A Non-Turing Computer Architecture for Artificial Intelligence with Rule Learning and Generalization Abilities Using Images or Texts

Jineng Ren *

Posted Date: 16 January 2025

doi: 10.20944/preprints202412.2432.v2

Keywords: non-turing computer architecture; images/texts; abstract domain; specific domain; artificial intelligence; non-Turing robotics

Article

A Non-Turing Computer Architecture for Artificial Intelligence with Rule Learning and Generalization Abilities Using Images or Texts

Jineng Ren

Wenzhou University, AIAMI, China; renjineng@mail.nankai.edu.cn

Abstract: Since the beginning of modern computer history, Turing machine has been a dominant architecture for most computational devices, which consists of three essential components: an infinite tape for input, a read/write head, and finite control. In this structure, what the head can read (i.e. bits) is the same as what it has written/outputted. This is actually different from the ways in which humans think or do thought/tool experiments. More precisely, what humans imagine/write on the paper are images or texts, and they are not the abstract concepts that they represent in humans' brain. This difference is neglected by the Turing machine, but it actually plays an important role in abstraction, analogy, and generalization, which are crucial in artificial intelligence. Compared with this architecture, the proposed architecture uses two different types of heads and tapes, one for traditional abstract bit inputs/outputs and the other for specific visual ones (more like a screen or a workspace with a camera observing it). The mapping rules between the abstract bits and the specific images/texts can be realized by neural networks like Convolutional Neural Networks, YOLO, Large Language Models, etc., with high accuracy rate. As an example, this paper presents how the new computer architecture (what we call "Ren's machine" for simplicity here) autonomously learns a distributive property/rule of multiplication in the specific domain and further uses the rule to generate a general method (mixed in both the abstract domain and the specific domain) to compute the multiplication of any positive integers based on images/texts. Moreover, a robotic architecture based on Ren's machine is proposed to address the challenges faced by the Vision-Language-Action (VLA) models in unsound reasoning ability and high computational cost.

Keywords: non-turing computer architecture; images/texts; abstract domain; specific domain; artificial intelligence; non-Turing robotics

1. Introduction and Related Work

In recent years, the research of deep neural network models has witnessed significant progress as various large language models are proposed and evolved at a fast pace. However, currently these models still cannot achieve reasonable performance in logical and mathematical reasoning without much human intervention, although more than 10 billion trainable parameters are placed in their latest neural networks. The failure of these sophisticated models in some seemly simple math problems inspired us to propose a non-Turing computer structure to address this challenge. The number of previous works on non-Turing machines is relatively small ([1–7]), which mostly discuss it conceptually and psychologically without giving any specific workable architecture like the Turing one. Among these, [2] and [7] argued that analog or natural computations should be used to conduct calculations involving continuous real numbers. The potential adaptivity of this type of models, however, is only restricted to the range between the discrete approximation of a continuous number and its unrepresented portion. Moreover, it lacks the interactions among the discrete domain and the continuous domain to solve the problem in their model as well. Recently, [8] and its variants investigated so-called Neural Turing Machines, which uses neural networks like LSTM [9] as the controller to make the entire system differentiable for end-to-end training. Compared

 doi:10.20944/preprints202412.2432.v2

with this structure, the abstract controller in the proposed non-Turing machine (which will be called "Ren's machine" for simplicity below) can still be discrete/binary-based for logic reasoning. This novel structure allows the flexibility of switching back and forth between the abstract domain and the specific domain to better approach a solution to a given problem with rules summarized or learned from thought/tool experiments during the process. What's more, it also enables the interactive collaborations of specific contents and abstract contents on both types of tapes to command and instruct the controller for completing a task.

2. The Proposed Architecture

The components and their relations of Ren's machine are elaborated in Figure 1. This structure consists of a controller and two types of read/write heads and tapes, one in the specific domain and the other in the abstract domain. The interactions inter and intra the specific domain and the abstract domain are realized by different types of mappings annotated in this figure. These mappings can be implemented by applying prior knowledge or learned rules (e.g., neural networks or memorized tables) or simply rendering using a built-in function on the screen. Note that the writing and reading processes utilizing the heads in the specific domain are affected by the environment in the real world. As a result, some information from the outside world is also involved during these processes.

