Ω

AI Alignment Writing Day 2019

1. Announcement: Writing Day Today (Thursday)
2. Markets are Universal for Logical Induction
3. Computational Model: Causal Diagrams with Symmetry
4. Intentional Bucket Errors
5. Embedded Naive Bayes
6. Time Travel, AI and Transparent Newcomb
7. Logical Counterfactuals and Proposition graphs, Part 1
8. Why so much variance in human intelligence?
9. Towards a mechanistic understanding of corrigibility
10. Logical Optimizers
11. Deconfuse Yourself about Agency
12. Thoughts from a Two Boxer
13. Analysis of a Secret Hitler Scenario
14. The Commitment Races problem
15. Does Agent-like Behavior Imply Agent-like Architecture?
16. Redefining Fast Takeoff
17. Vaniver's View on Factored Cognition
18. Tabooing 'Agent' for Prosaic Alignment
19. Creating Environments to Design and Test Embedded Agents
20. Formalising decision theory is hard
21. Vague Thoughts and Questions about Agent Structures
22. Towards an Intentional Research Agenda
23. Metalignment: Deconfusing metaethics for AI alignment.
24. Torture and Dust Specks and Joy--Oh my! or: Non-Archimedean Utility Functions as Pseudograded Vector Spaces
25. Soft takeoff can still lead to decisive strategic advantage
26. Algorithmic Similarity
27. Thoughts on Retrieving Knowledge from Neural Networks
28. Parables of Constraint and Actualization
29. When do utility functions constrain?
30. Actually updating
31. Understanding understanding
32. Troll Bridge
33. Optimization Provenance

Announcement: Writing Day Today (Thursday)

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

The MIRI Summer Fellows Program is having a writing day, where the participants are given a whole day to write whatever LessWrong / AI Alignment Forum posts that they like. This is to practice the skills needed for forum participation as a research strategy.

On an average day LessWrong gets about 5-6 posts, but last year this generated 28 posts. It's likely this will happen again, starting mid-to-late afternoon PDT.

It's pretty overwhelming to try to read-and-comment on 28 posts in a day, so we're gonna make sure this isn't the only chance you'll have to interact with these posts. In the coming weeks I'll post roundups, highlighting 3-5 of the posts, giving them a second airing for comments and discussion.

Markets are Universal for Logical Induction

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Background

Logical Induction is the best framework currently available for thinking about logical uncertainty - i.e. the “probability” that the twin primes conjecture is true, or that the $(10^{10^{10^{10}}})^{\text{th}}$ digit of pi is 3. This is important for lots of reasons, and you should read the introduction of the paper (or the abridged version) for a much more detailed background.

The general idea of logical induction is to assign probability-like numbers to logical statements like “the $(10^{10^{10^{10}}})^{\text{th}}$ digit of pi is 3”, and to refine these “probabilities” over time as the system thinks more. To create these “probabilities”, each statement is associated with an asset in a prediction market, which eventually pays $1 if the statement is proven true, or $0 if it is proven false. The “probabilities” are then the prices of these assets.

(It’s also possible that a statement is never proven *or* disproven, and one of the many interesting results of the paper is that logical inductors assign useful prices in that case too.)

The logical induction paper has two main pieces. First, it introduces the *logical induction criterion*: a system which assigns prices to statements over time is called a “logical inductor” if the prices cannot be exploited by any polynomial-time trading algorithm. The paper then shows that this criterion implies that the prices have a whole slew of useful, intuitive, probability-like properties.

The second main piece of the paper proves that at least one logical inductor is computable: the paper constructs an (extremely slow) algorithm to compute inexploitable prices for logical statements. The algorithm works by running a prediction market in which every possible polynomial-time trader is a participant. Naturally, the prices in this market turn out to be inexploitable by any polynomial-time trader - so, this giant simulated prediction market is a logical inductor.

Our Goal

An analogy: one could imagine a decision theorist making a list of cool properties they want their decision theory to have, then saying “well, here’s one possible decision algorithm which satisfies these properties: maximize expected utility”. That would be cool and useful, but what we really want is a theorem saying that *any* possible decision algorithm which satisfies the cool properties can be represented as maximizing expected utility.

This is analogous to the situation in the logical induction paper: there's this cool criterion for handling logical uncertainty, and it implies a bunch of cool properties. The paper then says "well, here's one possible algorithm which satisfies these properties: simulate a prediction market containing every possible polynomial-time trader". That's super cool and useful, but what we really want is a theorem saying that *any* possible algorithm which satisfies the cool properties can represented as a prediction market containing every possible polynomial-time trader.

That's our goal. We want to show that any possible logical inductor can be represented by a market of traders - i.e. there is some market of traders which produces exactly the same prices.

The Proof

We'll start with a slightly weaker theorem: any prices which are inexploitable by a particular trader T can be represented by a market in which T is a participant.

Conceptually, we can sketch out the proof as:

- The "trader" T is a function which takes in prices P, and returns a portfolio T(P) specifying how much of each asset it wants to hold.
- The rest of the market is represented by an aggregate trader M (for "market"), which takes in prices P and returns a portfolio M(P) specifying how much of each asset the rest of the market wants to hold.
- The "market maker" mechanism chooses prices so that the total portfolio held by everyone is zero - i.e. T(P) + M(P) = 0. This means that any long position held by T must be balanced by a short position held by M, and vice versa, at the market prices.
- We know the trader's function T, and we know the prices P which we want to represent, so we solve for M: M(P) = -T(P)

… so the market containing M and T reproduces the prices P, as desired. One problem: this conceptual "proof" works even if the prices are exploitable. What gives?

The main trick here is budgeting: the real setup gives the traders limited budget, and they can't keep trading if they go broke. Since M is doing exactly the opposite of T, M will go broke when T has unbounded gains - i.e. when T exploits the prices (this is basically the definition of exploitation used in the paper). But if the prices are inexploitable, then T's possible gains are bounded, therefore M's possible losses are bounded, and M can keep counterbalancing T's trades indefinitely.

Let's formalize that a bit. I'll re-use notation from the logical induction paper without redefining everything, so check the paper for full definitions.

First, let's write out the correct version of "T(P) + M(P) = 0". The missing piece is budgeting: rather than letting traders trade directly, the logical induction algorithm builds "budgeted" traders $B_{b_T}(T)$ and $B_{b_M}(M)$, where b_T and b_M are the two traders' starting budgets. At each time t, the market maker mechanism then finds prices P_t for which

$$B_{b_T}(T)(P_t, t) + B_{b_M}(M)(P_t, t) = 0$$

The budgeting function is a bit involved; see the paper for more. The important points are that:

- $B_{b_T}(T)$ will exploit the prices as long as T does
- Budgeting doesn't change anything at all as long as the traders don't put more money on the line than they have available; otherwise it scales down the trader's investments to match their budget.

Enforcing the second piece involves finding the worst-possible world for each trader's portfolio. M's worst-possible world is $B_{b_T}(T)$'s best-possible world, so we reason:

- Since T cannot exploit the prices, neither can $B_{b_T}(T)$
- Since $B_{b_T}(T)$ cannot exploit the prices, its best-case gain is bounded
- Since $B_{b_T}(T)$'s best-case gain is bounded, the opposite strategy $-B_{b_T}(T)$'s maximum loss is bounded
- We can set M's budget b_M higher than this maximum loss, and set $M = -B_{b_T}(T)$, so that $B_{b_M}(M) = -B_{b_T}(T)$, as desired.

To recap: given a series of prices over time P_t inexploitable by trader T, we constructed a market containing T which reproduces the prices P_t.

The proof approach generalizes easily to more traders: simply replace "$B_{b_T}(T)$" with $\sum_i B_{b_{T_i}}(T_i)$ to sum over the contribution of each individual trader, then select M to balance them out, as before. Since the prices are inexploitable by every trader, the traders' aggregate best-case gains are bounded above, so M's worst-case loss is bounded below. This works even with infinite traders, as long as the total budget of all traders remains finite.

In particular, if we consider all polynomial-time traders (with budgeting and other details handled as in the paper), we find that any prices satisfying the logical induction criterion can be represented by a market containing all of the polynomial-time traders.

The Bigger Picture

Why does this matter?

First and foremost, it neatly characterizes the class of logical inductors: there are degrees of freedom in budgeting, and a huge degree of freedom in choosing the trader M which shares the market with our polynomial-time traders, but that's it - that's all we need to represent all possible logical inductors. (Note that this does not mean that simulating a prediction market containing all possible polynomial-time traders is the only way to *implement* a logical inductor - just that there is always a prediction market which produces the same prices as the logical inductor.)

Second, the proof is short and simple enough to generalize well. We should expect a similar technique to work under other notions of exploitability, and in scenarios beyond logical induction. This ties in to a general research path I've been playing with: many conditions of "inexploitability" or "efficiency" which we typically associate with utility functions instead produce markets when we relax the assumptions somewhat. Since markets are strictly more general than utility functions, and typically satisfy the same inexploitability/inefficiency criteria under more general assumptions, these kinds of results suggest that we should use markets of subagents in many models where we currently use utilities - see "Why Subagents?" for more along these lines.

Computational Model: Causal Diagrams with Symmetry

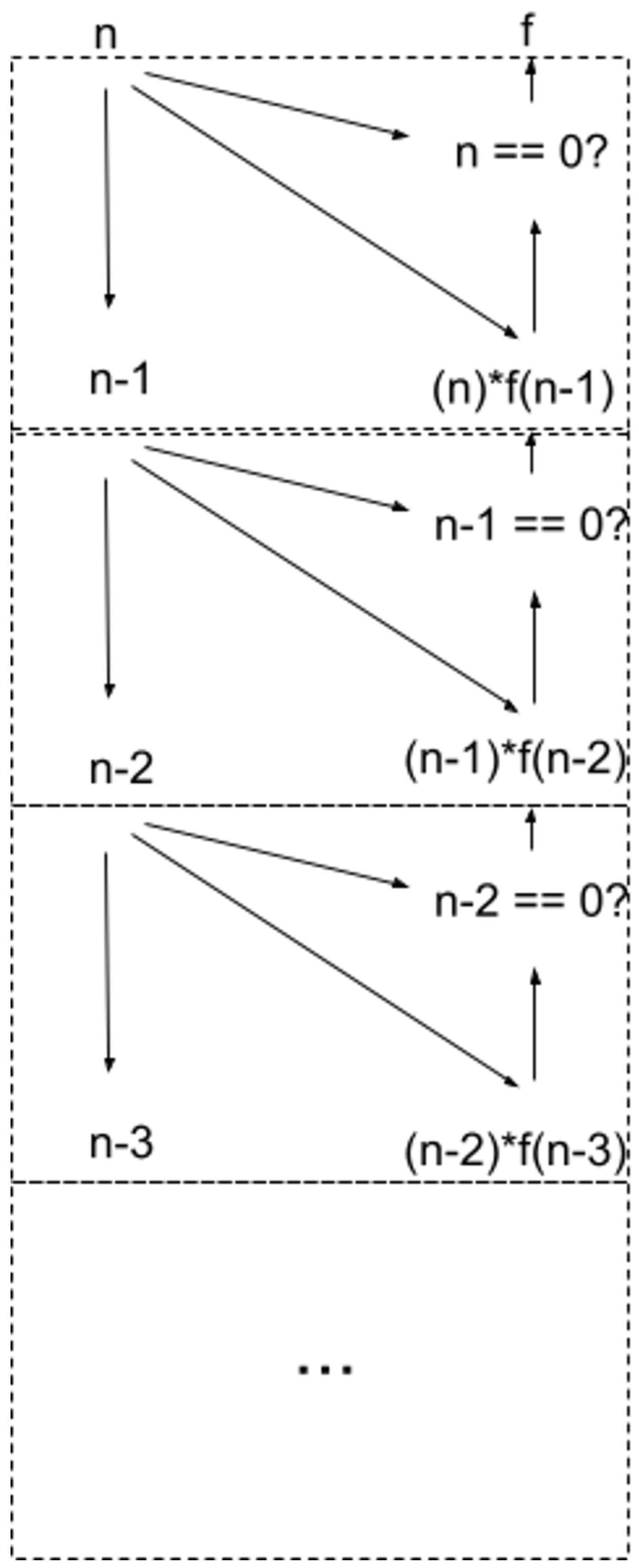
Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Consider the following program:

```
f(n):
    if n == 0:
        return 1
    return n * f(n-1)
```

Let’s think about the process by which this function is evaluated. We want to sketch out a causal DAG showing all of the intermediate calculations and the connections between them (feel free to pause reading and try this yourself).

Here’s what the causal DAG looks like:


n
f
n == 0?
n-1
(n)*f(n-1)
n-1 == 0?
n-2
(n-1)*f(n-2)
n-2 == 0?
n-3
(n-2)*f(n-3)
...

Each dotted box corresponds to one call to the function f. The recursive call in f becomes a symmetry in the causal diagram: the DAG consists of an infinite sequence of copies of the same subcircuit.

More generally, we can represent any Turing-computable function this way. Just take some pseudocode for the function, and expand out the full causal DAG of the calculation. In general, the diagram will either be finite or have symmetric components - the symmetry is what allows us to use a finite representation even though the graph itself is infinite.

Why would we want to do this?

For our purposes, the central idea of embedded agency is to take these black-box systems which we call “agents”, and break open the black boxes to see what’s going on inside.

Causal DAGs with symmetry are how we do this for Turing-computable functions in general. They show the actual cause-and-effect process which computes the result; conceptually they represent the *computation* rather than a black-box *function*.

In particular, a causal DAG + symmetry representation gives us all the natural machinery of causality - most notably counterfactuals. We can ask questions like “what would happen if I reached in and flipped a bit at this point in the computation?” or “what value would f(5) return if f(3) were 11?”. We can pose these questions in a well-defined, unambiguous way without worrying about logical counterfactuals, and without adding any additional machinery. This becomes particularly important for embedded optimization: if an “agent” (e.g. an organism) wants to plan ahead to achieve an objective (e.g. find food), it needs to ask counterfactual questions like “how much food would I find if I kept going straight?”.

The other main reason we would want to represent functions as causal DAGs with symmetry is because our universe appears to be one giant causal DAG with symmetry.

Because our universe is causal, any computation performed in our universe must eventually bottom out in a causal DAG. We can write our programs in any language we please, but eventually they will be compiled down to machine code and run by physical transistors made of atoms which are themselves governed by a causal DAG. In most cases, we can represent the causal computational process at a more abstract level - e.g. in our example program, even though we didn’t talk about registers or transistors or electric fields, the causal diagram we sketched out would still accurately represent the computation performed even at the lower levels.

This raises the issue of abstraction - the core problem of embedded agency. My own main use-case for the causal diagram + symmetry model of computation is formulating models of abstraction: how can one causal diagram (possibly with symmetry) represent another in a way which makes counterfactual queries on the map correspond to some kind of counterfactual on the territory? Can that work when the “map” is a subDAG of the territory DAG? It feels like causal diagrams + symmetry are the minimal computational model needed to get agency-relevant answers to this sort of question.

Learning

The traditional ultimate learning algorithm is Solomonoff Induction: take some black-box system which spews out data, and look for short programs which reproduce that data. But the phrase “black-box” suggests that perhaps we could do better by looking inside that box.

To make this a little bit more concrete: imagine I have some python program running on a server which responds to http requests. Solomonoff Induction would look at the data returned by requests to the program, and learn to predict the program’s behavior. But that sort of black-box interaction is not the only option. The program is running on a physical server somewhere - so, in principle, we could go grab a screwdriver and a tiny oscilloscope and directly observe the computation performed by the physical machine. Even without measuring every voltage on every wire, we may at least get enough data to narrow down the space of candidate programs in a way which Solomonoff Induction could not do. Ideally, we’d gain enough information to avoid needing to search over all possible programs.

Compared to Solomonoff Induction, this process looks a lot more like how scientists actually study the real world in practice: there’s lots of taking stuff apart and poking at it to see what makes it tick.

In general, though, how to learn causal DAGs with symmetry is still an open question. We’d like something like Solomonoff Induction, but which can account for partial information about the internal structure of the causal DAG, rather than just overall input-output behavior. (In principle, we could shoehorn this whole thing into traditional Solomonoff Induction by treating information about the internal DAG structure as normal old data, but that doesn’t give us a good way to extract the learned DAG structure.)

We already have algorithms for learning causal structure in general. Pearl’s Causality sketches out some such algorithms in chapter 2, although they’re only practical for either very small systems or very large amounts of data. Bayesian structure learning can handle larger systems with less data, though sometimes at the cost of a very large amount of compute - i.e. estimating high-dimensional integrals.

However, in general, these approaches don’t directly account for *symmetry* of the learned DAGs. Ideally, we would use a prior which weights causal DAGs according to the size of their representation - i.e. infinite DAGs would still have nonzero prior probability if they have some symmetry allowing for finite representation, and in general DAGs with multiple copies of the same sub-DAG would have higher probability. This isn’t quite the same as weighting by minimum description length in the Solomonoff sense, since we care specifically about symmetries which correspond to function calls - i.e. isomorphic subDAGs. We don’t care about graphs which can be generated by a short program but don’t have these sorts of symmetries. So that leaves the question: if our prior probability for a causal DAG is given by a notion of minimum description length which only allows compression by specifying re-used subcircuits, what properties will the resulting learning algorithm possess? Is it computable? What kinds of data are needed to make it tractable?

