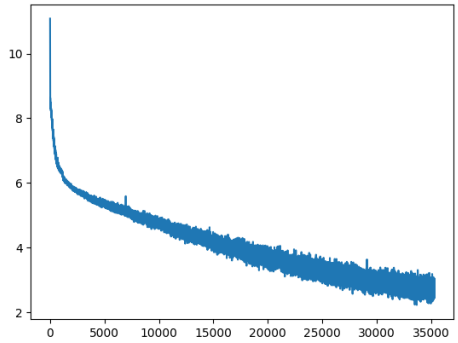


Fig. 11. Loss curve of LogicANN in the training set

Nevertheless, there are still some fundamental limitations for a neural network to *learn* or *acquire* a logic. The Transformer network summarizes its experience from instances, as a result, it learns some formal mappings between its input and output. The input is interpreted as the premises, while the output is interpreted as conclusions of logic rules. However, human beings *learn* or *acquire* a logic in a quite different way. Humans propose some logic rules from hypotheses, and then design semantics and give solid justifications for those hypotheses. In the current paradigm of deep learning, as well as neural symbolic reasoning, we do not see a possibility for a neural network to *think* and *acquire* a logic in this way. This implies a potential research direction for neural symbolic reasoning, and we might as well call it *neural logic acquisition*, which aims to model the procedure of humans to learn a logic.

References

1. Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y.T., Li, Y., Lundberg, S., et al.: Sparks of artificial general intelligence: Early experiments with gpt-4. arXiv preprint arXiv:2303.12712 (2023)
2. Garcez, A.S.D., Lamb, L.C., Gabbay, D.M.: Neural-Symbolic Cognitive Reasoning. Springer Science & Business Media (Oct 2008)
3. Qu, M., Chen, J., Xhonneux, L.P., Bengio, Y., Tang, J.: RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs (Oct 2020)
4. Riegel, R., Gray, A., Luus, F., Khan, N., Makondo, N., Akhalwaya, I.Y., Qian, H., Fagin, R., Barahona, F., Sharma, U., Ikbal, S., Karanam, H., Neelam, S., Likhyani, A., Srivastava, S.: Logical Neural Networks (Jun 2020), http://arxiv.org/abs/2006.13155
5. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, , Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017)
6. Wang, P.: Non-axiomatic logic: A model of intelligent reasoning. World Scientific (2013)