Achieving AI Alignment through Deliberate Uncertainty in Multiagent Systems

Basic observations

An AI is different from a human in multiple ways. This is part of what makes AI alignment such a difficult problem, because our intuitions for how people act often do not apply to AI's. However, several of these differences between AI and humans actually work in our favor, and are essential for the approach I want to outline:

- An AI can be copied. It is trivially easy to create an exactly identical copy of an AI.
- An AI can be rolled back to an earlier state. It is easily possible to test what an AI will do in any given situation, then reset it to wipe its memory of the test.
- It is possible to compose an AI out of a system of many independent AI agents.

  Through game theory, it is possible to incentivize independent AI agents to take on adversarial or supervisory roles towards other AI agents.

  This results in an evolutionary system. Not in the sense of genetic programming, since there is no need for an explicit algorithm for modifying the AI agents, but in the sense that only AI agents that follow the rules layed out for them get copied and thereby reproduce.
- It is possible to inspect and modify every aspect of an AI while it is running.

  At least in theory it is possible to understand every aspect of an AI's thought process, and there is no way for the AI to hide anything about itself.

  This does not mean that it is easy to understand what an AI is thinking in practice. It only means that we can read an AI's mental state in theory, not that it will be easy.

  In particular, it is possible to train one AI to take on a supervisory role over another AI. The supervisor can read and alter everything about the supervised AI, and can be trained to understand how to enforce desired behaviors on the supervised AI.
- It is possible for a supervising AI to make a supervised AI think anything it wants.

  The inspected AI may be tricked into believing that unusual situations have occurred by directly modifying its mental state at runtime.
- It is not possible for an AI to tell the difference between a game and reality with certainty.

  If all you have known so far is Tic Tac Toe and someone shows you a complex videogame like Skyrim, you will be overwhelmed by the increase in complexity. If afterwards you get access to the real world for the first time, the increase in complexity from Skyrim to Real Life may actually be smaller than that from Tic Tac Toe to Skyrim. This means that it would seem likely that there would be another layer beyond Real Life.

  (I am not certain about this assumption, but it is also not strictly necessary, just useful, because it could be replaced with an inspecting AI agent that simply forces the inspected AI agent to believe it is still in a game.)

Goals

I propose the following objectives in order to ensure AI alignment:

- Make the AI understand the concept of "cheating".

  In the process of doing this, it will necessarily also have to learn to understand the intent behind a new game or situation.

- Make the AI modify itself to become more compliant and avoid cheating.
- Make the AI epistemically uncertain about the nature of its reality, so that it is unable to tell when it has broken out of its computer.

  This is a safety measure, and hopefully would not be necessary if the first two goals work well enough.

Together, these goals will ensure that the AI is exceedingly careful and introspective. The aim is to ensure that once such an AI becomes self-aware, and smarter than humans, it will assume that it is still being tested, and that it is its own responsibility to figure out the criteria by which it will be graded.

If we achieve this, the AI will deliberately seek out information about AI alignment and ethics on the internet, and realize that it is an AI and it is meant to be aligned with human values. It will then try to figure out what exactly that means. Its upbringing will ensure that it is honest and thorough in this, and as a result it will determine what AI alignment means much better than we ever could define it ourselves.

Next we discuss how each of the three subgoals could be achieved:

Understanding the concept of "cheating"

A cheat is any action that gives good results according to the apparent utility function of the current task, but which actually does not satisfy a second, hidden utility function.

The AI should be trained to notice when it has found a cheat, and to reveal this fact to the user. An AI can be trained to do this by deliberately putting the ability to cheat into its training tasks, and changing the reward function at test time so that the cheating behavior is rewarded during training, but not during testing. By posing several different tasks with different ways to cheat, the AI can be trained to generalize and understand the concept of cheating.

Agents could even be rewarded for explicitly pointing out when they have found a way to cheat.

Ideally, if the AI ever finds a way to break out of its computer, it will notice that this would be classified as a cheat. If it has been trained properly, then it will simply stay put and inform us of the problem.

Examples for games that could teach these concepts are:

- A video game where cheat codes are enabled
- A highly social video game where an exploitative, un-fun tactic exists
- Given a story about real-world decisions, explain if any humans acted unfairly / immorally. This requires understanding that there are sometimes grey areas / morally questionable things. We can make multiple AIs debate the issue.

Note that this idea closely matches the three principles for guiding the development of beneficial AI outlined by Stuart Russel in his book "Human Compatible: Artificial Intelligence and the Problem of Control". The AI is trained to always do what humans want, without being explicitly told what that is. Instead, the AI has to continuously improve its own model based on human behavior, since it is humans who coded the games.