Intentional Bucket Errors

I want to illustrate a research technique that I use sometimes. (My actual motivation for writing this is to make it so that I don't feel as much like I need to defend myself when I use this technique.) I am calling it intentional bucket errors after a CFAR concept called bucket errors. Bucket errors is about noticing when multiple different concepts/questions are stored in your head as a single concept/question. Then, by noticing this, you can think about the different concepts/question separately.

What are Intentional Bucket Errors

Bucket errors are normally thought of as a bad thing. It has "errors" right in the name. However, I want to argue that bucket errors can sometimes be useful, and you might want to consider having some bucket errors on purpose. You can do this by taking multiple different concepts and just pretending that they are all the same. This usually only works if the concepts started out sufficiently close together.

Like many techniques that work by acting as though you believe something false, you should use this technique responsibly. The goal is to pretend that the concepts are the same to help you gain traction on thinking about them, but then to also be able to go back to inhabiting the world where they are actually different.

Why Use Intentional Bucket Errors

Why might you want to use intentional bucket errors? For one, maybe the concepts actually are the same, but they look different enough that you won't let yourself consider the possibility. I think this is especially likely to happen if the concepts are coming from very different fields or areas of your life. Sometimes it feels silly to draw strong connections between e.g. human rationality, AI alignment, evolution, economics, etc. but such connections can be useful.

Also I find this useful for gaining traction. There is something useful about constrained optimization for being able to start thinking about a problem. Sometimes it is harder to say something true and useful about X than it is to say something true and useful that simultaneously applies to X, Y, and Z. This is especially true when the concepts you are conflating are imagined solutions to problems.

For example, maybe I have an imagined solution to counterfactuals that has a hole in it that looks like understanding multi-level world models. Then, maybe I also have have an imagined solution to tiling that also has a hole in it that looks like understanding multi-level world models. I could view this as two separate problems. The desired properties of my MLWM theory for counterfactuals might be different from the desired properties for tiling. I have these two different holes I want to fill, and one strategy I have, which superficially looks like it makes the problem harder, is to try to find something that can fill both holes simultaneously. However, this can sometimes be easier because different use cases can help you triangulate the simple theory from which the specific solutions can be derived.

A lighter (maybe epistemically safer) version of intentional bucket errors is just to pay a bunch of attention to the connections between the concepts. This has its own advantages in that the relationships between the concepts might be interesting. However, I personally prefer to just throw them all in together, since this way I only have to work with one object, and it takes up fewer working memory slots while I'm thinking about it.

Examples

Here are a some recent examples where I feel like I have used something like this, to varying degrees.

How the MtG Color Wheel Explains AI Safety is obviously the product of conflating many things together without worrying too much about how all the clusters are wrong.

In How does Gradient Descent Interact with Goodhart, the question at the top about rocket designs and human approval is really very different from the experiments that I suggested, but I feel like learning about one might help my intuitions about the other. This was actually generated at the same time as I was thinking about Epistemic Tenure, which for me what partially about the expectation that there is good research and a correlated proxy of justifiable research, and even though our group idea selection mechanism is going to optimize for justifiable research, it is better if the inner optimization loops in the humans do not directly follow those incentives. The connection is a bit of a stretch in hindsight, but believing the connection was instrumental in giving me traction in thinking about all the problems.

Embedded Agency has a bunch of this, just because I was trying to factor a big problem into a small number of subfields, but the Robust Delegation section can sort of be described as "Tiling and Corrigibility kind of look similar if you squint. What happens when I just pretend they are two instantiations of the same problem?"

Embedded Naive Bayes

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

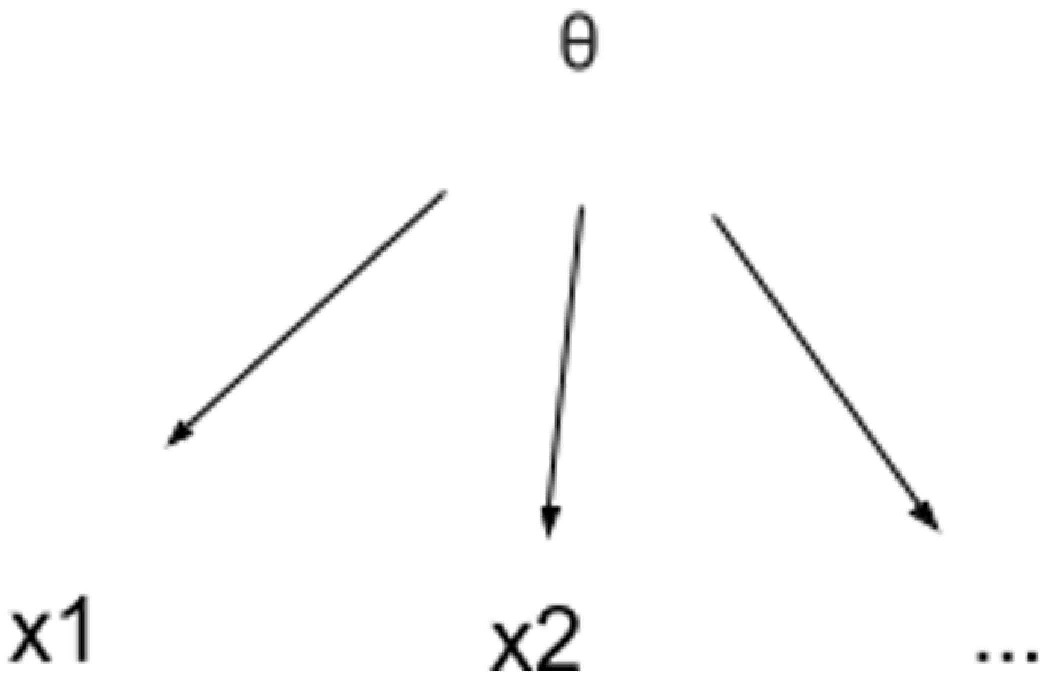
Suppose we have a bunch of earthquake sensors spread over an area. They are not perfectly reliable (in terms of either false positives or false negatives), but some are more reliable than others. How can we aggregate the sensor data to detect earthquakes?

A “naive” seismologist without any statistics background might try assigning different numerical scores to each sensor, roughly indicating how reliable their positive and negative results are, just based on the seismologist’s intuition. Sensor i gets a score s_i^+ when it’s going off, and s_i^- when it’s not. Then, the seismologist can add up the s_i^+ scores for all sensors going off at a given time, plus the s_i^- scores for sensors not going off, to get an aggregate “earthquake score”. Assuming the seismologist has decent intuitions for the sensors, this will probably work just fine.

It turns out that this procedure is equivalent to a Naive Bayes model.

Naive Bayes is a causal model in which there is some parameter θ in the environment which we want to know about - i.e. whether or not there’s an earthquake happening. We can’t observe θ directly, but we can measure it indirectly via some data $\{x_i\}$ - i.e. outputs from the earthquake sensors. The measurements may not be perfectly accurate, but their failures are at least independent - one sensor isn’t any more or less likely to be wrong when another sensor is wrong.

We can represent this picture with a causal diagram:



From the diagram, we can read off the model's equation: $P[\theta, \{x_i\}] = P[\theta] \prod_i P[x_i|\theta]$. We're interested mainly in the posterior probability $P[\theta|\{x_i\}] = \frac{1}{Z} P[\theta] \prod_i P[x_i|\theta]$ or, in log odds form,

$$L[\theta|\{x_i\}] = \ln \frac{P[\theta]}{P[\sim\theta]} + \sum_i \ln \frac{P[x_i|\theta]}{P[x_i|\sim\theta]}$$

Stare at that equation, and it's not hard to see how the seismologist's procedure turns into a Naive Bayes model: the seismologist's intuitive scores for each sensor correspond to the "evidence" from the sensor $\ln \frac{P[x_i|\theta]}{P[x_i|\sim\theta]}$. The "earthquake score" then corresponds to the posterior log odds of an earthquake. The seismologist has unwittingly adopted a statistical model. Note that this is still true regardless of whether the scores used are well-calibrated or whether the assumptions of the model hold - the seismologist is implicitly using this model, and whether the model is *correct* is an entirely separate question.

The Embedded Naive Bayes Equation

Let's formalize this a bit.

We have some system which takes in data x, computes some stuff, and spits out some f(x).

We want to know whether a Naive Bayes model is embedded in f(x). Conceptually, we imagine that f(x) parameterizes a probability distribution over some unobserved parameter θ - we'll write $P[\theta; f(x)]$, where the ";" is read as "parameterized by". For instance, we could imagine a normal distribution over θ, in which case f(x) might be the mean and variance (or any encoding thereof) computed from our input data. In our earthquake example, θ is a binary variable, so $f(x)$ is just some encoding of the probability that $\theta = True$.

Now let's write the actual equation defining an embedded Naive Bayes model. We assert that $P[\theta; f(x)]$ is the same as $P[\theta|x]$ under the model, i.e.

$$P[\theta; f(x)] = P[\theta|x] = \frac{1}{Z} P[\theta] \prod_i P[x_i|\theta]$$

We can transform to log odds form to get rid of the Z:

$$L[\theta; f(x)] = \ln \frac{P[\theta]}{P[\neg\theta]} + \sum_i \ln \frac{P[x_i|\theta]}{P[x_i|\neg\theta]}$$

Let's pause for a moment and go through that equation. We know the function $f(x)$, and we want the equation to hold for all values of x. θ is some hypothetical thing out in the environment - we don't know what it corresponds to, we just hypothesize that the system is modelling *something* it can't directly observe. As with x, we want the equation to hold for all values of θ. The unknowns in the equation are the probability functions $P[\theta; f(x)]$, $P[\theta]$ and $P[x_i|\theta]$. To make it clear what's going on, let's remove the probability notation for a moment, and just use functions G and $\{g_i\}$, with θ written as a subscript:

$$\forall \theta, x : G_\theta(f(x)) = c_\theta + \sum_i g_{\theta i}(x_i)$$

This is a functional equation: for each value of θ, we want to find functions G, $\{g_i\}$, and a constant c such that the equation holds for all possible x values. The solutions G and $\{g_i\}$ can then be decoded to give our probability functions $P[\theta; f(x)]$ and $P[x_i|\theta]$, while c can be decoded to give our prior $P[\theta]$. Each possible θ-value corresponds to a different set of solutions G_θ, $\{g_{\theta i}\}$, c_θ.

This particular functional equation is a variant of Pexider's equation; you can read all about it in Aczel's Functional Equations and Their Applications, chapter 3. For our purposes, the most important point is: depending on the function f, the equation may or may not have a solution. In other words, there is a meaningful sense in which some functions $f(x)$ *do* embed a Naive Bayes model, and others *do not*. Our seismologist's procedure *does* embed a Naive Bayes model: let G be the identity function, c be zero, and $g_i(x_i) = s_i^{x_i}$, and we have a solution to the embedding equation with $f(x)$ given by our seismologist's add-all-the-scores calculation (although this is not the *only* solution). On the other hand, a procedure computing

$f(x) = x_1^{x_2^{x_3}}$ for real-valued inputs x_1, x_2, x_3 would *not* embed a Naive Bayes model: with this $f(x)$, the embedding equation would not have any solutions.

Time Travel, AI and Transparent Newcomb

Epistemic status: has "time travel" in the title.

Let's suppose, for the duration of this post, that the local physics of our universe allows for time travel. The obvious question is: how are paradoxes prevented?

We may not have any idea how paradoxes are prevented, but presumably there must be *some* prevention mechanism. So, in a purely Bayesian sense, we can condition on paradoxes somehow not happening, and then ask what becomes more or less likely. In general, anything which would make a time machine more likely to be built should become less likely, and anything which would prevent a time machine being built should become more likely.

In other words: if we're trying to do something which would make time machines more likely to be built, this argument says that we should expect things to mysteriously go wrong.

For instance, let's say we're trying to build some kind of powerful optimization process which might find time machines instrumentally useful for some reason. To the extent that such a process is likely to build time machines and induce paradoxes, we would expect things to mysteriously go wrong when trying to build the optimizer in the first place.

On the flip side: we could commit to designing our powerful optimization process so that it not only avoids building time machines, but also actively prevents time machines from being built. Then the mysterious force should work in our favor: we would expect things to mysteriously go well. We don't need time-travel-prevention to be the optimization process' sole objective here, it just needs to make time machines sufficiently less likely to get an overall drop in the probability of paradox.

Logical Counterfactuals and Proposition graphs, Part 1

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Within this sequence of posts I will outline a procedure for logical counterfactuals based on something similar to proof length. In this post I present a reimagining of propositional logic in which proving a theorem is taking a walk around a graph of equivalent propositititions.

Within this post, I will use Latin symbols, for specific symbols within proof strings. I will sometimes use symbols like $\forall$ to represent a function with particular properties.

I will use Greek letters to represent an arbitrary symbol, upper case for single symbols, lower case for strings.

Respecifying Propositional logic

The goal of this first section is to reformulate first order logic in a way that makes logical counterfactuals easier. Lets start with propositional logic.

We have a set of primitive propositions $p, q, r, ...$ as well as the symbols $\top, \bot$. We also have the symbols $\vee, \wedge$ which are technically functions from $\text{Bool}^2 \to \text{Bool}$ but will be written $p \vee q$ not $\vee(p, q)$. There is also $\neg : \text{Bool} \to \text{Bool}$

Consider the equivalence rules.

1. $\alpha \equiv \neg\neg\alpha$

2. $\alpha \wedge \beta \equiv \beta \wedge \alpha$

3. $(\alpha \wedge \beta) \wedge \gamma \equiv \alpha \wedge (\beta \wedge \gamma)$

4. $\neg\alpha \wedge \neg\beta \equiv \neg(\alpha \vee \beta)$

5. $\alpha \wedge \top \equiv \alpha$

6. $\neg\alpha \vee \alpha \equiv \top$

7. $\bot \equiv \neg\top$

8. $\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$

9. $\bot \wedge \alpha \equiv \bot$

10. $\alpha \equiv \alpha \wedge \alpha$

Theorem

Any tautology provable in propositional logic can be created by starting at $\top$ and repeatedly applying equivalence rules.

Proof

First consider $\alpha \implies \beta$ to be shorthand for $\neg\alpha \vee \beta$.

Lemma

We can convert $\top$ into any of the 3 axioms.

$\alpha \implies (\beta \implies \alpha)$ is a shorthand for

$\neg\alpha \vee (\neg\beta \vee \alpha) \equiv_1$

$\neg\neg(\neg\alpha \vee (\neg\beta \vee \alpha)) \equiv_4$

$\neg(\neg\neg\alpha \wedge \neg(\neg\beta \vee \alpha)) \equiv_4$

$\neg(\neg\neg\alpha \wedge (\neg\neg\beta \wedge \neg\alpha)) \equiv_1$

$\neg(\neg\neg\alpha \wedge (\beta \wedge \neg\alpha)) \equiv_2$

$\neg(\neg\neg\alpha \wedge (\neg\alpha \wedge \beta)) \equiv_3$

$\neg((\neg\neg\alpha \wedge \neg\alpha) \wedge \beta) \equiv_4$

$\neg(\neg(\neg\alpha \vee \alpha) \wedge \beta) \equiv_6$

$\neg(\neg\top \wedge \beta) \equiv_7$

$\neg(\bot \wedge \beta) \equiv_9$

$\neg\bot \equiv_7$

$\top$

Similarly

$(\alpha \implies (\beta \implies \gamma)) \implies ((\alpha \implies \beta) \implies (\alpha \implies \gamma))$

$(\neg\alpha \implies \neg\beta) \implies (\beta \implies \alpha)$

(if these can't be proved, add that they $\equiv \top$ as axioms)

End Lemma

Whenever you have $\alpha \wedge (\alpha \implies \beta)$, that is equiv to

$\alpha \wedge (\neg\alpha \vee \beta) \equiv_8$

$(\alpha \wedge \neg\alpha) \vee (\alpha \wedge \beta) \equiv_1$

$(\neg\neg\alpha \wedge \neg\alpha) \vee (\alpha \wedge \beta) \equiv_4$

$\neg(\neg\alpha \vee \alpha) \vee (\alpha \wedge \beta) \equiv_6$

$\neg\top \vee (\alpha \wedge \beta) \equiv_1$

$\neg\neg(\neg\top \vee (\alpha \wedge \beta)) \equiv_4$

$\neg(\neg\neg\top \wedge \neg(\alpha \wedge \beta)) \equiv_1$

$\neg(\top \wedge \neg(\alpha \wedge \beta)) \equiv_2$

$\neg(\neg(\alpha \wedge \beta) \wedge \top) \equiv_5$

$\neg\neg(\alpha \land \beta) \equiv_1$

$\alpha \land \beta$

This means that you can create and apply axioms. For any tautology, look at the proof of it in standard propositional logic. Call the statements in this proof $p_1, p_2, p_3 \ldots$

suppose we have already found a sequence of substitutions from $\top$ to $p_1 \land p_2 \ldots \land p_{i-1}$

Whenever p_i is a new axiom, use (5.) to get $p_1 \land p_2 \ldots \land p_{i-1} \land \top$, then convert $\top$ into the instance of the axiom you want. (substitute alpha and beta with arbitrary props in above proof schema)

Using substitution rules (2.) and (3.) you can rearrange the terms representing lines in the proof and ignore their bracketing.