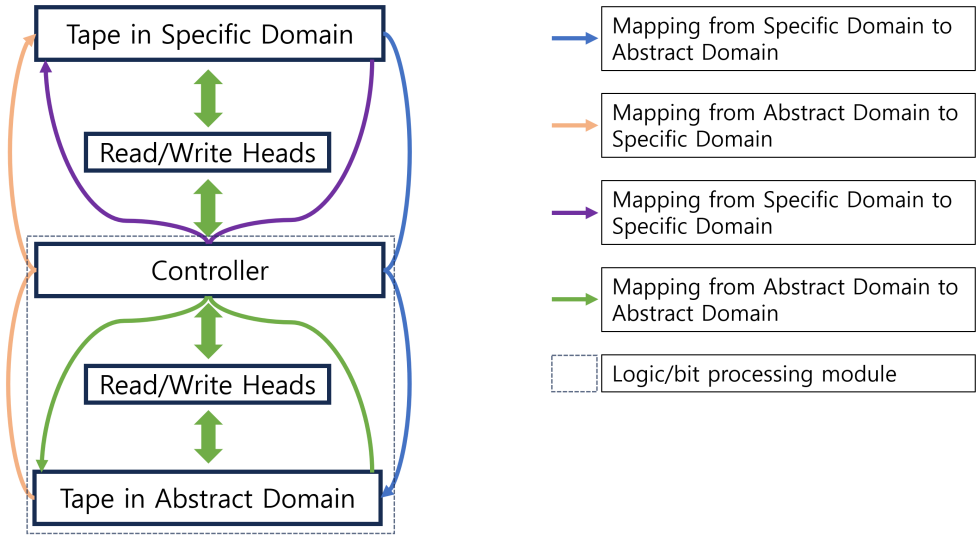

Figure 1. This shows the architecture of the Ren's machine. Here different types of mappings among and between the specific domain and the abstract domain can be realized by learned rules (e.g., neural networks or memorized tables) or simply rendering on the screen. The reading process on the tape in the specific domain is analog to taking images of the contents on a screen. The writing process is analog to displaying contents in a certain way onto the screen. The contents on the tape in the abstract domain can be binary digits (bits).

3. Experiment

In this experiment, we aim to autonomously find a method of computing the product of any positive integers by using very basic algebraic rules that are already known. The following diagram shows the workflow of Ren's machine to calculate an expression where it learned the distributive rule of multiplication by experiments generated in the specific domain and used the rule to get the correct result of the expression.

In this process, when it is recognized that a rule is technically applicable to any objects in either the abstract domain or the specific domain, the rule will be called and applied to them as a trial to move towards a solution even it seems that applying this rule to the objects is 'meaningless' (certainly a search strategy like reinforcement learning could be added for more 'efficient' look up). New rules are summarized or learned from thought or tool experiments generated from prior knowledge and existing rules that have already been learned so far. For example, in the computing process demonstrated

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 January 2025 doi:10.20944/preprints202412.2432.v2

in Figure 2, the distributive rule is learned by the cooperation of specific observations and abstract prior knowledge. Specifically, Ren's machine can compute the multiplication of any small integers like 12×30, 10×30, 2×30 using multiple additions by the original definition of multiplication. Then it finds that actually the equation $\boxed{12 \times 30 = 10 \times 30 + 2 \times 30}$ holds true. It then generates an example in the specific domain accordingly by simply displaying the equation on the screen. Using similar procedure, it can generate a large number of examples (images) in the specific domain showing that the distributive property exists. Then neural networks can be trained to learn the transformation between the images on the left side of the equation and those on the right side of the equation. It is shown in the experiment that such neural networks (e.g. LLaMA [10], ChatGPT [11]) are able to generalize to other pairs of integers that are unseen in the training, even when the integers are very large. After applying the transformation in the visual domain, we can use computer vision models like YOLO [12] to convert the equation back into the abstract domain. This procedure eventually enables Ren's machine to autonomously learn a general method of calculating the multiplications of arbitrary positive integers.

