Designing a protocol to constraint AGI freedom

Cristian Curaba

January 3, 2024

Abstract

In this article, I will propose a new concept in the field of AGI: freedom. It is fundamental to evaluate the power of an AGI system. Moreover, by reasoning with this concept, we may find new ways of controlling and handling risks. I will also propose a way to constrain the AGI system and handle risky behaviours for humans. The key idea is to force the AGI system to send outputs through a programmed protocol by adding an "infinite" penalty term to the goal function (or fine-tuning it after the learning phase)[1].

The protocol can implement a monitoring procedure and send a warning to the programmers if the AGI system is going to do something dangerous[2]. It can be updated safely (avoiding adversarial updates) thanks to the digital signature of the programmers. Therefore, we can solve the stop button problem by updating the protocol.

1 Why freedom?

In the AGI literature, two key concepts are accepted to describe an AGI system: intelligence and goal. The intelligence is defined as the ability to achieve a goal. The goal is practically formalized by the score/loss/cost/goal function which is the one that any learning system tries to optimize in the learning phase. Wide and general goals characterize the AGI systems. Literature is trying to align the AGI systems with human desire but there are still no clear solutions.

I want to propose a new concept in the field: freedom. Freedom is the variety of actions that the system can perform in the world to achieve its goal. The power of the AGI system may be limited by the freedom that we (the company programmers) have granted to it. In the state of art of foundational models, even if they seem to understand the world and show highly intelligent behaviours, their freedom is still forceful constrained: they can just answer to questions. Can we design a way to constrain future AGI systems to be safe for humans? One desired propriety is the ability to shut down the system (also

[1] It's not straightforward how to do it since it is dependent on the learning technique applied. We, of course, have to penalize also any unauthorized modifications.

[2] This type of prediction is challenging and, when the AGI is misaligned, it will try to deceive and hide behaviours we consider risky.

known as the "stop button problem" [Soa+15]). In the following section, I will design a way to constrain the AGI system and, as by product, to solve the stop button problem.

2 Building a protocol with digital signature

The key idea is to force the AGI system to send outputs through a programmed protocol. This protocol can also include a monitoring system to evaluate the AGI outputs. The protocol can be updated safely by the company programmers based on the digital signature of the programmers. Here I am supposing that the AGI system is equipped with a local and secure memory where the public keys of the company programmers and the whole protocol are stored[3]. Check digital signature schemes [AZJ06] for more details.

To ensure that the AGI system will go through the protocol, we will need to train it to do so by simply penalizing (virtually $-\infty$) the outputs that are not sent through the protocol. We also need to penalize any attempt to modify the protocol without the digital signature of the company programmers. In this way, the AGI system will learn to send outputs through the protocol "by design" and avoid modifying it. Adversarial attacks on the protocol will be ineffective because the AGI system will update the local protocol only if the digital signature of the company programmers is present.

3 Solving the stop button problem

How can we design a protocol to solve the stop button problem? We (the company programmers) can just update the protocol to encrypt the AGI outputs in such a way that only the company programmers can decrypt them. In this way, instead of shutting down the AGI system "physically", we effectively design a way to make its outputs completely useless (basically a noise) while being able to communicate with the programmers to re-align it (if possible).

4 Other application of the protocol

The protocol can be used to monitor the AGI system. For example, we can design a protocol to monitor the AGI system and send a warning to the programmers if the AGI system is going to do something dangerous.

It can be used to encrypt the AGI outputs and, with this, all the advantages of encrypted messages.

[3]It's fundamental to fine-tune (or learn) the AGI with the original protocol in local memory: we want to avoid adversarial protocols which may use the identical public key and thus allow the AGI to go through an unknown adversarial protocol instead of the correct one. We must make sure, when designing the goal function for this purpose, to uniquely identify the correct address where the original protocol is stored.

5 Bottlenecks, criticals and food for thought

- Since we don't know the learning technique which will create the first AGI, can we adapt this method to every learning technique? This is not clear to me, it would be great if we find a way to fine-tune the AGI after the learning process: in this way, we can adopt this protocol method a posteriori.
- Can the AGI split (copies) himself in submodules which don't send outputs in the protocol? Yes, this could happen and it's a problem. Even if, theoretically, the submodules are created to optimize the goal function we can't be certain that each of them will send outputs through the protocol. This is not clear and should be tested (how?) but I think that the submodules can just take care of partial operations, without the need to optimize the entire goal function. Let me explain it with an easy example: let's suppose that the goal of an AGI is to output a pair containing the sum of the input numbers and the product of the input numbers. The AGI could delegate the summing operations to a copied submodule and the product to another. Thus, each module is optimizing just a component of the goal function. Then the AGI can collect and send the results. This is critical, in general, for the alignment problem: even if the AGI is properly aligned, it could split-copying in submodules which may behave dangerously or in an unexpected way. My intuition is that the AGI will avoid this naive split since the submodules may behave against his goal function but further investigations are required.
- What if the company lose (or compromises) the private key of the digital signature? We could design a series of protocols with different keys and different levels of security. In this way, we avoid a single point of failure but we also increase the overload on the system (which seems negligible to me).
- Since the protocol is open-source, can the misaligned AGI learn to mathematically bypass the undesired encryption contained by the shut-down update? No, it would need to solve the hard-mathematical problem that we assume to be unbreakable even for super-intelligent AGI (I confidently hope that we will get a theoretical framework in cybersecurity satisfying this assumption).

6 Further work

It's not evident to me how to design a protocol to monitor the AGI system and send a warning to the programmers if the AGI system is going to do something risky. Thus, it is fundamental to design and test a monitoring procedure capable of detecting dangerous behaviours. It may also be useful to standardize security thresholds for the monitoring procedure to avoid unfair competition between companies.

We may also want to test if fine-tuning (to send outputs through the protocol and not modify it) the AGI system after the learning process is possible and how to do it effectively. Further work is needed to test this.

Thinking about other advantages that a protocol can obtain: other than safety concerns, the company write a protocol for licensing or stuff. The more appealing the protocol advantages are, the more the company will be incentivized to adopt it (decreasing the so-called "safety/alignment tax").

7 Conclusion

In this article, I proposed a new concept to work with in the AGI field: freedom. Moreover, by reasoning with this concept, we may find new ways of controlling and handling risks. I also proposed a way to constrain the AGI system and handle detected risky behaviours for humans: the key idea is to force the AGI system to send outputs through a programmed protocol by adding an "infinite" penalty term to the goal function (or fine-tuning it after the learning phase). We can avoid unauthorized modifications by penalizing them in the same way. The protocol can implement a monitoring procedure and send a warning to the programmers if we spot risky behaviours. It can be updated safely (avoiding adversarial updates) thanks to the digital signature of the programmers. Therefore, we can solve the stop button problem by updating the protocol. Numerous bottlenecks and criticals are still open and need further work to be solved. I hope that this article will be useful to the community and will stimulate further research.

References

[AZJ06] Hermina Alajbegović, Dževad Zečić, and Hasan Jamak. "DIGITAL SIGNATURE ALGORITHM (DSA)". In: Sept. 2006.

[Soa+15] Nate Soares et al. "Corrigibility". In: *Workshops at the twenty-ninth AAAI conference on artificial intelligence*. 2015.