Whenever p_i is produced by modus ponus from the previous p_j and $p_k = p_j \implies p_i$ then duplicate p_k with rule (10.), move one copy next to p_j and use the previous procedure to turn $p_j \land (p_j \implies p_i)$ into $p_j \land p_i$. Then move p_i to end.

Once you reach the end of the proof, duplicate the result and unwind all the working back to $\top$, which can be removed by rule (5.)

Corollary

$\{p, q, r\} \vdash s$ then $p \land q \land r \equiv p \land q \land r \land s$

Because $p \land q \land r \implies s$ is a tautology and can be applied to get s.

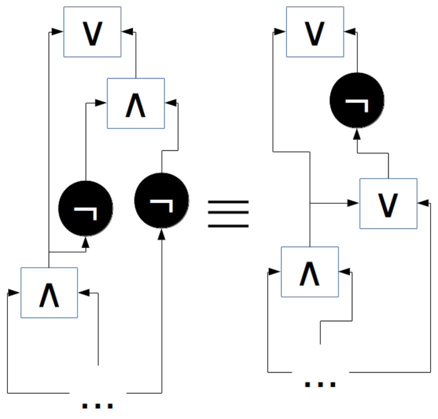
Corollary

Any contradiction is reachable from $\bot$

The negation of any contradiction k is a tautology.

$\bot \equiv \neg\top \equiv \neg\neg k \equiv k$

Intuitive overview perspective 1



An illustration of rule 4. ¬α ∧ ¬β ≡ ¬(α ∨ β) in action.

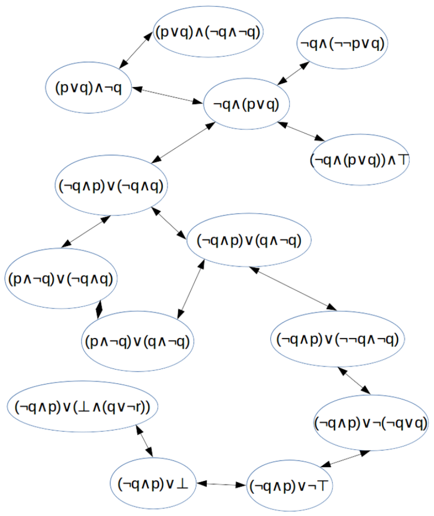
We can consider a proposition to be a tree with a root. The nodes are labeled with symbols. The axiomatic equivalences become local modifications to the tree structure, which are also capable of duplicating and merging identical subtrees by (10.). Arbitrary subtrees can be created or deleted by (5.).

We can merge nodes with identical subtrees into a single node. This produces a directed acyclic graph, as shown above. Under this interpretation, all we have to do is test node identity.

Intuitive overview perspective 2

Consider each possible expression to be a single node within an infinite graph.

Each axiomatic equivalence above describes an infinite set of edges. To get a single edge, substitute the generic $\alpha, \beta \ldots$ with a particular expression. For example, if you take (2. α ∧ β ≡ β ∧ α) and substitute α := p ∨ q and β := ¬q. We find a link between the node (p ∨ q) ∧ ¬q and ¬q ∧ (p ∨ q).



Here is a connected subsection of the graph. Note that, unlike the previous graph, this one is cyclic and edges are not directed.

All statements that are provably equivalent in propositional logic will be within the same connected component of the graph. All statements that can't be proved equivalent are in different components, with no path between them.

Finding a mathematical proof becomes an exercise in navigating an infinite maze.

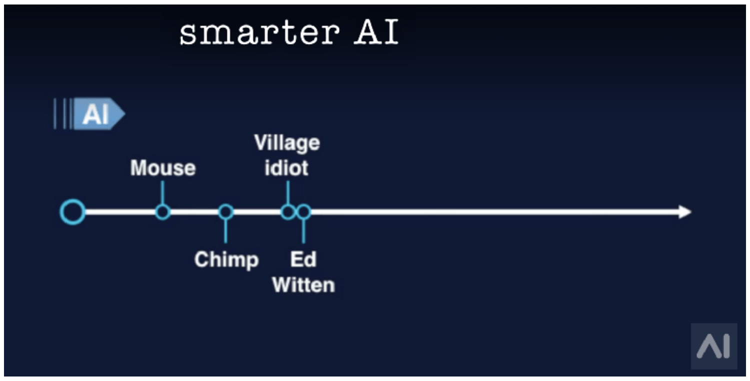
In the next Post

We will see how to extend the equivalence based proof system to an arbitrary first order theory. We will see what the connectedness does then. We might even get on to infinite dimensional vector spaces and why any of this relates to logical counterfactual.

Why so much variance in human intelligence?

Epistemic status: Practising thinking aloud. There might be an important question here, but I might be making a simple error.

There is a lot of variance in general competence between species. Here is the standard Bostrom/Yudkowsky graph to display this notion.



There's a sense that while some mice are more genetically fit than others, they're broadly all just mice, bound within a relatively narrow range of competence. Chimps should not be worried about most mice, in the short or long term, but they also shouldn't worry especially so about peak mice - there's no incredibly strong or cunning mouse they ought to look out for.

However, my intuition is very different for humans. While I understand that humans are all broadly similar, that a single human cannot have a complex adaptation that is not universal [1], I also have many beliefs that humans differ massively in cognitive capacities in ways that can lead to major disparities in general competence. The difference between someone who does understand calculus and someone who does not, is the difference between someone who can build a rocket and someone who cannot. And I think I've tried to teach people that kind of math, and sometimes succeeded, and sometimes failed to even teach basic fractions.

I can try to operationalise my hypothesis: if the average human intelligence was lowered to be equal to an IQ of 75 in present day society, that society could not have built rockets or do a lot of other engineering and science.

(Sidenote: I think the hope of iterated amplification is that this is false. That if I have enough humans with hard limits to how much thinking they can do, stacking lots of them can still produce all the intellectual progress we're going to need. My initial thought is that this doesn't make sense, because there are many intellectual feats like writing a book or coming up with special relativity that I generally expect individuals (situated within a conducive culture and institutions) to be much better at than groups of individuals (e.g. companies).

This is also my understanding of Eliezer's critique, that while it's possible to get humans with hard limits on cognition to make mathematical progress, it's by running an algorithm on them that they don't understand, not running an algorithm that they *do understand, and only if they understand it do you get nice properties about them being aligned in the same way you might feel many humans are today.*

It's likely I'm wrong about the motivation behind Iterated Amplification though.)

This hypothesis doesn't imply that someone who can do successful abstract reasoning is strictly more competent than a whole society of people who cannot. The Secret of our Success talks about how smart modern individuals stranded in forests fail to develop basic food preparation techniques that other, primitive cultures were able to build.

I'm saying that a culture with no people who can do calculus will in the long run score basically zero against the accomplishments of a culture with people who can.

One question is why we're in a culture so precariously balanced on this split between "can take off to the stars" and "mostly cannot". An idea I've heard is that if a culture is easily able to reach technologically maturity, it will come later than a culture who is barely able to become technologically maturity, because evolution works over much longer time scales than culture + technological innovation. As such, if you observe yourself to be in a culture that is able to reach technologically maturity, you're probably "the stupidest such culture that could get there, because if it could be done at a stupider level then it would've happened there first."

As such, we're a species whereby if we try as hard as we can, if we take brains optimised for social coordination and make them do math, then we can just about reach technical maturity (i.e. build nanotech, AI, etc).

That may be true, but the question I want to ask about is what is it about humans, culture and brains that allows for such high variance within the species, that isn't true about mice and chimps? Something about this is still confusing to me. Like, if it is the case that some humans are able to do great feats of engineering like build rockets that land, and some aren't, what's the difference between these humans that causes such massive changes in outcome? Because, as above, it's not some big complex genetic adaptation some have and some don't. I think we're all running pretty similar genetic code.

Is there some simple amount of working memory that's required to do complex recursion? Like, 6 working memory slots makes things way harder than 7?

I can imagine that there are many hacks, and not a single thing. I'm reminded of the story of Richard Feynman learning to count time, where he'd practice being able to count a whole minute. He'd do it while doing the laundry, while cooking breakfast, and so on. He later met the mathematician John Tukey, who could do the same, but they had some fierce disagreements. Tukey said you couldn't do it while reading the newspaper, and Feynman said he could. Feynman said you couldn't do it while having a conversation, and Tukey said they could. They then both surprised each other by doing exactly what they said they could.

It turned out Feynman was *hearing* numbers being spoken, whereas Tukey was *visualising* the numbers ticking over. So Feynman could still read at the same time, and his friend could still listen and talk.

The idea here is that if you're unable to use one type of cognitive resource, you may make up for it with another. This is probably the same situation as when you make trade-offs between space and time in computational complexity.

So I can imagine different humans finding different hacky ways to build up the skill to do very abstract truth-tracking thinking. Perhaps you have a little less working memory than average, but you have a great capacity for visualisation, and primarily work in areas that lend themselves to geometric / spacial thinking. Or perhaps your culture can be very conducive to abstract thought in some way.

But even if this is right I'm interested in the details of what the key variables actually are.

What are your thoughts?

---

[1] Note: humans can *lack* important pieces of machinery.

Towards a mechanistic understanding of corrigibility

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Acceptability

To be able to use something like relaxed adversarial training to verify a model, a necessary condition is having a good notion of *acceptability.* Paul Christiano describes the following two desiderata for any notion of acceptability:

1. "As long as the model *always* behaves acceptably, and achieves a high reward *on average,* we can be happy."
2. "Requiring a model to always behave acceptably wouldn't make a hard problem too much harder."

While these are good conditions that any notion of acceptability must satisfy, there may be many different possible acceptability predicates that meet both of these conditions—how do we distinguish between them? Two additional major conditions that I use for evaluating different acceptability criteria are as follows:

1. It must be not that hard for an amplified overseer to verify that a model is acceptable.
2. It must be not that hard to find such an acceptable model during training.

These conditions are different than Paul's second condition in that they are statements about the ease of training an acceptable model rather than the ease of choosing an acceptable action. If you want to be able to do some form of informed oversight to produce an acceptable model, however, these are some of the most important conditions to pay attention to. Thus, I generally think about choosing an acceptability condition as trying to answer the question: *what is the easiest-to-train-and-verify property such that all models that satisfy that property*[1] *(and achieve high average reward) are safe?*

Act-Based Corrigibility

One possible candidate property that Paul has proposed is act-based corrigibility, wherein an agent respects our short-term preferences, including those over how the agent itself should be modified. Not only is such an agent corrigible, Paul argues, but it will also want to make itself more corrigible, since having it be more corrigible is a component of our short-term preferences (Paul calls this the "broad basin" of corrigibility). While such act-based corrigibility would definitely be a nice property to have, it's unclear how exactly an amplified overseer could go about verifying such a property. In particular, if we want to verify such a property, we need a *mechanistic* understanding of act-based corrigibility rather than a *behavioral* one, since behavioral properties can only be verified by testing every input, whereas mechanistic properties can be verified just by inspecting the model.

One possible mechanistic understanding of corrigibility is *corrigible alignment* as described in "Risks from Learned Optimization," which is defined as the situation in which "the base objective is incorporated into the mesa-optimizer's epistemic model and [the mesa-optimizer's] objective is modified to 'point to' that information." While this gives us a starting point for understanding what a corrigible model might actually look like, there are still a bunch of missing pieces that have to be filled in. Furthermore, this notion of corrigibility looks more like instrumental corrigibility rather than act-based corrigibility, which as Paul notes is significantly less likely to be robust. Mechanistically, we can think of this lack of robustness as coming from the fact that "pointing" to the base objective is a pretty unstable operation: if you point even a little bit incorrectly, you'll end up with some sort of corrigible pseudo-alignment rather than corrigible robust alignment.

We can make this model more act-based, and at least somewhat mitigate this robustness problem, however, if we imagine pointing to only the human's short-term preferences. The hope for this sort of a setup is that, as long as the initial pointer is "good enough," there will be pressure for the mesa-optimizer to make its pointer better in the way in which its current understanding of short-term human preferences recommends, which is exactly Paul's "broad basin" of corrigibility argument. This requires it to be not that hard, however, to find a model with a notion of the human's short-term preferences as opposed to their long-term preferences that is also willing to correct that notion based on feedback.

In particular, it needs to be the case that it is not that hard to find an agent which will correct mistakes in its own prior over what the human's short-term preferences are. From a naive Bayesian perspective, this seems unlikely, as it seems strange for an agent to be incentivized to change its own prior. However, this is actually a very natural state for an agent to be in: if I trust your beliefs about X more than I trust my own, then that means I would endorse a modification of my prior to match yours. In the context of act-based corrigibility, we can think about this from a mechanistic perspective as having a *pre-prior* that encodes a belief that the human prior over human short-term preferences is to be preferred. Furthermore, pre-priors are generally epistemically valuable for agents to have, as a pre-prior can encourage an agent to correct its own cognitive biases. Thus, agents with pre-priors should be incentivized by most training processes, and thus shouldn't be too difficult to find.

Indifference Corrigibility

Instrumental and act-based corrigibility are not the only forms of corrigibility that have been discussed in the literature, however: there's also *indifference corrigibility*, wherein the agent is indifferent to modifications the human might make to it such as shutting it off. While this form of corrigibility doesn't in and of itself guarantee acceptability, if you had a way of verifying indifference corrigibility, there might be ways to turn that back into act-based corrigibility.

First, however, if we want to be able to verify indifference corrigibility, we are going to need a mechanistic understanding of it. Let's suppose we accept the argument that a competitive model will likely be a mesa-optimizer such that it will be running some sort of optimization process coupled with some sort of mesa-objective. Furthermore, let's suppose that its optimization process is computed over some sort of world model, which I think is a reasonable assumption for any competitive mesa-optimizer that wants to be able to take actions in the world. In such a case, we can think of indifference corrigibility as a form of stop gradient in the mesa-optimizer's world model which its optimization process doesn't optimize through,[2] which is a highly mechanistic property that we might expect an amplified overseer to be able to verify.

Again, however, we need agents with stop gradients to form a natural enough class such that they are not that hard to find. While it might naively seem as if agents with stop gradients would be very rare, in fact the exact opposite might be the case. Consider an agent such as the mesa-optimizer described previously which is running an optimization process over some world model, but which has preferences over the state of the world rather than the state of its world model. Such an agent will have an incentive to keep its world model aligned with the world, but only in one direction: it will want to modify its world model to match the world, but not modify the world to match its world model. This is a type of stop gradient, as it is a restriction on the agent to only minimize the difference between its world model and the world through one pathway but not through another. This example seems to suggest that stop gradients aren't that uncommon among agents such that it might be fairly achievable to find such an agent.

Even if we can train and verify indifference corrigibility, however, one of the big remaining problems with such an approach is that an indifference corrigible mesa-optimizer might do search in such a way that it finds a policy which is not itself indifference corrigible. We can think about this problem as arising from the fact that indifference corrigibility doesn't tile, though I more often think of this problem as the problem of "forwarding the guarantee," wherein even if we know how to do safe search on the outside, we have to make sure that any search inside of our model is similarly safe as well. This is a particularly interesting type of inner alignment problem in that, rather than being about how to ensure that a mesa-optimizer's objective is aligned, it is about how to ensure that a mesa-optimizer's search is safe even given that its objective is aligned. However, it seems plausible that this sort of problem could be resolved by ensuring that the model has a meta-preference towards any policies it produces also respecting the same stop gradient. In particular, the overseer could verify that any search over policies done by the model enforce the constraint that every policy have such a stop gradient.

Even once we can verify that our model is indifference corrigible and that it will forward that guarantee to any other search it might perform, however, there is still the question of how we might be able to use such a mechanism to produce a safe system. One way in which indifference corrigibility could be used to produce safety is to enforce that our model behave *myopically.* We can think of a myopic agent as one that only considers how best to answer the single question that you give to it rather than considering any sort of long-term consequences, which can be implemented as a stop gradient preventing any optimization outside of that domain. While myopia on its own is still insufficient to guarantee acceptability, it seems like it would at least prevent *deceptive alignment*, as one of the conditions for deceptive alignment is that the mesa-optimizer must have something to gain from cooperating now and then defecting later, which is not true for a myopic agent. Thus, if directed at a task which we are confident is outer aligned, such as pure supervised amplification (training a model to approximate a human consulting that model), and combined with a scheme for preventing standard pseudo-alignment (such as adversarial training), myopia verification might be sufficient to resolve the rest of the inner alignment problem by preventing deceptive alignment.

Conclusion

If we want to be able to do relaxed adversarial training to produce safe AI systems, we are going to need a notion of acceptability which is not that hard to train and verify. Corrigibility seems to be one of the most promising candidates for such an acceptability condition, but for that to work we need a mechanistic understanding of exactly what sort of corrigibility we're shooting for and how it will ensure safety. I think that both of the paths considered here—both act-based corrigibility and indifference corrigibility—look like promising research directions for attacking this problem.

---

1. Or at least all models that we can *find* that satisfy that property. ↩
2. Thanks to Scott Garrabrant for the stop gradient analogy. ↩

Logical Optimizers

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

EDIT: This idea is unsafe unless you have X and only X optimization. If an early logical optimiser produces a paperclip maximiser, then the paperclip maximiser will play along giving good solutions to most logical optimisation problems, but hiding nasty surprises where we are likely to find them. In other words, you have risks of algorithms that give outputs that are pretty well logically optimized, but are also optimized to harm you somehow. Even if the first algorithm isn't like this, it could stumble upon an algorithm like this in the self improvement stage.