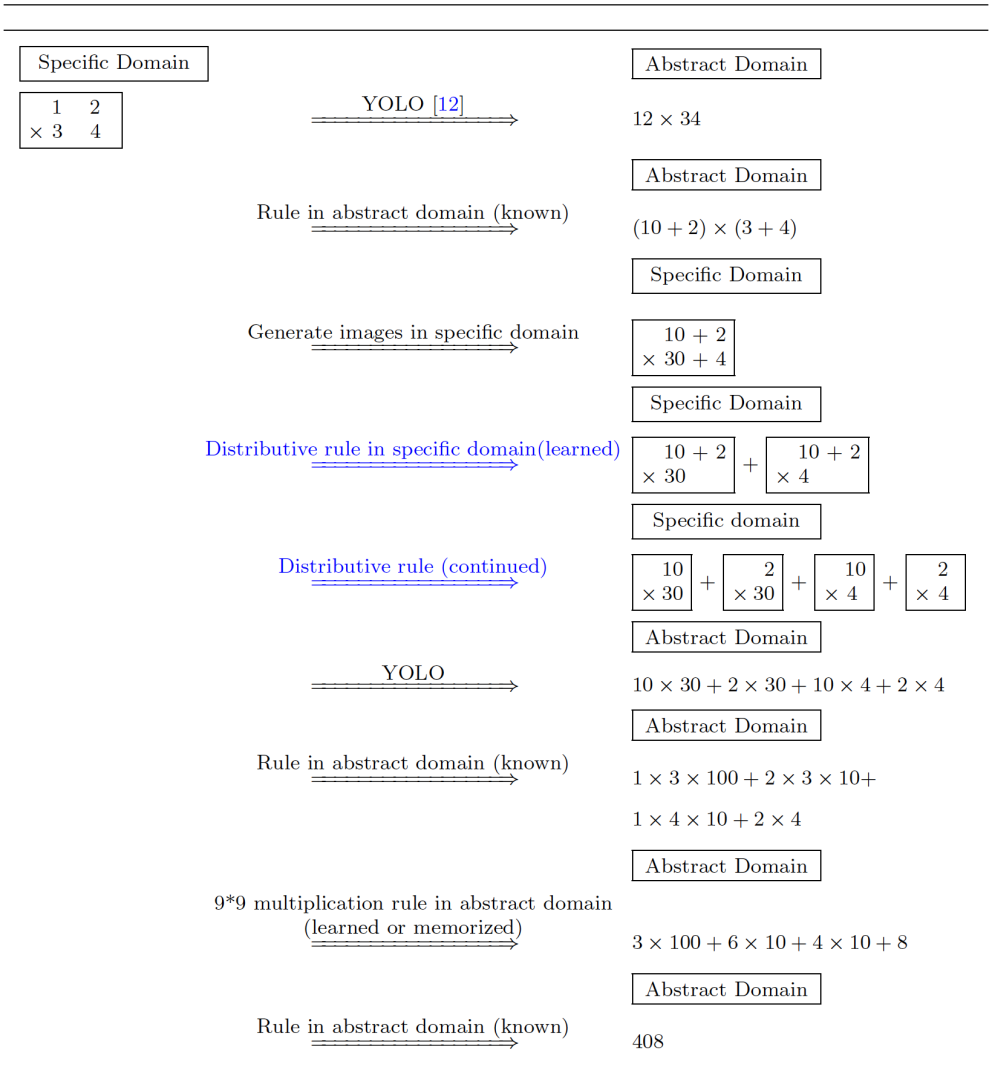

Figure 2. The Workflow of Computing The Multiplication of Positive Integers in The Ren's Machine.