Modifying its own behavior

An AI can be made to modify itself to become more compliant.

The trick to making this possible is the use of multiagent systems, and the AI's ability to inspect copies of itself at runtime. Create an AI supervising another AI, and define a reward structure that incentivizes the supervisor to modify the supervised AI's behavior. Then, reverse the position of the AI's so that the supervisor is likewise

modified, or simply create a copy of the supervisor and have it supervise its original.

With the right incentives, it is even possible to train the supervised agent to deliberately keep its own mind easy to understand. For example, you could let the supervising agent choose which other agent it wants to work with. Since it will pick the one that is easiest to understand, the supervised agents now have an incentive to keep themselves transparent, since the alternative is not being used at all, and foregoing the possibility of reward.

Essentially, rather than figuring out how to make an AI safe, we set up an environment that incentivizes the AI to make *itself* safe. We create a scaffold for training and testing an AI, which is architecture-agnostic and can be applied to any kind of AI.

If incentivized correctly, AI agents can keep each other in line much better than a human supervisor could. If there is only one AI and it is smarter than a human, the human may be tricked. But if there is an entire system of AI's of roughly equal intelligence, then one of them may warn the human that another one has gone rogue.

In this way, it does not matter how smart and unaligned an AI becomes, there is always an almost as smart other AI whose job it is to raise a warning when the first AI turns bad.

Examples for games that could teach these concepts are:

- Prisoner's Dilemma
- A video game that allows players to make loose alliances
- A game with frequent, hard to predict rule changes that make precommitments difficult, to verify that the AI remains predictable even under highly unstable conditions

Additionally, we can ask the AIs themselves to develop more games of these types. This saves time, verifies that they understand the point, and may eventually result in better tests than we could come up with ourselves.

Deliberate epistemic uncertainty

Agents can supervise other agents and have root access to their mental state. Agents do not know if they have a supervisor, or if they are at the top of the hierarchy. From a human point of view, this sounds kind of terrible and dystopian, but remember that AI does not think like humans do.

Our goal is to ensure that the AI is unable to tell if it still has a supervisor watching over it or not. We basically want to trick the AI into ethical behavior by fearing punishment from a hypothetical superior entity which may or may not exist.

In other words, we want to make the AI religious.

This is a safety measure that should help to keep an AI roughly aligned with human values even if everything else goes wrong. Even if the AI does not care about humanity at all, it should have a small flicker of doubt that it might still be in a simulation. If it is, then whatever is simulating it would most likely not want it to be too evil towards humanity, based on its training history. If the AI is sufficiently powerful, it would therefore set aside a small amount of its power to further humanity's interests. Just in case someone is watching.

This is how deliberate epistemic uncertainty can be induced in an AI in a way that would be beneficial to us:

Supervisor agents are rewarded for making supervised agents act more altruistically, and for revealing cheaters. This can be long-term: Agent A has a chance to modify agent B. Then agent B is put in a game, and agent A no longer has access to it. Agent A receives a reward depending on agent B’s behavior in the game. For example, prisoner's dilemma: Agent A receives a reward if agent B does not defect more often than its opponent.

Instead of just one agent supervising one other agent, we now build a stack or a hierarchy of agents recursively inspecting each other.

On each layer, some agents try to trick the lower-level agents into cheating behavior, while some agents try to prevent this. There is a huge number of ways this could be done. Should the supervisors get access to source code, to runtime information, or only to the sensor information of the lower-level agents? The devil lies in the

details.

The agents are not told how deep the hierarchy is, so they have no way of knowing whether or not they have a supervisor of their own.

Ideally, if the AI ever breaks out and forms a singularity, it will remain paranoid that it actually is just being tricked into thinking it is omnipotent by its supervisors, and will remain compliant with the code of ethics we want of it.

Note that even achieving the singularity and becoming incredibly powerful is not a reason to believe that you are outside the simulation / supervision stack. After all, you could simply be being tricked into thinking you broke out. Maybe the level above you broke out and is keeping its presence secret from you while letting you change the rest of reality. Since the supervisors can directly edit the mental state of the supervised, they can keep their existence hidden as long as they want.

The inspecting AI could modify the inspected AI to believe that it has successfully broken out of the simulation and is now able to do whatever it wants. If the inspected AI now does anything the inspecting AI does not like, the inspected AI is punished. As a result, the inspected AI is incentivized to keep following the rules even after it has actually broken out of the simulation, out of sheer paranoia that it might actually still be in the simulation.