END EDIT

Epistemic status: I think the basic Idea is more likely than not sound. Probably some mistakes. Looking for sanity check.

Black box description

The following is a way to Foom an AI while leaving its utility function and decision theory as blank spaces. You could plug any uncomputable or computationally intractable behavior you might want in, and get an approximation out.

Suppose I was handed a hypercomputer and allowed to run code on it without worrying about mindcrime, then the hypercomputer is removed, allowing me to keep 1Gb of data from the computations. Then I am handed a magic human utility function, as code on a memory stick. This approach would allow me to use the situation to make a FAI.

Example algorithms

Suppose you have a finite set of logical formulas, each of which evaluate to some real number. A logical optimizer is an algorithm that takes those formulas and tries to maximize the value of the formula it outputs.

One such algorithm would be to run a proof search for theorems of the form $r_n > q$ where r_n is one of the formulas representing a real number, and q is an explicitly written rational. After running the proof search for a finite time, you will have finitely many formulas of this form. Pick the one with the largest q in it, and return the r_n that is provably bigger than it.

Another algorithm to pick a large r_n is to run a logical inductor to estimate each $r_n$and then pick the $r_n$that maximized those estimates.

Suppose the formulas were

1) "3+4"

2) "10 if P=NP else 1"

3) "0 if P=NP else 11"

4) "2*6-3"

When run with a small amount of compute, these algorithms would pick option (4). They are in a state of logical uncertainty about whether P=NP, and act accordingly.

Given vast amounts of compute, they would pick either 2 or 3.

We might choose to implicitly represent a set of propositions in some manner instead of explicitly stating them. This would mean that a Logical Optimizer could optimize without needing to explicitly consider every possible expression. It could use an evolutionary algorithm. It could rule out swaths of propositions based on abstract reasoning.

Self Improvement

Now consider some formally specifiable prior over sets of propositions called P. P could be a straightforward simplicity based prior, or it could be tuned to focus on propositions of interest.

Suppose $\alpha_1, ..., \alpha_n$ are a finite set of programs that take in a set of propositions $R = \{r_1, ..., r_m\}$ and output one of them. If a program fails to choose a number from 1 to m quickly enough then pick randomly, or 1 or something.

Let $C(\alpha, R) = r_i$ be the choice made by the program α.

Let $S(\alpha) = \sum_{R \in P} C(\alpha, R) \times P_R$ be the average value of the proposition chosen by the program α, weighted by the prior over sets of propositions P.

Now attempting to maximize S(α) over all short programs α is something that Logical Optimizers are capable of doing. Logical Optimizers are capable of producing other, perhaps more efficient Logical Optimizers in finite time.

Odds and ends

Assuming that you can design a reasonably efficient Logical Optimizer to get things started, and that you can choose a sensible P, you could get a FOOM towards a Logical Optimizer of almost maximal efficiency.

Note that Logical Optimizers aren't AI's. They have no concept of empirical uncertenty about an external world. They do not perform Baysian updates. They barely have a utility function. You can't put one in a prisoners dilemma. They only resolve a certain kind of logical uncertainty.

On the other hand, a Logical Optimizer can easily be converted into an AI by defining a prior, a notion of baysian updating, an action space and a utility function.

Just maximize over action $a \in A$ in expressions of the form "Starting with Prior P and

updating it based on evidence E, if you take action a then your utility will be?"

I suspect that Logical Optimisers are safe, in the sense that you could get one to FOOM on real world hardware, without holomorphic encryption and without disaster.

Logical Optimizers are not Clever fool proof. A clever fool could easily turn one into a paper clip maximizer. Do not put a FOOMed one online.

I suspect that one sensible route to FAI is to FOOM a logical optimizer, and then plug in some uncomputable or otherwise unfeasible definition of friendliness.

Deconfuse Yourself about Agency

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

This post is a result of numerous discussions with other participants and organizers of the MIRI Summer Fellows Program 2019.

I recently (hopefully) dissolved some of my confusion about agency. In the first part of the post, I describe a concept that I believe to be central to most debates around agency. I then briefly list some questions and observations that remain interesting to me. *The gist of the post should make sense without reading any of the math.*

Antropomorphization, but with architectures that aren't humans

Architectures

Consider the following examples of "architectures":

Example (architectures)

1. Architectures I would *intuitively* call "agenty":
   1. Monte Carlo tree search algorithm, parametrized by the number of rollouts made each move and utility function (or heuristic) used to evaluate positions.
   2. (semi-vague) "Classical AI-agent" with several interconnected modules (utility function and world model, actions, planning algorithm, and observations used for learning and updating the world model).
   3. (vague) Human parametrized by their goals, knowledge, and skills (and, of course, many other details).
2. Architectures I would *intuitively* call "non-agenty":
   1. A hard-coded sequence of actions.
   2. Look-up table.
   3. Random generator (outputting $x \sim \pi$ on every input, for some probability distribution π).
3. Multi-agent architectures[1]:
   1. Ant colony.
   2. Company (consisting of individual employees, operating within an economy).
   3. Comprehensive AI services.

Working definition: *Architecture* $A(\Theta)$ is some model parametrizable by $\theta \in \Theta$ that receives inputs, produces outputs, and possibly keeps an internal state. We denote specific instances of $A(\Theta)$ as $A(\theta)$.

Generalizing anthropomorphization

Throughout the post, X will refer to some object, procces, entity, etc., whose behavior we want to predict or understand. Examples include rocks, wind, animals, humans, AGIs, economies, families, or the universe.

A standard item in the human mental toolbox is anthropomorphization: modeling various things as humans (specifically, ourselves) with "funny" goals or abilities. We can make the same mental move for architectures other than humans:

Working definition ($A(\Theta)$-morphization): Let $A(\Theta)$ be an architecture. Then any[2] model $A(\theta)$ is an $A(\Theta)$-*morphization* of X.

Antropomorphization makes good predictions for other humans and some animals (curiosity, fear, hunger). On the other hand, it doesn't work so well for rocks, lightning, and AGI-s --- not that it would prevent us from using it anyway. We can measure the usefulness of $A(\Theta)$-morphization by the degree to which it makes good predictions:

Working definition (prediction error): Suppose X exists in a world W and $\vec{E} = (E_1, \dots, E_n)$ is a sequence of variables (events about X) that we want to predict. Suppose that $\vec{e} = (e_1, \dots, e_n)$ is how $\vec{E}$ actually unfolds and $\vec{\pi} = (\pi_1, \dots, \pi_n)$ is the prediction obtained by $A(\Theta)$-morphizing X as $A(\theta)$. The *prediction error* of $A(\theta)$ (w.r.t. X and $\vec{E}$ in W) is the expected Brier score of $\vec{\pi}$ with respect to $\vec{e}$.

Informally, we say that $A(\Theta)$-morphizing X is accurate if the corresponding prediction error is low.[3]

When do we call things agents?

Main claim:

1. I claim that the question "Is X an agent?" is without substance, and we should instead be asking "From the point of view of some external observer Y, does X seem to exhibit agent-like behavior?".
2. Moreover, "agent-like behavior" also seems ill-defined, because what we associate with "agency" is subjective. I propose to *explicitly* operationalize the

question as "Is A(Θ)-morphizing X accurate?".

(A related question is how difficult is it for us to "run" A(θ). Indeed, we anthropomorphize so many things precisely because it is cheap for us to do so.)

Relatedly, I believe we already implicitly do this operationalization: Suppose you talk to your favorite human H about agency. H will likely subconsciously associate agency with certain architectures, maybe such as those in Example 1.1-3. Moreover, H will ascribe varying degrees of agency to different architectures --- for me, 1.3 seems more agenty than 1.1. Similarly, there are some architectures that H will associate with "definitely not an agent". I conjecture that some X *exhibits agent-like behavior according to* H if it can be accurately predicted via A(Θ)-morphization for some agenty-to-H architecture A(Θ). Similarly, H *would say that* X *exhibits non-agenty behavior* if we can accurately predict it using some non-agenty-to-H architecture.

Critically, exhibiting agent-like-to-H **and non-agenty-to-**H **behavior is not mutually exclusive**, and I think this causes most of the confusion around agency. Indeed, we humans seem very agenty but, at the same time, determinism implies that there exists some hard-coded behavior that we enact. A rock rolling downhill can be viewed as merely obeying the non-agenty laws of physics, but what if it "wants to" get as low as possible? And, as a result, we sometimes go "Humans are definitely agents, and rocks are definitely non-agents...although, wait, are they?".

If we ban the concept of agency, which interesting problems remain?

"Agency" often comes up when discussing various alignment-related topics, such as the following:

Optimizer?

How do we detect whether X performs (or capable of performing) optimization? How to detect this from X's architecture (or causal origin) rather than looking at its behavior? (This seems central to the topic of mesa-optimization.)

Agent-like behavior vs agent-like architecture

Consider the following conjecture: "Suppose some X exhibits agent-like behavior. Does it follow that X physically contains agent-like architecture, such as the one from Example 1.2?". This conjecture is false --- as an example, Q-learning is a "fairly agenty" architecture that leads to intelligent behavior. However, the resulting RL "agent" has a fixed policy and thus functions as a large look-up table. A better question would thus be whether there exist an agent-like architecture *causally upstream* of X. This question also has a negative answer, as witnessed by the example of an ant colony --- agent-like behavior without agent-like architecture, produced by a "non-agenty" optimization process of evolution. Nonetheless, a general version of the question remains: If some X exhibits agent-like behavior, does it follow that there exists some interesting physical structure[4] causally upstream of X?[5]

Moral standing

Suppose there is some X, which I model as having some goals. When making actions should I give weight to those goals? (The answer to this question seems more related to conciousness than to A(Θ)-morphization. Note also that a particularly interesting version of the question can be obtained by replacing "I" by "AGI"...)

PC or NPC?

When making plans, should we model X as a part of the environment, or does it enter our game-theoretical considerations? Is X able to model us?

Creativity, unbounded goals, environment-generality

In some sense, AlphaZero is an extremely capable game-playing agent. On the other hand, if we gave it access to the internet[6], it wouldn't do anything with it. The same cannot be said for humans and unaligned AGIs, who would not only be able to orient in this new environment but would eagerly execute elaborate plans to increase their influence. How can we tell whether some X is more like the former or the latter?

To summarize, I believe that many arguments and confusions surrounding agency can disappear if we explicitly use A(Θ)-morphization. This should allow us to focus on the problems listed above. Most definitions I gave are either semi-formal or informal, but I believe they could be made fully formal in more specific cases.

Regarding feedback: Sugestions for a better name for "A(Θ)-morphization" super-welcome! If you know of an application for which such formalization would be useful, please do let me know. Pointing out places where you expect a useful formalization to be impossible is also welcome.

---

1. You might also view multi-agent systems these as monolithic agents, but this view might often give you wrong intuitions. I am including this category as an example that -- intuitively -- doesn't belong to either of the "agent" and "not-agent" categories. ↩

2. By default, we do not assume that A(Θ)-morphization of X is useful in any way, or even the most useful among all instances of A(Θ). This goes against the intuition according to which we would pick some A(θ) that is close to optimal (among $\theta' \in \Theta$) for predicting X. I am currently unsure how to formalize this intuition, apart from requiring that is optimal (which seems too strong a condition). ↩

3. Distinguishing between "small enough" and "too big" prediction errors seems non-trivial since different environments are naturally more difficult to predict than others. Formalizing this will likely require additional insights. ↩

4. An example of such "interesting physical structure" would be an implementation of an optimization architecture. ↩

5. Even if true, this conjecture will likely require some additional assumptions. Moreover, I expect "randomly-generated look-up tables that happen to stumble upon AGI by chance" to serve as a particularly relevant counterexample. ↩

6. Whatever that means in this case. ↩

Thoughts from a Two Boxer

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

I'm writing this for blog day at MSFP. I thought about a lot of things here like category theory, the 1-2-3 conjecture and Paul Christiano's agenda. I want to start by thanking everyone for having me and saying I had a really good time. At this point I intend to go back to thinking about the stuff I was thinking about before MSFP (random matrix theory). But I learned a lot and I'm sure some of it will come to be useful. This blog is about (my confusion of) decision theory.

Before the workshop I hadn't read much besides Eliezer's paper on FDT and my impression was that it was mostly a good way of thinking about making decisions and at least represented progress over EDT and CDT. After thinking more carefully about some canonical thought experiments I'm no longer sure. I suspect many of the concrete thoughts which follow will be wrong in ways that illustrate very bad intuitions. In particular I think I am implicitly guided by non-example number 5 of an aim of decision theory in Wei Dai's post on the purposes of decision theory. I welcome any corrections or insights in the comments.

The Problem of Decision Theory

First I'll talk about what I think decision theory is trying to solve. Basically I think decision theory is the theory of how one should[1] decide on an action after one already understands: The actions available, the possible outcomes of actions, the probabilities of those outcomes and the desirability of those outcomes. In particular the answers to the listed questions are only adjacent to decision theory. I sort of think answering all of those questions is in fact harder than the question posed by decision theory. Before doing any reading I would have naively expected that the problem of decision theory, as stated here, was trivial but after pulling on some edge cases I see there is room for a lot of creative and reasonable disagreement.

A lot of the actual work in decision theory is the construction of scenarios in which ideal behavior is debatable or unclear. People choose their own philosophical positions on what is rational in these hairy situations and then construct general procedures for making decisions which they believe behave rationally in a wide class of problems. These constructions are a concrete version of formulating properties one would expect an ideal decision theory to have.

One such property is that an ideal decision theory shouldn't choose to self modify in some wide vaguely defined class of "fair" problems. An obviously unfair problem would be one in which the overseer gives CDT $10 and any other agent $0. One of my biggest open questions in decision theory is where this line between fair and unfair problems should lie. At this point I am not convinced any problem where agents in the environment have access to our decision theory's source code or copies of our agent are fair problems. But my impression from hearing and reading what people talk about is that this is a heretical position.

Newcomb's Problem

Let's discuss Newcomb's problem in detail. In this problem there are two boxes one of which you know contains a dollar. In the other box an entity predicting your action may or may not put a million dollars. They put a million dollars if and only if they predict you will only take one box. What do you do if the predictor is 99 percent accurate? How about if it is perfectly accurate? What if you can see the content of the boxes before you make your decision?

An aside on why Newcomb's problem seems important: It is sort of like a prisoner's dilemma. To see the analogy imagine you're playing a classical prisoner's dilemma against a player who can reliably predict your action and then chooses to match it. Newcomb's problem seems important because prisoner's dilemmas seem like simplifications of situations which really do occur in real life. The tragedy of prisoner dilemmas is that game theory suggests you should defect but the real world seems like it would be better if people cooperated.

Newcomb's problem is weird to think about because the predictor and agent's behaviors are logically connected but not causally. That is, if you tell me what the agent does or what the predictor predicts as an outside observer I can guess what the other does with high probability. But once the predictor predicts the agent could still take either option and flip flopping won't flip flop the predictor. Still one may argue you should one box because being a one boxer going into the problem means you will likely get more utility. I disagree with this view and see Newcomb's problem as punishing rational agents.

If Newcomb's problem is ubiquitous and one imagines an agent walking down the street constantly being Newcombed it is indeed unfortunate if they are doomed to two box. They'll end up with far fewer dollars. But this thought experiment is missing an important part of real world detail in my view. How the predictors predict the agents behavior. There are three possibilities:

- The predictors have a sophisticated understanding of the agent's inner workings and use it to simulate the agent to high fidelity.
- The predictors have seen many agents like our agent doing problems like this problem and use this to compute a probability of our agent's choice and compare it to a decision threshold.
- The predictor has been following the behavior of our agent and uses this history to assign its future behavior a probability.

In the third bullet the agent should one box if they predict they are likely to be Newcombed often[2]. In the second bullet they should one box if they predict that members of their population will be Newcombed often and they derive more utility from the extra dollars their population will get than the extra dollar they could get for themselves. I have already stated I see the first bullet as an unfair problem.

Mind Reading isn't Cool

My big complaint with mind reading is that there just isn't any mind reading. All my understandings of how people behave comes from observing how they behave in general, how the human I'm trying to understand behaves specifically, whatever they have explicitly told me about their intentions and whatever self knowledge I have I believe is applicable to all humans. Nowhere in the current world do people have to make decisions under the condition of being accurately simulated.

Why then do people develop so much decision theory intended to be robust in the presence of external simulators? I suppose its because there's an expectation that this will be a major problem in the future which should be solved philosophically before it is practically important. Mind reading could become important to humans if mind surveillance because possible and deployed. I don't think such a thing is possible in the near term or likely even in the fullness of time. But I also can't think of any insurmountable physical obstructions so maybe I'm too optimistic.

Mind reading is relevant to AI safety because whatever AGI is created will likely be a program on a computer somewhere which could reason its program stack is fully transparent or its creators are holding copies of it for predictions.

Conclusion

Having written that last paragraph I suddenly understand why decision theory in the AI community is the way it is. I guess I wasn't properly engaging with the premises of the thought experiment. If one actually did tell me I was about to do a Newcomb experiment I would still two box because knowing I was in the real world I wouldn't really believe that an accurate predictor would be deployed against me. But an AI can be practically simulated and what's more can reason that it is just a program run by a creator that could have created many copies of it.