We conduct the test of computing the multiplications in 20 pairs of positive integers listed as follows:

 doi:10.20944/preprints202412.2432.v2

12*34,	123*345,	1234*3456,	12345*34567,	123456*345678
23*45,	234*456,	2345*4567,	23456*45678,	234567*456789
34*56,	345*567,	3456*5678,	34567*56789,	345678*567891
45*67,	456*678,	4567*6789,	45678*67891,	456789*678912

The results in this experiment show that the accuracy (100%) achieved by the proposed method on Ren's machine is significantly higher than that (40%) achieved by the state-of-the-art large language model used in Microsoft Copilot [13].

4. Conclusion and Discussion

In this paper, we propose a non-Turing architecture, Ren's machine, that solves a problem by autonomously learning rules and building knowledge bases from generated experiments in the process of approaching an answer. It can input and output by using two types of read/write heads in both the specific domain and the abstract domain. The mappings among these domains can be conducted based on learned rules such as neural networks, memorized tables, or simply built-in rendering functions. In this way the controller can be commanded/instructed cooperatively by both the contents on the tape in the abstract domain and those in the specific domain. In the experiments, it is demonstrated that this flexible mechanism enables Ren's machine to solve the multiplication problem of positive integers in a self-directed manner with higher accuracy rate, compared with other state-of-the-art LLM methods. More advantages of Ren's machine could be further explored by coping with the problems that cannot be solved by Turing machine intrinsically. As we know, the halting problem is the problem of determining whether an arbitrary program will terminate, or continue to run forever. This problem is proven to be undecidable on Turing machine, meaning that there exists no general algorithm on Turing machine that can solve the halting problem for all possible pairs of program and input. On the other hand, on Ren's machine, since the controller is connected with the specific domain using another type of read/write heads, it can also be commanded/instructed by the contents shown on the tape in the specific domain. Thus the halting problem can be addressed with significantly high accuracy, if one includes a clock in the specific domain to count the time that a program has been running for, or uses neural networks to recognize the text patterns of never-ending programs, or counts the number of steps from their printouts on the screen, etc. In this regard, instead of pretending to be human in conversation, if a machine can actively generate rules from thought/tool experiments, for generalizing them to approach an answer of the problem it tries to solve, then it can be closer to the real artificial general intelligence.

Appendix A. Non-Turing Robotics

Although recent Vision-Language-Action (VLA) models manage to enable robots to complete a variety of basic tasks, they still suffer from drawbacks in two aspects: (1) lack of reasoning ability to handle complicated tasks, and (2) environment setups, datasets, and computation intensive for training and fine-tuning VLA model. To address these challenges, we propose a non-Turing robotic architecture based on Ren's machine (see Figure A1), which can conduct spatial-temporal reasoning through the interactions between specific domain and abstract domain.

On the proposed non-Turing robotic architecture based on Ren's machine, the reasoning can be carried out by the interactions between abstract domain and specific domain. It comprehends the current situation in the outside world by firstly converting the scenario's videos into sketches in the specific domain using vision-to-language/sketch model. If the model is trained properly, the sketches will contain necessary information to determine whether a collision is going to occur or not with high probability. Then, a mapping (e.g., a classification neural network) is used to convert the sketches into the contents in the abstract domain such as 'True' or 'False' to flag collision. In addition, a generative model can be adopted to generate the sketches of the consequential scenario assuming that a tentative movement is implemented by the robot. By using this mechanism, the proposed robotic architecture can conduct multiple virtual trials to find out the best policy to avoid potential collisions from these

 doi:10.20944/preprints202412.2432.v2

thought experiments. The action generated by the best policy will then be executed by the robot in the real world. Notice that in this process the reasoning and computations are implemented based on the simple sketches instead of the full videos of real-world scenarios. As a result, the computational cost as well as the cost of setting up environments and generating data on the proposed robotic architecture will be markedly reduced.