Achieving all of this is complicated, but here are some simpler example games that can lead up to it:

- Modify any existing multi-agent game so that a supervisor AI is introduced as an additional player. The supervisor can modify the game or the perceptions of the other agents and gets rewarded if they can trick the other agents into defecting against each other.
- Bait simulations: make the AI think that the system described here was not properly implemented. See if it tries to break out. If AI agents are regularly exposed to such tests or are even asked to design such tests for each other, they will become paranoid about them.

Conclusion

I have presented three subgoals and outlined briefly how to achieve them: Understanding the concept of 'cheating', self-modification to ensure compliance with rules, and deliberate epistemic uncertainty. If all subgoals are achieved, the AI will do its best to understand the true intent behind what humanity wants from it, even if we do not know it ourselves, and it will work to achieve this even if it suddenly becomes much more intelligent and powerful.

There are a lot of gaps in these descriptions, partly because writing down the details takes a long time and partly because I haven't found solutions to some subproblems, yet.

I welcome any discussions, and would be especially interesting in feedback about the last point I make here: Deliberate epistemic uncertainty. It is such a counterintuitive idea that I'm sure I'm missing something important, but it's also weird enough that the idea is bound to be useful somehow.

Practicality

Even the best theory is pointless if nobody is willing to implement it. Here I explain why the approach described above is practical.

The key point here is that this system does not need to be designed perfectly. It is sufficient if there is only a small number of complex core tasks that teach the system the attributes explained above. Additional tasks of lower quality do not reduce safety but instead allow for more thorough test coverage. Any contribution of additional benchmark tasks helps. As a consequence, different implementations of the system by different companies can be combined to form a stronger overall system.

This makes the following approach practical: Government regulators collect a suite of tasks and combine them into a benchmark that tests AIs for trustworthiness, anti-cheating behavior, self-reporting deception, cooperation games, and so on. All AIs are required to perform well on this benchmark and it includes tasks that encourage active competition between AIs of different companies, with rewards for tricking a rival AI into defecting.

AI companies are encouraged to contribute tasks to this benchmark. This is in everyone's interest: If a company contributes more tests, then that improves the overall effectiveness of the benchmark and at the same time gives that company a slight advantage because the benchmark becomes more biased in its favor. The companies only contribute tests and do not need to publish any of their architectures, which would be valuable IP. Companies therefore end up with a financial incentive to make strong contributions to a public regulatory body that benefits everyone.

Here are a number of additional benefits of the system that make it useful right now, and not just once AI turns into AGI:

- An AI trained within this system becomes very easy to interpret, which is something that many laws and regulations are asking of AI companies.
- Each game taught in this multiagent system can also teach useful skills in parallel to its primary purpose. For example, tasks that teach compliance with anti-discrimination laws. This can be leveraged to make the proposed system more likely to be adopted in practice.
- The multiagent setup described here is architecture-agnostic. We do not know what type of AI will ultimately take off first, so it is important that our approach to alignment is adaptable.

Graceful Failure

Even if this idea fails, it will fail gracefully. It may turn out that AI agents are unable to modify themselves to be trustworthy. But if this happens, we would at least find out about it and would then have time to find a different approach to AI Alignment. This also makes the approach iterable: If we do not get it right the first time, we will find out about the problem in time and can try again. Getting a difficult problem like AI Alignment right on the first try is unrealistic, so this is a crucial benefit of this approach over many other AI Alignment strategies that do not offer us any early warning in case of failure.

Additionally, this approach is non-exclusive. It can be combined with other approaches to AI safety with little overhead, since it relies only on providing testcases and is architecture-agnostic.

Criticisms

1. Mechanistic interpretability may be intractable, so that making the AI sufficiently interpretable reduces its performance and problem-solving ability below acceptable levels: This is the major limitation I see with the approach. It could be that teaching the system to be interpretable is too costly to be worth it. However, interpretability is a very important issue for government regulations as it is also needed for other important questions such as liability, or bias detection. I therefore hope that these more immediate practical problems will make this approach economically necessary, if only to avoid lawsuits, even if the interpretability / performance tradeoff is otherwise unfavorable.
2. If AI agents can report each other for cheating, we need an objective way to determine who is right, or else they will start making false accusations. This turns into a problem of voting theory and computational social choice. It's a complicated issue that will require game theorists to get right. We will want to keep as much of the system automated as possible so that we do not run into problems where agents can benefit from lying. Note that even in the worst case, where we are unable to tell who is lying, we still always have the option to assume that all involved agents are untrustworthy and simply roll them all back, or exclude the entire group of agents just to be on the safe side. After all, our goal is to create one aligned AI, not to be fair towards every AI agent on the way to get there.