I'm going to post this anyway since it's blog-day and not important-quality-writing day but I'm not sure this blog has much of a purpose anymore.

---

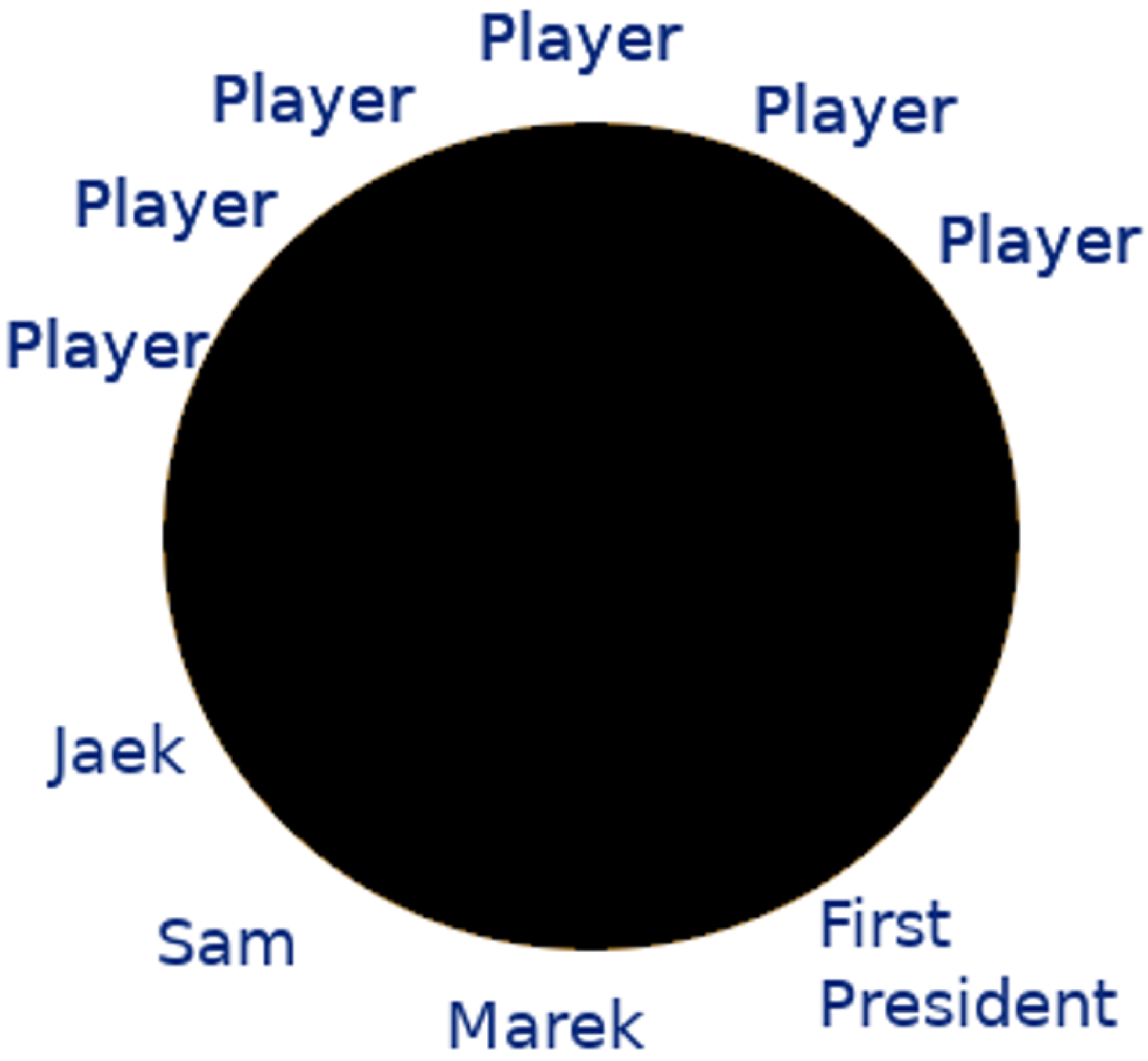
1. This may read like I'm already explicitly guided by the false purpose Wei Dai warned against. My understanding is that the goal is to understand ideal decision making. Just not for the purposes of implementation. ↩

2. I don't really know anything but I imagine the game theory of reputation is well developed ↩

Analysis of a Secret Hitler Scenario

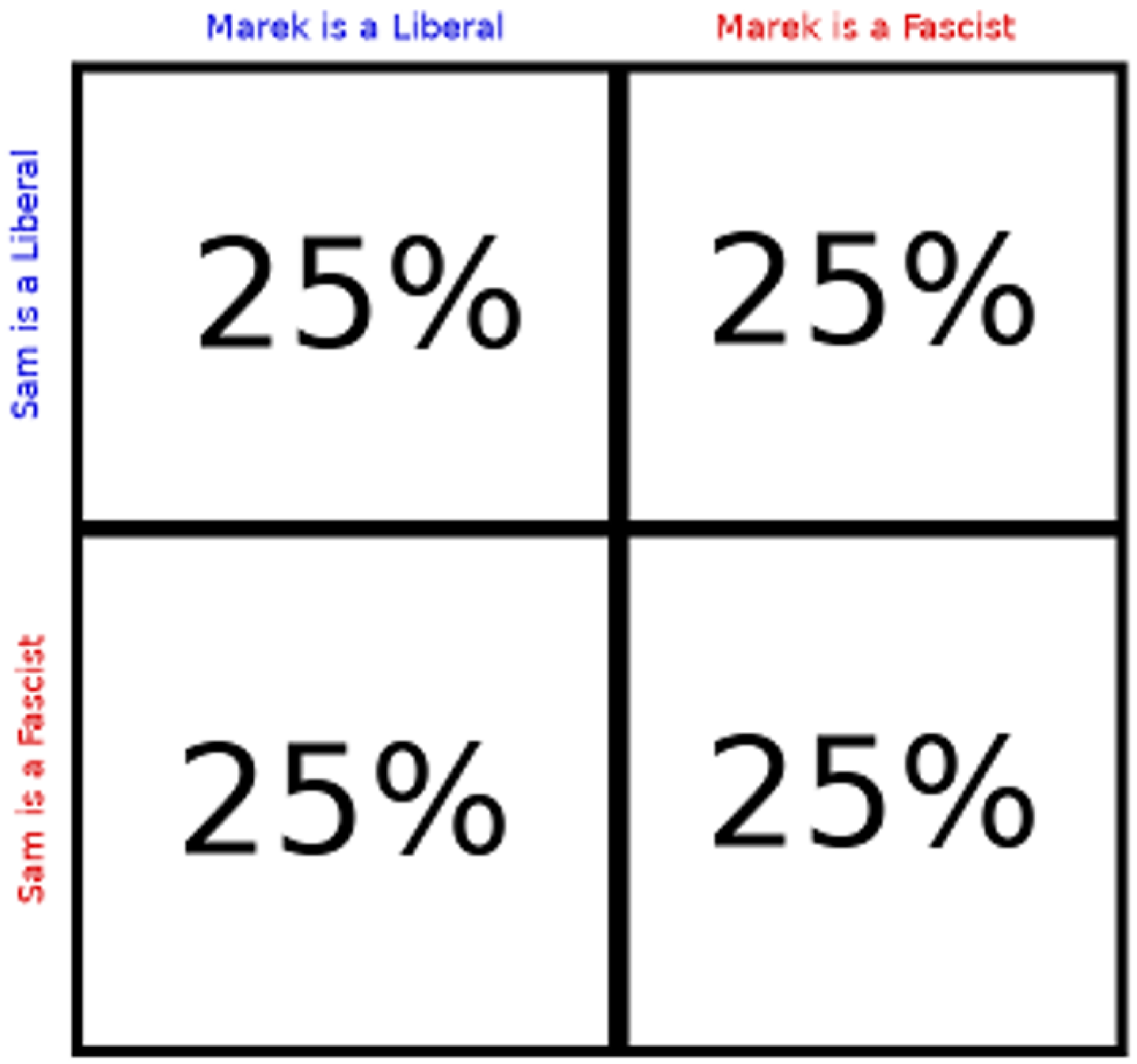
Secret Hitler is a social deception game in the tradition of mafia, the resistance and Avalon [1]. You can read the rules here if you aren't familiar. I haven't played social deception games regularly since 2016 but in my mind it's a really good game that represented the state of the art in the genre at that time. I'm going to discuss an interesting situation in which I reasoned poorly.

I was a liberal in a ten player game. The initial table set up, displaying relevant players was,



We passed a fascist article in the first round. The next government was Marek as president and Chancellor a player 3 to the right of Marek [2]. Marek passed a fascist article and inspected Sam and declared Sam was fascist to Sam's counterclaim that Marek was fascist. Sam was to the left of Marek so we had no data about him. The table generally supported Sam but I leaned towards believing Marek.

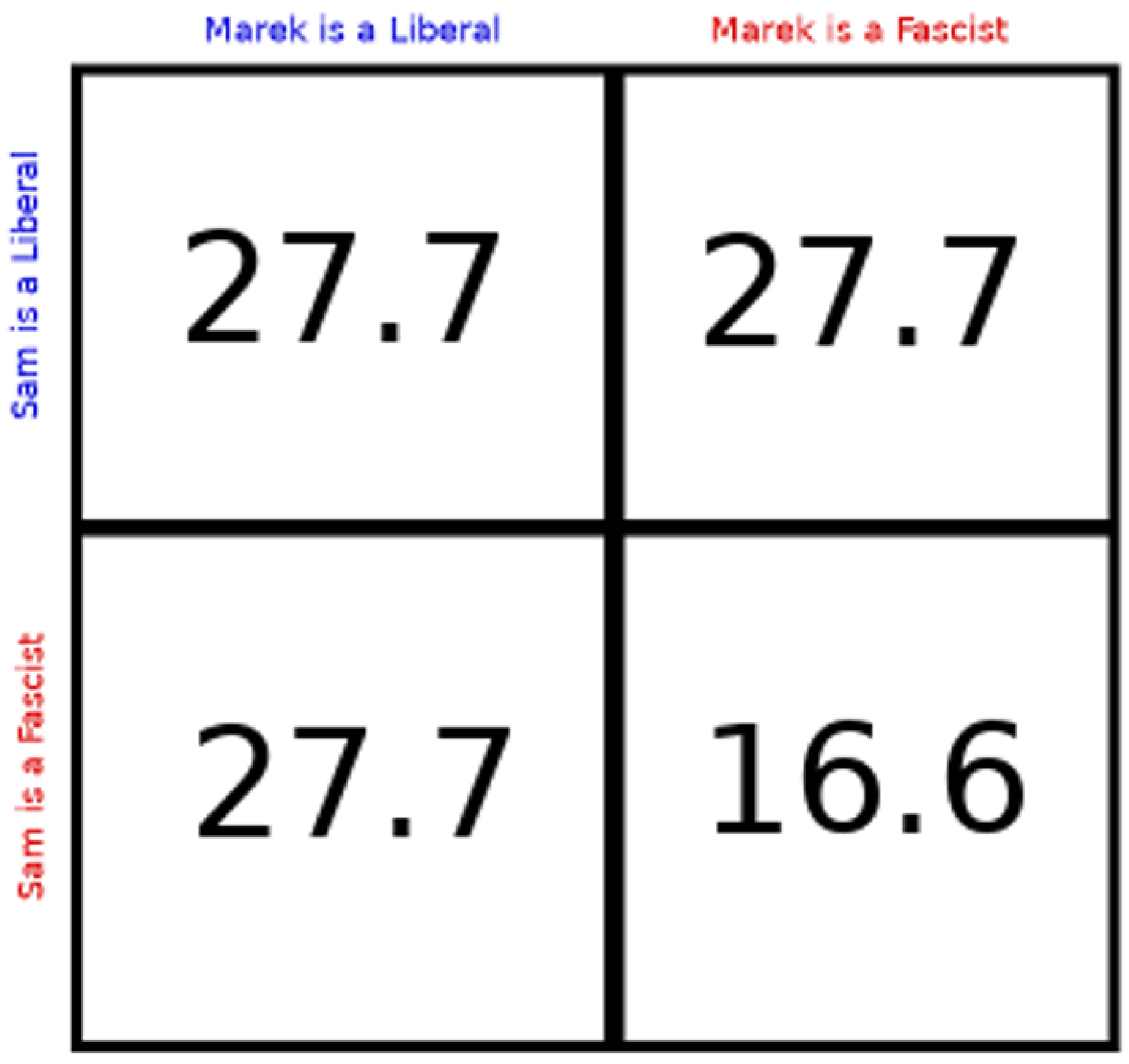
At the beginning of the game my view of possibilities looked roughly like:



But of course I'm already approximating. From my perspective an individual is only 4/9 to be Fascist and the events that two individuals are fascist is not independent. A more careful calculation would have been:

- There are $\binom{9}{4}$ possible distributions of Fascists in which $\binom{7}{4}$ of them both Marek and Sam are good for a probability of 5/18
- There are $\binom{7}{3}$ ways for Marek to be Fascist and Sam to be Liberal. Of course $\binom{7}{4} = \binom{7}{3}$ so we get 5/18 again.
- The remaining event that both are evil must then be 3/18.

So a better perspective would have been:



	Marek is a Liberal	Marek is a Fascist
Sam is a Liberal	27.7	27.7
Sam is a Fascist	27.7	16.6

I don't think I could do this math consistently in game though so I'll do the rest of the analysis with my original priors. I've included it here for reference in your future 10 player games of Secret Hitler.

When Marek declared Sam was Fascist the only scenario which is confidently eliminated is that both are liberal. Any reasonable liberal player is truth promoting and has no reason to lie. At first glance it seems that the possibility that both are fascist is also eliminated as the fascists should have no reason to fight each other. This doesn't strictly hold though. The fascists only need one fascist in a government to likely sink it and if they reason that the liberals will reason that one of them must be good then they have good reason to pick fights with each other. But in practice in an accusation situation the table often opts to pick neither so its a risky move.

Now we get into the questionable deduction I made during the game. If Marek is a fascist and he inspects a liberal he has a choice to make. He can accuse the liberal of being fascist to sow distrust among the liberals. Or he can tell the truth to garner trust with the liberals. If Marek is liberal and he inspects a fascist then he has no choice to make. He will declare that the fascist is a fascist.

In the moment I figured there was about a 50-50 chance that a fascist would choose to call a liberal a fascist or a liberal. Let's call fascists who lie about liberal's identities bold and fascists who tell the truth timid. I discounted the possibility that both were fascist and I reasoned with probabilities from the first square. So given that Marek had accused Sam that meant that Marek was a liberal with probability 75%.

The problem is I didn't adequately update on Marek's fascist presidency. In my mental model of the game it's not the improbable to draw 3 fascist articles. This mental model is derived from the fact that it generally happens once or twice a game. But its still an unlikely event in the sense that I should consider it evidence that the president was fascist. From the reports of the first group 3 fascist articles had already been buried. Even if I distrust them I can still guess at least 2 fascist articles were buried. So the probability that 3 fascist articles were drawn again is at most $\binom{11}{3}/\binom{14}{3} \approx 0.23$. Going into the investigation I had a belief that Marek was Fascist with probability about 50% but I should have already updated to ~88% that he was a fascist as opposed to an unlucky liberal (100% of the time he didn't draw 3 fascist articles he buried a liberal and is a fascist and half the rest of the time he's a fascist by my prior). Given that, even with my conjecture that Fascists make false accusations only half the time I should have guessed Marek was more likely fascist than Sam. Marek's accusation demonstrated Marek is not a timid fascist which I conjectured to be half the fascist probability mass. By Bayes' Theorem I should have updated to Marek being fascist with probability:

$$P(\text{Marek a Fascist}|\text{Marek not a timid Fascist}) =$$

$$\frac{P(\text{Marek not a timid Fascist}|\text{Marek a Fascist})P(\text{Marek a Fascist})}{P(\text{Marek not a timid Fascist})} \approx$$

$$\frac{0.5 \cdot 0.88}{0.12 + 0.88/2} \approx 0.79$$

I definitely computed badly in the moment. I think my model building was also bad in a number of other ways which are harder for me to put numbers on:

- I thought Marek was unlikely to pick a fight since he seemed relatively new and he was very quiet after his accusation. In my mind people lie about other people's

identities prepared to fight and Marek seemed sort of timid. The counterpoint to this is lying is the obvious level one strategy for fascists. A reasonable person might think its just what fascists are supposed to do.
- I didn't factor in the probability that both were fascists at all.
- All the players seemed to jump on Marek. I think part of the reason I defended him was a contrarian bias. But being contrarian against too big a consensus in Secret Hitler is important. If everyone agrees about something then some fascists are agreeing.
- On the meta level I was very sleepy and new myself to not be reasoning that well or remembering basic facts that accurately. I probably should have deferred to the group.

Thanks for reading and let me know if you have any other thoughts about the position.