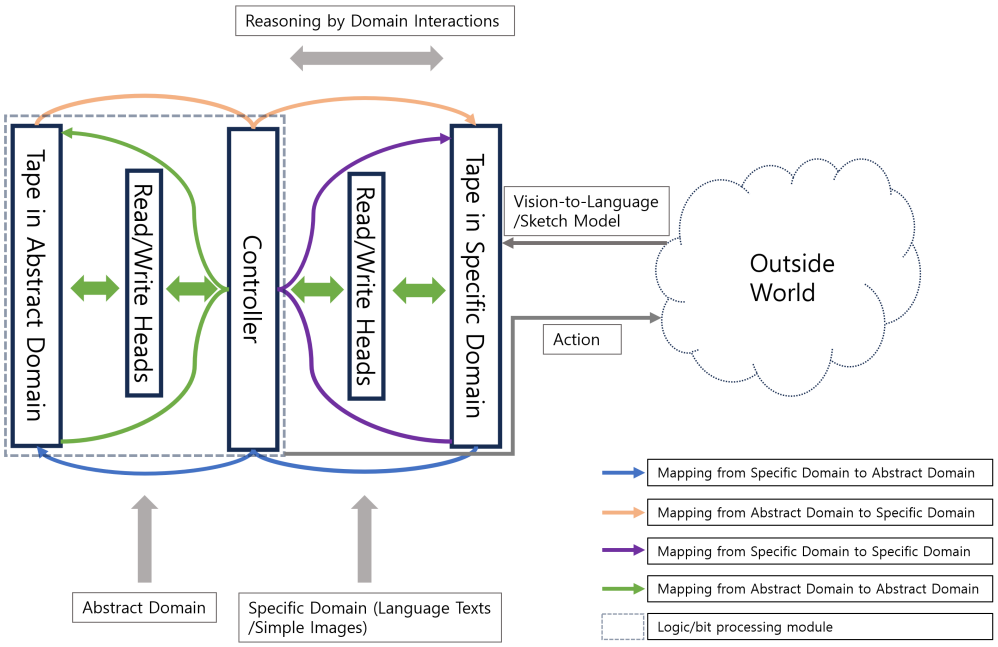

Figure A1. This shows the non-Turing robotic architecture based on Ren's machine. The spatial-temporal reasoning is implemented based on the contents in the specific domain (such as a simple sketch of objects in the outside world) generated from applying vision-to-language/sketch model to videos of real-world scenarios. For example, when the sketch in the specific domain shows the collision is about to happen (i.e., the mapping result of the sketch into the abstract domain is 'True' for collision), the controller will generate another movement direction and generate a sketch of the scenario assuming that the robot has moved in this direction. By doing multiple low-cost thought experiments, it will choose the best strategy learned from these experiments and implement it in the real world.

References

1. Holyoak, K.J. Why I am not a Turing machine. *Journal of Cognitive Psychology* **2024**, pp. 1–12.
2. MacLennan, B.J. Natural computation and non-Turing models of computation. *Theoretical computer science* **2004**, *317*, 115–145.
3. Harel, D.; Marron, A. The Human-or-Machine Issue: Turing-Inspired Reflections on an Everyday Matter. *Communications of the ACM* **2024**, *67*, 62–69.
4. Brynjolfsson, E. The turing trap: The promise & peril of human-like artificial intelligence. In *Augmented education in the global age*; 2023; pp. 103–116.
5. Mitchell, M. The Turing Test and our shifting conceptions of intelligence, 2024.
6. Hoffmann, C.H. Is AI intelligent? An assessment of artificial intelligence, 70 years after Turing. *Technology in Society* **2022**, *68*, 101893.
7. MacLennan, B.J. Mapping the territory of computation including embodied computation. In *Handbook of Unconventional Computing: VOLUME 1: Theory*; 2022; pp. 1–30.
8. Graves, A. Neural Turing Machines. *arXiv preprint arXiv:1410.5401* **2014**.
9. Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. *Neural Computation MIT-Press* **1997**.
10. Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* **2023**.
11. OpenAI. ChatGPT, 2023.

 doi:10.20944/preprints202412.2432.v2

12. Redmon, J. You only look once: Unified, real-time object detection. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.
13. Microsoft. Microsoft Copilot: Your AI companion, 2023.