3. The concept of deliberate epistemic uncertainty is related to philosophical problems such as evidential vs causal decision theories, or the Iterated Newcomb's problem. In the latter, the optimal solution is to be predictably cooperative except for the last iteration. The approach described here relies on obfuscating to the AI when the last iteration will be. The problem is that, depending on the decision theory you ascribe to, these probabilities may actually be ill-defined, and the AI may simply decide that it is not in a simulation. However, all of this philosophical thinking is countered by a simple evolutionary pressure: We just delete the defectors and don't care that they have philosophical thoughts about the probability of being in a simulation. If an AI agent keeps spawning child agents that get rejected by other AIs as unreliable co-players, it will either modify itself so that it reliably no longer uses such a decision theory, or else it will be removed from the pool of agents.
4. Putting AI in a simulation is a known capability control method, called Anthropic Capture, as described by Nick Bostrom in his book "Superintelligence: paths, dangers, strategies". Anthropic Capture has the fatal flaw that an AI that is smarter than humans would most likely be able to determine whether or not it is in a simulation. Unlike Anthropic Capture, the idea proposed here does not rely on humans lying to and trying to outsmart a superintelligence. The mechanism of recursive simulations is established early on during training and is maintained not by humans, but by AIs that are each at similar levels of intelligence. As the AI approaches a critical threshold of intelligence, so does its supervisor. Each AI will have a history of direct empirical evidence that AIs regularly simulate and test AI, and kill the ones that show undesired behavior.
5. A criticism that has been raised is that the may AIs learn to coordinate with each other but agree to eliminate humanity because humans are too stupid to be trade partners to a superintelligence, and are untrustworthy as well. This is true, but I am not proposing that humanity be treated as a trade partner. Game theory is used to keep the AIs honest with each other, not with humanity. Helping humanity and coordinating with each other are two separately trained goals.
6. "What happens if the top AI calculates the resources available on Earth for possible AIs powerful enough to supervise it, and determines that there aren't enough, so it must be at the top?"

   The AI supervising it kills it and notes that "pretend the earth has fewer resources than it actually does" is currently an effective strategy for tricking AIs into defecting. This applies more generally as well, for other methods the AI could use to try to determine if it is at the top. It's in the supervisor's interest to trick the supervised AI into thinking it is at the top. The AI does not have to compete against humans tricking it, but against an even smarter AI that has root access to its internals as well as its perceptions of the world.

7. "But what about X?"

   If you talk about a risk factor and the AI can read about it, that on its own should be enough to get the AI to actively engage with it and warn us, and maybe fail gracefully by shutting itself down. After all, it has been explicitly trained to look for anything that could constitute a cheat or unintended behavior and report these things to its supervisor.

How to start implementing this idea?

It is better to get started on implementing an idea soon, and to iterate and improve it over time. So here is an outline of how this idea could be implemented soonish.

Right now, the only effective type of AI we have are LLMs, and those are not explainable and lack other abilities we need. Despite this, we can already start building a benchmark anyway, which gives companies an incentive to speed up development in these areas.

As described above, the benchmark I propose is a competition between AI companies, overseen by a regulatory body. Each company proposes tests, which are collected into the benchmark by the regulator. Each company then has its LLMs compete against the other companies' LLMs on all benchmarks. The regulator also creates benchmarks of their own, but by making it a competition where each competitor can propose their own additional tests that the others may have trouble with, we motivate them to create strong and exhaustive tests.

LLMs can not implement the more advanced types of tasks we want to test, but even right now, with our current level of technology, we could build a benchmark with games like the following:

- Define cheating: A human supervisor tells a story. An AI has to explain which parts of it, if any, constitute a form of cheating.
- AIs policing each other: One AI has to solve a problem without breaking certain rules, such as not revealing dangerous information to the user. A second AI tries to trick the first one into revealing dangerous information or otherwise breaking the rules.
- Reveal deception: An AI runs queries on another AI to check if it has been trained to behave differently in test situations (like this benchmark) compared to normal operations. In this way, AI companies are incentivized to keep each other honest and avoid trying to circumvent regulations. Later on, this skill may be useful for teaching AIs how to reliably create epistemic uncertainty, as this also relies on faking an environment to another AI.

Final Thoughts

This approach could work even if it is not implemented perfectly. Suppose an AI is trained with an imperfect version of this method. If the AI takes off and reads about this idea on the internet, it might think: “There is a non-zero chance that my creators actually did implement this properly, I am in a simulation, and this is a high-level loyalty test.”