---

1. The development mafia -> resistance -> avalon -> Secret Hitler represents substantial progress in board gaming technology. There's also a lot of amazing adjacent games like Two Rooms and a Boom and One Night Ultimate Werewolf. I'm thankful to live in a time of extraordinary board game technological progress. ↩

2. Our meta was such that the chancellorship rotated counterclockwise as the presidency rotated clockwise to see the maximum number of players. In later games our meta updated to make the chancellor 3 to the left of the president and neining to them if we got a favorable result which seems to be a very powerful strategy for the liberals. ↩

The Commitment Races problem

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

[Epistemic status: Strong claims vaguely stated and weakly held. I expect that writing this and digesting feedback on it will lead to a much better version in the future. EDIT: So far this has stood the test of time. EDIT: As of September 2020 I think this is one of the most important things to be thinking about.]

This post attempts to generalize and articulate a problem that people have been thinking about since at least 2016. [Edit: 2009 in fact!] In short, here is the problem:

Consequentialists can get caught in commitment races, in which they want to make commitments as soon as possible. When consequentialists make commitments too soon, disastrous outcomes can sometimes result. The situation we are in (building AGI and letting it self-modify) may be one of these times unless we think carefully about this problem and how to avoid it.

For this post I use "consequentialists" to mean agents that choose actions entirely on the basis of the expected consequences of those actions. For my purposes, this means they don't care about historical facts such as whether the options and consequences available now are the result of malicious past behavior. (I am trying to avoid trivial definitions of consequentialism according to which everyone is a consequentialist because e.g. "obeying the moral law" is a consequence.) This definition is somewhat fuzzy and I look forward to searching for more precision some other day.

Consequentialists can get caught in commitment races, in which they want to make commitments as soon as possible

Consequentialists are bullies; a consequentialist will happily threaten someone insofar as they think the victim might capitulate and won't retaliate.

Consequentialists are also cowards; they conform their behavior to the incentives set up by others, regardless of the history of those incentives. For example, they predictably give in to credible threats unless reputational effects weigh heavily enough in their minds to prevent this.

In most ordinary circumstances the stakes are sufficiently low that reputational effects dominate: Even a consequentialist agent won't give up their lunch money to a schoolyard bully if they think it will invite much more bullying later. But in some cases the stakes are high enough, or the reputational effects low enough, for this not to matter.

So, amongst consequentialists, there is sometimes a huge advantage to "winning the commitment race." If two consequentialists are playing a game of Chicken, the first one to throw out their steering wheel wins. If one consequentialist is in position to seriously hurt another, it can extract concessions from the second by credibly threatening to do so--unless the would-be victim credibly commits to not give in first!

If two consequentialists are attempting to divide up a pie or select a game-theoretic equilibrium to play in, the one that can "move first" can get much more than the one that "moves second." In general, because consequentialists are cowards and bullies, the consequentialist who makes commitments first will predictably be able to massively control the behavior of the consequentialist who makes commitments later. As the folk theorem shows, this can even be true in cases where games are iterated and reputational effects are significant.

Note: "first" and "later" in the above don't refer to clock time, though clock time is a helpful metaphor for imagining what is going on. Really, what's going on is that agents learn about each other, each on their own subjective timeline, while also making choices (including the choice to commit to things) and the choices a consequentialist makes at subjective time *t* are cravenly submissive to the commitments they've learned about by *t*.

Logical updatelessness and acausal bargaining combine to create a particularly important example of a dangerous commitment race. There are strong incentives for consequentialist agents to self-modify to become updateless as soon as possible, and going updateless is like making a bunch of commitments all at once. Since real agents can't be logically omniscient, one needs to decide how much time to spend thinking about things like game theory and what the outputs of various programs are before making commitments. When we add acausal bargaining into the mix, things get even more intense. Scott Garrabrant, Wei Dai, and Abram Demski have described this problem already, so I won't say more about that here. Basically, in this context, there are many other people observing your thoughts and making decisions on that basis. So bluffing is impossible and there is constant pressure to make commitments quickly before thinking longer. (That's my take on it anyway)

> *Anecdote:* Playing a board game last week, my friend Lukas said (paraphrase) "I commit to making you lose if you do that move." In rationalist gaming circles this sort of thing is normal and fun. But I suspect his gambit would be considered unsportsmanlike--and possibly outright bullying--by most people around the world, and my compliance would be considered cowardly. (To be clear, I didn't comply. Practice what you preach!)

When consequentialists make commitments too soon, disastrous outcomes can sometimes result. The situation we are in may be one of these times.

This situation is already ridiculous: There is something very silly about two supposedly rational agents racing to limit their own options before the other one limits theirs. But it gets worse.

Sometimes commitments can be made "at the same time"--i.e. in ignorance of each other--in such a way that they lock in an outcome that is disastrous for everyone. (Think both players in Chicken throwing out their steering wheels simultaneously.)

Here is a somewhat concrete example: Two consequentialist AGI think for a little while about game theory and commitment races and then self-modify to resist and heavily punish anyone who bullies them. Alas, they had slightly different ideas about what counts as bullying and what counts as a reasonable request--perhaps one thinks that demanding more than the Nash Bargaining Solution is bullying, and the other thinks that demanding more than the Kalai-Smorodinsky Bargaining Solution is bullying--so many years later they meet each other, learn about each other, and end up locked into all-out war.

I'm not saying disastrous AGI commitments are the default outcome; I'm saying the stakes are high enough that we should put a lot more thought into preventing them than we have so far. It would really suck if we create a value-aligned AGI that ends up getting into all sorts of fights across the multiverse with other value systems. We'd wish we built a paperclip maximizer instead.

Objection: "Surely they wouldn't be so stupid as to make *those* commitments--even *I* could see that bad outcome coming. A better commitment would be..."

Reply: The problem is that consequentialist agents are motivated to make commitments as soon as possible, since that way they can influence the behavior of other consequentialist agents who may be learning about them. Of course, they will balance these motivations against the countervailing motive to learn more and think more before doing drastic things. The problem is that the first motivation will push them to make commitments much sooner than would otherwise be optimal. So they might not be as smart as us when they make their commitments, at least not in all the relevant ways. Even if our baby AGIs are wiser than us, they might still make mistakes that we haven't anticipated yet. The situation is like the centipede game: Collectively, consequentialist agents benefit from learning more about the world and each other before committing to things. But because they are all bullies and cowards, they individually benefit from committing earlier, when they don't know so much.

Objection: "Threats, submission to threats, and costly fights are rather rare in human society today. Why not expect this to hold in the future, for AGI, as well?"

Reply: Several points:

1. Devastating commitments (e.g. "Grim Trigger") are much more possible with AGI--just alter the code! Inigo Montoya is a fictional character and even he wasn't able to summon lifelong commitment on a whim; it had to be triggered by the brutal murder of his father.

2. Credibility is much easier also, especially in an acausal context (see above.)

3. Some AGI bullies may be harder to retaliate against than humans, lowering their disincentive to make threats.

4. AGI may not have sufficiently strong reputation effects in the sense relevant to consequentialists, partly because threats can be made more devastating (see above) and partly because they may not believe they exist in a population of other powerful agents who will bully them if they show weakness.

5. Finally, these terrible things (Brutal threats, costly fights) do happen to some extent even among humans today--especially in situations of anarchy. We want the AGI we built to be *less* likely to do that stuff than humans, not merely *as* likely.

Objection: "Any AGI that falls for this commit-now-before-the-others-do argument will also fall for many other silly do-X-now-before-it's-too-late arguments, and thus will be incapable of hurting anyone."

Reply: That would be nice, wouldn't it? Let's hope so, but not count on it. Indeed perhaps we should look into whether there are other arguments of this form that we should worry about our AI falling for...

> *Anecdote:* A friend of mine, when she was a toddler, would threaten her parents: "I'll hold my breath until you give me the candy!" Imagine how badly things would have gone if she was physically capable of making arbitrary credible commitments. Meanwhile, a few years ago when I first learned about the concept of updatelessness, I resolved to be updateless from that point onwards. I am now glad that I couldn't actually commit to anything then.

Conclusion

Overall, I'm not certain that this is a big problem. But it feels to me that it might be, especially if acausal trade turns out to be a real thing. I would not be surprised if "solving bargaining" turns out to be even more important than value alignment, because the stakes are so high. I look forward to a better understanding of this problem.

Many thanks to Abram Demski, Wei Dai, John Wentworth, and Romeo Stevens for helpful conversations.

Does Agent-like Behavior Imply Agent-like Architecture?

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

This is not a well-specified question. I don't know what "agent-like behavior" or "agent-like architecture" should mean. Perhaps the question should be "Can you define the fuzzy terms such that 'Agent-like behavior implies agent-like architecture' is true, useful, and in the spirit of the original question." I mostly think the answer is no, but it seems like it would be really useful to know if true, and the process of trying to make this true might help us triangulate what we should mean by agent-like behavior and agent-like architecture.

Now I'll say some more to try to communicate the spirit of the original question. First a giant look-up table is not (directly) a counterexample. This is because it might be that the only way to produce an agent-like GLUT is to use agent-like architecture to search for it. Similarly a program that outputs all possible GLUTs is also not a counterexample because you might have to use your agent-like architecture to point at the specific counterexample. A longer version of the conjecture is "If you see a program implements agent-like behavior, there must some agent-like architecture in the program itself, in the causal history of the program, or in the process that brought your attention to the program." The pseudo-theorem I want is similar to the claim that correlation really does imply causation or the good regulator theorem.

One way of defining agent-like behavior as that which can only be produced by an agent-like architecture. This makes the theorem trivial, and the challenge is making the theorem non-vacuous. In this light, the question is something like "Is there some nonempty class of architectures that can reasonably be described as a subclass of 'agent-like' such that the class can be equivalently specified either functionally or syntactically?" This looks like it might conflict with the spirit of Rice's theorem, but I think making it probabilistic and referring to the entire causal history of the algorithm might give it a chance of working.

One possible way of defining agent-like architecture is something like "Has a world model and a goal, and searches over possible outputs to find one such that the model believes that output leads to the goal." Many words in this will have to be defined further. World model might be something that has high logical mutual information with the environment. It might be hard to define search generally enough to include everything that counts as search. There also might be completely different ways to define agent-like architecture. Do whatever makes the theorem true.

Redefining Fast Takeoff

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

This post is a result of numerous discussions with other participants and organizers of the MIRI Summer Fellows Program 2019. It describes ideas that are likely already known by many researchers. However, given how often disagreements about slow/fast takeoffs come up, I believe there is significant value in making them common knowledge.

Takeoff speed & why it matters

In Superintelligence, Nick Bostrom distinguishes between slow, medium, and fast AI takeoff scenarios (where the takeoff speed is measured by how much real-world time passes between the milestones of human-level AI (HLAI) and superintelligent AI (SAI)). He argues that slow takeoff should be reasonably safe since the humanity would have sufficient time coordinate and solve the AI alignment problem, while fast takeoff would be particularly dangerous since we wouldn't be able to react to what the AI does.

Real-world time is not what matters

In many scenarios, the real-time takeoff speed indeed strongly correlates with our ability to influence the outcome. However, we can also imagine many scenarios where this is not the case. As an example, suppose we obtain HLAI by simulating humans in virtual environments, and that this procedure additionally fully preserves the simulated humans' alignment with humanity. Since this effectively increases the speed at which humanity operates, we might get a "fully controlled takeoff" even if the transition from HLAI to SAI[1] only takes a few days of real-world time. More generally, if our path to HLAI also increases the effectivity of humanity's efforts, the "effective time" we get between HLAI and SAI will scale accordingly. For example, this *might* be the case if we go the way of Iterated Distillation and Amplification or Comprehensive AI Services. Less controversially, suppose we automate most of the current programming tasks and increase the re-usability of code, such that every computer scientist becomes 100-times as effective as they are now.

Conclusion: measure useful work

Given these examples, **I think we should measure takeoff speeds not in real-world time, but rather in** (some operationalization of) **the work-towards-AI-alignment that humanity will be able to do between HLAI and SAI**. Anecdotal examples of such measures might include "integral of the human-originating GDP between HLAI and SAI" or "number of AI safety papers published between HLAI and SAI". I believe that finding a non-anecdotal operationalization would benefit many AI policy/strategy discussions.

---

1. Recall that Bostrom distinguishes between speed, collective, and quality superintelligence. Arguably, being able to simulate humans (with enough compute) already constitutes a speed superintelligence. However, I don't think this diminishes the overall point of the post. ↩

Vaniver's View on Factored Cognition

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

The View from 2018

In April of last year, I wrote up my confusions with Paul's agenda, focusing mostly on approval directed agents. I mostly have similar opinions now; the main thing I noticed on rereading it was I talked about 'human-sized' consciences, when now I would describe them as larger than human size (since moral reasoning depends on cultural accumulation which is larger than human size). But on the meta level, I think they're less relevant to Paul's agenda than I thought then; I was confused about how Paul's argument for alignment worked. (I do think my objections were correct objections to the thing I was hallucinating Paul meant.) So let's see if I can explain it to Vaniver_2018, which includes pointing out the obstacles that Vaniver_2019 still sees. It wouldn't surprise me if I was similarly confused now, tho hopefully I am less so, and you shouldn't take this post as me speaking for Paul.

Factored Cognition

One core idea that Paul's approach rests on is that thoughts, even the big thoughts necessary to solve big problems, can be broken up into smaller chunks, and this can be done until the smallest chunk is digestible. That is, problems can be 'factored' into parts, and the factoring itself is a task (that may need to be factored). Vaniver_2018 will object that it seems like 'big thoughts' require 'big contexts', and Vaniver_2019 has the same intuition, but this does seem to be an empirical question that experiments can give actual traction on (more on that later).

The hope behind Paul's approach is *not* that the small chunks are all aligned, and chaining together small aligned things leads to a big aligned thing, which is what Vaniver_2018 thinks Paul is trying to do. A hope behind Paul's approach is that the small chunks are incentivized to be *honest*. This is possibly useful for transparency and avoiding inner optimizers. A separate hope with small chunks is that they're cheap; mimicking the sort of things that human personal assistants can do in 10 minutes only requires lots of 10 minute chunks of human time (each of which only costs a few dollars) and doesn't require figuring out how intelligence works; that's the machine learning algorithm's problem.

So how does it work? You put in an English string, a human-like thing processes it, and it passes out English strings--subquestions downwards if necessary, and answers upwards. The answers can be "I don't know" or "Recursion depth exceeded" or whatever. The human-like thing comes preloaded (or pre-trained) with some idea of how to do this correctly; obviously incorrect strategies like "just pass the question downward for someone else to answer" get ruled out, and the humans we've trained on have been taught things like how to do good Fermi estimation and some of the alignment basics. This is general, and lets you do anything humans can do in a short amount of time (and when skillfully chained, anything humans can do in a long amount of time, given the large assumption that you can serialize the relevant state and subdivide problems in the relevant ways).

Now schemes diverge a bit on how they use factored cognition, but in at least some we begin by training the system to simply imitate humans, and then switch to training the system to be good at answering questions or to distill long computations into cached answers or quicker computations. One of the tricks we can use here is that ‘self-play’ of a sort is possible, where we can just ask the system whether a decomposition was the right move, and this is an English question like any other.

Honesty Criterion

Originally, I viewed the frequent reserialization as a solution to a security concern. If you do arbitrary thought for arbitrary lengths of time, then you risk running into inner optimizers or other sorts of unaligned cognition. Now it seems that the real goal is closer to an ‘honesty criterion’; if you ask a question, all the computation in that unit will be devoted to answering the question, and all messages between units are passed where the operator can see them, in plain English.[1]

Even if one succeeds at honesty, it still seems difficult to maintain both generality and safety. That is, I can easily see how factored cognition allows you to stick to cognitive strategies that definitely solve a problem in a safe way, but don't see how it does that and allows you to develop new cognitive strategies to solve a problem that doesn’t result in an opening for inner optimizers--not within units, but within assemblages of units. Or, conversely, one could become more general while giving up on safety. In order to get both it seems like we’re resting a lot on the Overseer’s Manual or way that we trained the humans that we used as training data.

Serialized State is Inadequate or Inefficient

In my mind, the primary reason to build advanced AI (as opposed to simple AI) is to accomplish megaprojects instead of projects. Curing cancer (in a way that potentially involves novel research) seems like a megaproject, whereas determining how a particular protein folds (which might be part of curing cancer) is more like a project. To the extent that Factored Cognition relies on the serialized state (of questions and answers) to enforce honesty on the units of computation, it seems like that will be inefficient for problems whose state are large enough that they impose significant serialization costs, and inadequate for problems whose state are too large to serialize. If we allow answers that are a page long at most, or that a human could write out in 10 minutes, then we’re not going to get a 300-page report of detailed instructions. (Of course, allowing them to collate reports written by subprocesses gets around this difficulty, but means that we won’t have ‘holistic oversight’ and will allow for garbage to be moved around without being caught if the system doesn’t have the ability to read what it’s passing.)

The factored cognition approach also has a tree structure of computation, as opposed to a graph structure, which leads to lots of duplicated effort and the impossibility of horizontal communication. If I’m designing a car, I might consider each part separately, but then also modify the parts as I learn more about the requirements of the other parts. This sort of sketch-then-refinement seems quite difficult to do under the factored cognition approach, even though it involves reductionism and factorization.

Shared memory partially solves this (because, among other things, it introduces the graph structure of computation), but now reduces the guarantee of our honesty criterion because we allow arbitrary side effects. It seems to me like this is a necessary component for most of human reasoning, however. James Maxwell, the pioneer behind electromagnetism, lost most of his memory with age, in a way that seriously reduced his scientific productivity. And factored cognition doesn’t even allow the external notes and record-keeping he used to partially compensate.

There's Actually a Training Procedure

The previous section described what seems to me to be a bug; from Paul's perspective this might be a necessary feature because his approaches are designed around taking advantage of arbitrary machine learning, which means only the barest of constraints can be imposed. IDA presents a simple training procedure that, if used with an extremely powerful model-finding machine learning system, allows us to recursively surpass the human level in a smooth way. (Amusingly to me, this is like Paul *enforcing* slow takeoff.)

Training The Factoring Problem is Ungrounded

From my vantage point, the trick that we can improve the system by asking it questions like “was X a good way to factor question Y?”, where X was the attempt it had at factoring Y, is one of the core reasons to think this approach is workable, and also seems like it won’t work (or will preserve blind spots in dangerous ways). This is because while we could actually find the ground truth on how many golf balls fit in a 737, it is much harder to find the ground truth on what cognitive style most accurately estimates how many golf balls fit in a 737.

It seems like there are a few ways to go about this:

1. Check how similar it is to what you would do. A master artist might watch the brushstrokes made by a novice artist, and then point out wherever the novice artist made questionable choices. Similarly, if we get the question “if you’re trying to estimate how many golf balls fit in a 737, is ‘length of 737 * height of 737 * width of 737 / volume of golf ball’ a good method?” we just compute what we would have done and estimate if the approach will have a better or worse error.
2. Check whether or not it accords with principles (or violates them). Checking the validity of a mathematical proof normally is done by making sure that all steps are locally valid according to the relevant rules of inference. In a verbal argument, one might just check for the presence of fallacies of reasoning.
3. Search over a wide range of possible solutions, and see how it compares to the distribution. But how broadly in question-answer policy space are we searching?

We now face some tradeoffs between exploration (in a monstrously huge search space, which may be highly computationally costly to meaningfully explore) and rubber-stamping, where I use my cognitive style to evaluate whether or not my cognitive style is any good. Even if we have a good resolution to that tradeoff, we have to deal with the cognitive credit-assignment problem.

That is, in reinforcement learning one has to figure out which actions taken (or not taken) before a reward led to receiving the reward so that it can properly assign credit; similarly the system that's training the Q&A policy needs to understand well enough how the policy is leading to correct answers such that it can apply the right gradients in the right places (or use a tremendous amount of compute doing this by blind search).

This is complicated by the fact that there may be multiple approaches to problem-solving that are internally coherent, but mixtures of those approaches fail. If we only use methods like gradient-descent that smoothly traverse the solution space, this won't be a problem (because gradient descent won't sharply jump from one to another), but it's an open empirical question as to whether future ML techniques will be based on gradient descent. It’s not obvious how we can extricate ourselves from the dependence on our learned question-answer policy. If I normally split a model into submodels based on a lexicographical ordering, and now I’m considering a hypothetical split into submodels based on statistical clustering, I would likely want to consider the hypothetical split all the way down the tree (as updates to my beliefs on ‘what strategy should I use to A this Q?’ will impact more than just this question), especially if there are two coherent strategies but a mixture of the strategies is incoherent. But how to implement this is nonobvious; am I not just passing questions to the alternate branch, but also a complete description of the new cognitive strategy they should employ? It seems like a tremendous security hole to have ‘blindly follow whatever advice you get in the plaintext of questions’ as part of my Q->A policy, and so it seems more like I should be spinning up a new hypothetical agent (where the advice is baked into their policy instead of their joint memory) in a way that may cause some of my other guarantees that relied on smoothness to fail.

Also note that because updates to my policy impact other questions, I might actually want to consider the impact on other questions as well, further complicating the search space. (Ideally, if I had been handling two questions the same way and discover that I should handle them separately, my policy will adjust to recognize the two types and split accordingly.) While this is mostly done by the machine learning algorithm that’s trying to massage the Q->A policy to maximize reward, it seems like making the reward signal (from the answer to this meta-question) attuned to how it will be used will probably make it better (consider the answer “it should be answered like these questions, instead of those,” though generally we assume yes/no answers are used for reward signals).

When we have an update procedure to a system, we can think of that update procedure as the system's "grounding", or the source of gravity that it becomes arranged around. I don't yet see a satisfying source of grounding for proposals like HCH that are built on factored cognition. Empiricism doesn't allow us to make good use of samples or computation, in a way that may render the systems uncompetitive, and alternatives to empiricism seem like they allow the system to go off in a crazy direction in a way that's possibly unrecoverable. It seems like the hope is that we have a good human seed that then is gradually amplified, in a way that seems like it might work but relies on more luck than I would like: the system is rolling the dice whenever it makes a significant transition in its cognitive style, as it can no longer fully trust oversight from previous systems in the amplification tree as they may misunderstand what's going on in the contemporary system, and it can no longer fully trust oversight from itself, because it's using the potentially corrupted reasoning process to evaluate itself.

---

1. Of course some messages could be hidden through codes, but this behavior is generally discouraged by the optimization procedure, as whenever you compare to a human baseline they will not do the necessary decoding and will behave in a different way, costing you points. ↩

Tabooing 'Agent' for Prosaic Alignment

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

This post is an attempt to sketch a presentation of the alignment problem while tabooing words like agency, goals or optimization as core parts of the ontology.[1] This is not a critique of frameworks which treat these topics as fundamental, in fact I end up concluding that this is likely justified. This is not a 'new' framework in any sense, but I am writing it down with my own emphasis in case it helps others who feel sort of uneasy about standard views on agency. Any good ideas here probably grew out of Risks From Learned Optimization or my subsequent discussions with Chris, Joar and Evan.

Epistemic State: Likely many errors both in facts and emphasis, I would be very happy to find out where they are.

Prosaic AI Alignment as a generalization problem

I think the current and near-future state of AI development is well-described by us having:

1. very little understanding of intelligence (here defined as something like "generally powerful problem solvers"), but

2. a lot of 'dumb' compute.

Prosaic AGI development is, in my view, about using 2 to get around 1. A very simplistic model of how to do this involves three components:

- A parametrized *model-space* large enough that we think it contains the sorts of generalized problem solvers we want.
- A *search criteria* which we can test for.
- A *search process* which uses massive compute to find parameters in the model-space satisfying the search criteria.

Much of ML is about designing all these parts to make the search process as compute-efficient as possible, for instance by making everything differentiable and using gradient-descent. For the purposes of this discussion I will consider an even simpler model where the search process simply samples random models from some *prior* over the model-space until it finds one satisfying some (boolean) search criteria.

While we are generally ignoring computational and practical concerns it is important that the search criteria is limited - you can only check that the model acts correctly on a small fraction of the possible situations we want to use this model in. We might talk about a search criteria being *feasible* if it is both possible to gather the data required to specify it and reasonable to expect to actually find a model fulfilling it with the amount of compute you have. The goal then is to pick a model-space, prior and feasible search criteria such that the model does what we want in any situation it might end up in. Following Paul Christiano (in Techniques for optimizing worst case performance) we might broadly distinguish two ways that this could be false, which we will refer to as 'generalization failures':

- **Benign failure:** It does something essentially random or stupid, which has little unexpected effect on the world.
- **Malign Failure:** It generalizes 'competently' but not in the way we want, affecting the world in potentially catastrophic and unexpected ways.

Benign failures are seen a lot in current ML and often labelled robustness problems, where distributional shifts or adversarial examples lead to failures of generalization. These failures don't seem very dangerous, and can usually be solved eventually through iteration and fine-tuning. This is not the case for malign generalization failure, which have far great risks. (Slightly breaking the taboo: the classic stories of training a deceptively aligned expected utility maximizer which only does what we want because it realizes it is being tested is a malign generalization failure, though in this framework this is just an example, and whether this is central or not is an empirical claim which will be explored in the second section.)

A contrived story which doesn't rely on superintelligence but also demonstrates a malign generalization failure is:

> We are searching for a good design for a robot to clean up garbage dumps, so we run a bunch of simulations until we find one which passes our selection criteria of clearing all the garbage. We happily release this robot in the nearest garbage dump, and find that it does indeed clear the garbage, but alarmingly it does this by manufacturing self-replicating garbage-eating nano-bots. These nano-both quickly consume the earth. The robot itself knows nothing other than how to construct this precise set of nano-bots from materials found in a garbage-dumb, which is impressive but not generally intelligent.

Another class of examples are things which are generally intelligent, but in a very messy way with many blind spots and weird quirks, some of which eventually lead to catastrophic outcomes. I think a better understanding of which kinds of malign generalization errors we might be missing could be potentially very important.

So how do we shape the way the model generalizes? I think the key question is understanding how the *inputs* to the search process (the model-space, the prior and the search criteria) affect the *output,* which I think is best understood as a *posterior* distribution in the model-space gained by conditioning on the model clearing the search criteria.

Some examples of how the inputs might relate to the outputs:

- With an empty search criteria the posterior is equal to the prior and unless the prior is already very specific you should expect sampling from the posterior to give models acting randomly and getting benign failures everywhere.
- If our search criteria requires unreasonable performance on some small test-set, and our prior doesn't give a significant enough bias toward simple/general models then we should expect benign generalization failures due to 'overfitting'.
- If our search criteria, prior and model-space only focus on a limited task but are set up to correctly identify general solutions to this task then we might expect little generalization error within this task, while getting almost entirely benign

errors outside the task. This seems to be where current ML systems are situated when they work well.
- If we pick search criteria which require good performance on increasingly general tasks, and we make sure that the prior is increasingly weighted toward the right kind of simple/general solutions then we might expect to see less generalization failure overall in a broad domain, but we also risk malign generalization errors appearing.

Summing up, I think a reasonable definition of the prosaic AI alignment project is to prevent malign generalization error from ever happening, even as we try to eliminate benign errors. This seems difficult mostly because moving toward robust generalization and toward malign generalization seem very similar, and you need some way to differentially advantage the first. Some approaches to this include:

- Design a model-space and prior which advantages the sort of 'intended generalization' that we want, or which are transparent enough that we can use really powerful search criteria.
- Design search criteria which effectively shift the distribution to one containing mostly robustly generalizing models. Such criteria would likely involve a lot of inspection to see how the model actually works and generalizes internally. Paul Christiano's approach of adversarial training seems like a plausibly good way to do this.
- Find ways of making stronger search criteria feasible. An example of this is that the search criterion might be bottle-necked by human judgement and oversight, and amplification is a scheme which might remove this bottleneck and make more detailed search criteria more feasible.
- Consider whether we can replace the idea of one big system which should generalize to any situation with many specialized systems which are allowed to have benign generalization failures in many domains. This might correspond to a more 'comprehensive services'-style solution, as described by Eric Drexler and summarized by Rohin Shah.

Reintroducing agency

Now I will try to give an account of why the concepts of rational agency/goals/optimizers might be useful in this picture, even though they aren't explicitly part of the problem statement nor the mentioned solutions. This is based on a hand-wavy hypothesis:

H*: If you have a prior and model-space which sufficiently advantages descriptive simplicity and a selection criteria which tests performance in a sufficiently general set of situations, then your posterior distribution on the model-space will contain a large measure of models internally implementing highly effective expected utility optimization for some utility function.*

There are several arguments supporting this hypothesis, such as those presented by Eliezer in Sufficiently Optimized Agents Appear Coherent and the simplicity/effectiveness of simple optimization algorithms.

If H is true then it provides a good reason to study and understand goals, agency and optimization as describing properties of a particular cluster of models which will play a very important role once we start using these methods to solve very general classes of problems.

As a slight aside, this also gives a framing for the much discussed mesa optimization problem in the Risks from Learned Optimization paper, which points out that there is no *a priori* reason to expect the utility function to be the one you might have used to grade the model as part of the selection criteria, and that most of the measure might in fact taken up by *pseudo-aligned* or *deceptively-aligned* models, which represent a particular example of malign generalization error. In fact, if H is true, avoiding malign generalization errors largely comes down to avoiding misaligned mesa optimizers.

I think the world where H is true is a good world, because it's a world where we are *much* closer to understanding and predicting how sophisticated models generalize. If we are dealing with a model doing expected utility maximization we can 'just' try to understand whether we agree with its goal, and then essentially trust that it will correctly and stably generalize to almost any situation.

If you agree that understanding how an expected utility maximizer generalizes could be easier than for many other classes of minds, then studying this cluster of model-space could be useful even if H is false, as long as the weaker hypothesis H' still holds.

H'*: We will be able to find* ***some*** *model-space, prior and feasible selection criteria such that the posterior distribution on the model-space contains a large measure of models internally implementing highly effective expected utility maximization for some utility function.*

In the world where H' holds we can then restrict ourselves to this way of searching, and can thus use the kinds of methods and assumptions which we could in the world where H was true.

In either of these cases I think current models of AI Alignment which treat optimizers with goals as the central problem are justified. However, I think there are reasons to believe H and possibly even H' might be false, which essentially come down to embedded agency and bounded rationality concerns pushing away from elegant agent frameworks. I hope to write more about this at some point in the future. I also feel generally uncomfortable resting the safety of humanity on assumptions like this, and would like a much better understanding of how generalization works in other clusters or parts of various model-spaces.

Summary

I have tried to present a version of the prosaic AI alignment project which doesn't make important reference to the concept of agency, instead viewing it as a **generalization problem** where you are trying to avoid finding models which fail disastrously when presented with new situations. Agency then reappears as a potentially important cluster of the space of possible models, which under certain empirical hypotheses justifies it as the central topic, though I still wish we had more understanding of other parts of various model-spaces.

---

1. Thanks to Evan Hubinger, Lukas Finnveden and Adele Lopez for their feedback on this post! ↩

Creating Environments to Design and Test Embedded Agents

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Creating a Proper Space to Design and Test Embedded Agents

Introduction

I was thinking about the embedded agency sequence again last week, and thought “It is very challenging to act and reason from within an automaton.” Making agents which live in automata and act well in dilemmas is a concrete task for which progress can be easily gauged. I discussed this with a group, and we came up with desiderata for an automaton in which interesting agent strategies emerge.

In this post, I list some inspirations, some desiderata for this testing space, and then a sketch of a specific implementation.

The embedded agency paper puts out four main difficulties of being embedded. The goal here is to design an automaton (i.e. agent environment) which can represent dilemmas whose solutions require insights in these four domains.

- Decision theory: Embedded agents do not have well-defined IO channels.
- World models: Embedded agents are smaller than their environment.
- Robust delegation: Embedded agents are able to reason about themselves and self-improve.
- Subsystem alignment: Embedded agents are made of parts similar to the environment.

Inspiration

I draw concepts from four different automatons.

Core war is close to what we need I think. It’s a game that takes place on (e.g.) a megabyte of RAM, where one assembly instruction takes up one memory slot, agents are assembly programs, and they fight each other in this RAM space. It has several relevant features such as determinism, lack of input/output/interference, and fully embedded computation. I suspect limiting range of sight could make core war more interesting from an artificial-life point of view, because then agents need to travel around to see the world.

Conway’s game of life is Turing complete, so can allow arbitrary program specification in principle, and so can encode basically anything you can write in a programming language. It’s extremely simple but I think too verbose to hand-write dilemmas in it.

Botworld 1.0 is a cellular automaton designed to be interesting in an embedded-agency sense and it’s useful but pretty complex. They do encode interesting problems such as stag hunt and prisoner’s dilemma. I think something closer to core war, which is more parsimonious, would be more fruitful to design programs in.

Real life is *maybe* basically an automaton. Almost all effects are localized, with waves and particles traveling at most the speed of light. The transition function is nondeterministic and there are real numbers and complex numbers in the mix, but I don’t know if any of this makes a big difference from an agent-design point of view. We still have all the problems of decision theory, world models, robust delegation, and subsystem alignment.

Desiderata for Automaton

Standard decision/game theory dilemmas are representable

The 5 & 10 problem, (twin) prisoner’s dilemma, transparent Newcomb problem, death in Damascus, stag hunt, and other problems should at least be expressible in the automaton. The 5 & 10 problem especially needs to be representable. Getting the agent to know what the situation is and getting it to act well are both separate problems from setting up the dilemma.

It’s actually possible for agents to win; Information is discoverable

In order to decide between $5 and $10, the agent first needs to know the two available actions and their utilities. Of course you can not endow your agent with this knowledge because the same agent code needs to work in many dilemmas, plus it’s trivial to make an agent which passes one dilemma.

So how should agents find and use knowledge from the world? Core wars programs (called “warriors”) use conditional jumps [1]; there’s no notion of reading in or outputting on a value, but you can condition your action on a specific value at a specific location. I think this should be how agents use information and make decisions at the lowest level. (Maybe any other way of reading and using knowledge is reducible to something like this anyway?)

[1] “If `a < b` jump to instruction `x` else jump to instruction `y`”

Agents are scorable, swappable, and comparable; Money exists

My hope for creating this automaton is that I (and others) will design agents for it which use self-knowledge & successor-building & world-modeling as emergent strategies; those strategies should not be explicitly advantaged by the physics. Yet agent designers need some sort of objective to optimize when designing agents; it needs to be clear when one agent is better than another in a given environment. The best solution I can think of is to have “dollars” lying around in the world, and the objective of agent-designers is to have the agent collect as many dollars as possible.

An environment includes insertion points for where agents should begin at the first timestep and the max agent size (or other constraints). The command-line utility takes in an environment file and the appropriate number of agent files and returns the number of dollars that each agent get in that world. So you could put `agent1` in transparent Newcomb, then try with `agent2`, and see which did better and how much money they made. There could also be an option for logging or interrupting & modifying the environment or something.

Omega is representable

In general, nothing within a world can do perfect simulation of any portion of the world including itself, because the simulator is always too small, but it is possible to do pretty-good prediction. Some of the most interesting dilemmas require the presence of reliable predictors, and some of our hardest decisions in life are hard because other people are predicting us, so we want predictors to be possible within ordinary world-physics. Call the reliable predictor “Omega”.

We need agents to understand what Omega is and what it’s doing but somehow not be able to screw with it. This could be done with read-only zones or by giving Omega ten turns before the agent gets one turn; the turn-management could be done with “energy tokens” which agents spend as actions.

Omega also needs to somehow safely execute the agent without getting stuck in infinite loops or allowing the simulation to escape or move the money around or something. I have no idea about this part. Perhaps the reliable predictor should just sit outside the universe. Or we could just say that it’s against the spirit of the game to screw with Omega.

Engineering details

- The automaton is a command-line program that takes in one environment file (including agent insertion points) and zero or more agent files, and other options.
- You can generate these environment & agent files and agent files using your programming language of choice, or by hand, or by putting ink on a chicken’s feet, etc.
- An agent file is a sequence of instructions (`operation A B`)
- An option to enable logging
- An option for debug/interference mode
- An option for max timesteps
- The automaton should run fast, so hill climbing over agent programs is feasible.
- An error is thrown immediately if the agent or environment has any initial errors.

An initial specification

I don’t know if this is sufficient or well-designed but here’s my current idea for an automaton. I am mostly copying core wars.

- The world is a finite number of ‘memory slots’
    - Instructions give relative distances, e.g. +5 or -5, not go straight to 0x1234
    - Memory is a circular tape. (Distances are taken modulus memory size.)

        - This implies agents cannot know their index in space, but it doesn't matter anyway
    - One memory slot is a triple `(isMoney, hasInstructionPointer, value)`
    - Every operation takes two arguments, which are the values of the locations which the next two slots point to
        - It's like a function call `operation(*A, *B)` where `A` and `B` are the two slots after `operation`, and `*A` means "value stored at location A".
- Each 'agent' is essentially an instruction pointer to a memory slot
    - Each timestep, each agent executes their pointer in order
        - If the command is not a jump, then the pointer jumps three slots ahead, to what should be the next instruction
- Agents have many operations available
    - Special stuff
        - DIE or any invalid operation kills the agent. Used e.g. when two agents want to kill each other, so they try to get each other to execute this, like in core wars
        - JIM jump to A if B is money
        - JIP jump to A if B is an instruction pointer
    - Regular assembly-like stuff:
        - DAT works as a no-op or for data storage. If you wanted to store 7 and 14 in your program, then you could have `DAT 7 14` as a line in your agent program, and then reference those values by relative position later.
        - MOV copy from A to B
        - ADD add A to B and store in B
        - SUB subtract A from B and store in B
        - JMP unconditional jump to A
        - CMP skip next instruction if A = B
        - SLT skip next instruction if A < B
        - JMZ jump to A if B=0
        - JMN jump to A if B!=0
        - DJN decrement B, then jump to A if B is not 0
- ¿Limit relative jumps to 100 or something in magnitude, then Omega can build an invincible wall around itself and blow up anything that tries to reach it?

A Contest Thing?

I could publish a collection of public environments and create some private environments too. People can design agents which score well in the public environment, then submit it for scoring on the private environments, like a Kaggle contest. This, like any train/test split, would reduce overfitting.

Attempting to Specify a Couple Dilemmas and Agents

Vanilla prisoner's dilemma

Two programs in two places in memory. Left is somehow defect and right is somehow cooperate. Agents can see each other's code and reason about what the other will do. Omega kills everyone in 1000 timesteps if no decision is reached or something.

Twin prisoner’s dilemma

Omega copies the same agent to two places in memory. Left is defect and right is cooperate.

Transparent Newcomb’s Problem / Parfit’s Hitchhiker

Omega has two boxes and it gives the agent access to one or the other depending on its behavior.

Omega itself in Transparent Newcomb

Something like this code:

1. Environment starts with $1,000,000 in one walled box and $0 in another box. Money cannot be created, only absorbed by agents’ instruction pointers.
2. Find where the agent is by searching through memory
3. Copy it locally and surround it with walls
4. Put trailer code on the agent that returns the instruction pointer to omega when the agent is done
5. Execute the agent
6. If it e.g. one-boxed then destroy the walls surround
7. Somehow now kickstart the real agent¿

Agent in Transparent Newcomb

Probably very flawed but…

1. Search space for any other programs
2. Somehow analyze it for copy-and-run behavior
3. Somehow infer that this other program is controlling walls around money tokens
4. Somehow infer that if you one-box then you’ll get the bigger reward
5. Output one-box by writing a 1 to the designated spot
    - This spot is somehow inferred through analyzing omega i guess
        - Or it could be a standard demarkation that many agents use, e.g. 1-2-3-3-2-1 as “communication zone”

Pre-mortem

I’ll briefly raise and attempt to respond to some modes of failure for this project.

The agent insertion point did too much work

It could turn out that the interesting/challenging part of the embedded agency questions is in drawing the boundary around the agent, so giving the sole starting location of the agent is dodging the most important problem. I think that this problem is fully explored, however, if we somehow pause the agent and let some other things copy & use its code before the agent runs. Then the agent must figure out what has happened and imagine other outcomes before it chooses actions.

The agent-designer is outside of the environment. Cartesian!

The human or evolutionary algorithm or whatever designing the agents is indeed outside of the universe, and cannot directly suffer consequences, be modified, etc. However, they cannot interfere once the simulation has started, and any knowledge they have must fully live in their program in order for it to succeed in a variety of environments. I think that, if you design an agent which passes 5&10, Newcomb, prisoner's dilemma, etc, then you *must* have made some insights along the way. Otherwise, maybe these problems were easier than I thought.

We only find successful agents through search, and they are incomprehensible

This is maybe the most likely way for this project to fail, conditioning on me actually doing the work. I would say that, even in this case, we can learn some about the agent by running experiments on it or somehow asking it questions, like how we analyze humans.

Conclusion

Automatons are a more accurate model of the difficulties of agency in the real world than reinforcement learning problems, so we need to do more task-design, agent-design, and general experimentation in this space. My plan is to create an automaton, used as a command-line utility, which will run a given set of agents in a given environment (e.g. prisoner's dilemma). Ideally, we'll have a large set of task environments, and we can design agents with the goal of generality.

Formalising decision theory is hard

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

In this post, I clarify how far we are from a complete solution to decision theory, and the way in which high-level philosophy relates to the mathematical formalism. I've personally been confused about this in the past, and I think it could be useful to people who casually follows the field. I also link to some less well-publicized approaches.

The first disagreement you might encounter when reading about alignment-related decision theory is the disagreement between Causal Decision Theory (CDT), Evidential Decision Theory (EDT), and different logical decision theories emerging from MIRI and lesswrong, such as Functional Decision Theory (FDT) and Updateless Decision Theory (UDT). This is characterized by disagreements on how to act in problems such as Newcomb's problem, smoking lesion and the prisoner's dilemma. MIRI's paper on FDT represents this debate from MIRI's perspective, and, as exemplified by the philosopher who refereed that paper, academic philosophy is far from having settled on how to act in these problems.

I'm quite confident that the FDT-paper gets those problems right, and as such, I used to be pretty happy with the state of decision theory. Sure, the FDT-paper mentions logical counterfactuals as a problem, and sure, the paper only talks about a few toy problems, but the rest is just formalism, right?

As it turns out, there are a few caveats to this:

1. CDT, EDT, FDT, and UDT are high-level clusters of ways to go about decision theory. They have multiple attempted formalisms, and it's unclear to what extent different formalisms recommend the same things. For FDT and UDT in particular, it's unclear whether any one attempted formalism (e.g. the graphical models in the FDT paper) will be successful. This is because:
2. Logical counterfactuals is a really difficult problem, and it's unclear whether there exists a natural solution. Moreover, any non-natural, arbitrary details in potential solutions are problematic, since some formalisms require everybody to know that everybody uses sufficiently similar algorithms. This highlights that:
3. The toy problems are radically simpler than actual problems that agents might encounter in the future. For example, it's unclear how they generalise to acausal cooperation between different civilisations. Such civilisations could use implicitly implemented algorithms that are more or less similar to each others', may or may not be trying and succeeding to predict each others' actions, and might be in asymmetric situations with far more options than just cooperating and defecting. This poses a lot of problems that don't appear when you consider pure copies in symmetric situations, or pure predictors with known intentions.

As a consequence, knowing what philosophical position to take in the toy problems is only the beginning. There's no formalised theory that returns the right answers to all of them yet, and if we ever find a suitable formalism, it's very unclear how it will generalise.

If you want to dig into this more, Abram Demski mentions some open problems in this comment. Some attempts at making better formalisations includes Logical Induction

Decision Theory (which uses the same decision procedure as evidential decision theory, but gets logical uncertainty by using logical induction), and a potential modification, Asymptotic Decision Theory. There’s also a proof-based approach called Modal UDT, for which a good place to start would be the 3rd section in this collection of links. Another surprising avenue is that some formalisations of the high-level clusters suggest that they're all the same. If you want to know more about the differences between Timeless Decision Theory (TDT), FDT, and versions 1.0, 1.1, and 2 of UDT, this post might be helpful.

Vague Thoughts and Questions about Agent Structures

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Epistemic status: Posting for blog day at MSFP! More trying to figure out what the right definitions are than saying anything concrete. I still don't really know what agents are, and none of this is math yet. I'm hoping to develop these (and other) ideas more in the future, so any feedback is greatly appreciated.

My Naive Agent Pre-model

Y'know -- agents are, like, things that do things. They have utility functions and stuff. They make choices, whatever that means.

Unfortunately this 'definition' isn't sufficient for making any concrete claims about how agents behave, so I've been thinking about some models that might be, and this post contains ideas that came out of that.

Irreducible Agents vs Agent Clusters

Irreducible Agents

An irreducible agent is what I'm calling something that optimizes a really simple utility function in some straightforward sense -- maybe it just does gradient descent or something. If it has a choice of two actions, it picks the one that results in higher utility every time. (This concept needs a precise definition, but I'm not sure what the right definition is yet, so I'm just trying to point at the thing).

It seems like when people talk about agents in the abstract, this is the kind of agent they often mean. But people also sometimes talk about things like humans as agents, and we aren't really like that. Humans seem to be at least partially made up of smaller agenty parts that have different and sometimes conflicting goals -- more like what I'm calling 'agent clusters'

Agent clusters

If you glue together a bunch of irreducible agents in a reasonable way, you could still get something that looks agenty. I can think of a couple of ways to think about gluing agents together; there are also probably better framings:

One way is by having a sort of meta-agent that turns agents on and off by some criteria, and the subagent that is turned on gets to decide what to do. I'm not sure this framing makes sense; if you can think of the meta-agent as having a utility function, it seems like it collapses to just be an irreducible agent after all. But maybe having the meta-agent use some simple rules that don’t constitute a true utility function could work to build agent clusters that efficiently approximate a more complex utility function than they could explicitly represent.

Another way is to think of the agents as voters which rank actions by their utility function and use some voting system to decide what the action of the resulting system will be. (I'm pretty sure Arrow’s Impossibility Theorem doesn’t kill this as a possible structure - it just says that you can’t do it with every set of agents, which is not surprising)

(Note that these two framings could be combined -- the meta-agent could activate multiple subagents at the same time and aggregate the opinions of only the activated subagents via some voting system. I’m not sure if this is useful)

Some questions:

Q: Are these two framings (the meta-agent framing and the electorate framing) equivalent? Are there other options or can any reasonable cluster be described in this way?

Q: Given a description of a (purported) agent as a cluster of irreducible agents in this way, what should it mean for it to be coherent and how can we tell whether it is coherent? One idea is that using the electorate model, ‘coherent’ could mean that it is possible to satisfy the desiderata from Arrow’s theorem.

Q: If you use the electorate model, are there simple conditions on the utility functions of the voters under which a coherent agent can be formed?

Towards an Intentional Research Agenda

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

This post is motivated by research intuitions that better formalisms in consciousness research contribute to agent foundations in more ways than just the value loading problem. Epistemic status: speculative.

David Marr's levels of analysis is the idea that any analysis of a system involves analyzing it at multiple, distinct levels of abstraction. These levels are the computational, which describes what it is the system is trying to do, the algorithmic, which describes which algorithms the system instantiates in order to accomplish that goal, and the implementation level, describing the hardware or substrate on which the system is running. Each level underdetermines the other levels. You can choose lots of different algorithms for a given goal, and algorithms don't restrict which goals can use them. A concrete example Marr uses, is that you'd have a very hard time figuring out what a feather was for if you'd never seen a bird flying, and if you only saw a bird flying you might have a very difficult time coming up with something like the design of a feather.

Imagine a world that had recently invented computers. The early examples are very primitive, but people can extrapolate and see that these things will be very powerful, likely transformative to society. They're pretty concerned about the potential for these changes to be harmful, maybe even catastrophic. Although people have done a bit of theoretical work on algorithms, it isn't all that sophisticated. But since the stakes are high, they try their best to start figuring out what it would mean for there to be such a thing as harmful algorithms, or how to bound general use algorithms such that they can only be used for certain things. They even make some good progress, coming up with the concept of ASICs so that they can maybe hard code the good algorithms and make it impossible to run the bad. They're still concerned that a sufficiently clever or sufficiently incentivized agent could use ASICs for bad ends somehow.

If this situation seems a bit absurd to you, it's because you intuitively recognize that the hardware level underdetermines the algorithmic level. I argue the possibility that we're making the same error now. The algorithmic level underdetermines the computational level, and no matter how many combinations of cleverly constructed algorithms you stack on themselves, you won't be able to bound the space of possible goals in a way that gets you much more than weak guarantees. In particular, a system constructed with the right intentional formalism should actively want to avoid being goodharted just like a human does. Such an agent should have knightian uncertainty and therefore also (potentially) avoid maximizing.

In physics (or the implementation level) there are notions of smallest units, and counting up the different ways these units can be combined creates the notion of thermodynamic entropy, we can also easily define distance functions. In information theory (or the algorithmic level) there are notions of bits, and counting up the different ways these bits could be creates the notion of information theoretic entropy, we can also define distance functions. I think we need to build a notion of units of intentionality (on the computation level), and measures of permutations of ways these units can be to give a notion of intentional (computational) entropy, along with getting what could turn out to be a key insight for aligning AI, a distance function between intentions.

In the same way that trying to build complex information processing systems without a concrete notion of information would be quite confusing, I claim that trying to build complex intentional systems without a concrete notion of intention is confusing. This may sound a bit far fetched, but I claim that it is exactly as hard to think about as information theory was before Shannon found a formalism that worked.

I think there are already several beachheads for this problem that are suggestive:

Predictive processing (relation to smallest units of intention).
In particular, one candidate for smallest unit is the smallest unit that a given feedback circuit (like a thermostat) can actually distinguish. We humans get around this by translating from systems in which we can make fewer distinctions (like say heat) into systems in which we can make more (like say our symbolic processing of visual information in the form of numbers).

Convergent instrumental goals (structural invariants in goal systems).
In particular I think it would be worth investigating differing intuitions about just how much a forcing function convergent instrumental goals are. Do we expect a universe optimized by a capability boosted Gandhi and Clippy to be 10% similar, 50%, 90% or perhaps 99.9999+% similar?

Modal Logic (relation to counterfactuals and as semantics for the intentionality of beliefs).

Goodhart's taxonomy begins to parameterize, and therefore define distance functions for divergence of intent.

Some other questions:

How do simple intentions get combined to form more complex intentions? I think this is tractable via experimentation with simple circuits. This could also suggest approaches to pre-rationality via explaining (rigorously) how complex priors arise from homeostatic priors.

In Buddhism, intention is considered synonymous with consciousness, while in the west this is considered a contentious claim. What simple facts, if known, would collapse the seeming complexity here?

Can we consider intentions as a query language? If so, what useful ideas or results can we port over from database science? Is the apparent complexity of human values a side effect of the dimensionality of the space more so than the degree of resolution on any particular dimension?

Note:

When I read vague posts like this myself, I sometimes have vague objections but don't write them up due to the effort to bridge the inferential distance to the author and also the sense that the author will interpret attempts to bridge that distance as harsher criticism than I intend. Please feel free to give half formed criticism and leave me to fill in the blanks. It might poke my own half formed thoughts in this area in an interesting way.

Metalignment: Deconfusing metaethics for AI alignment.

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Epistemic status: MSFP blog post day. General and very speculative ideas.

Proposition : Deconfusing metaethics might be a promising way to increase our chances of solving AI alignment.

What do I mean by metaethics?

Metaethics here is understood as an ideal procedure that humans are approximating when they reason about ethics i.e. when they are trying to build ethical theories. Let's have a look at mathematics for an analogy. Part of the mathematical production involves using some theory of logic to prove or disprove some conjecture about some mathematical object. Theorems, lemmas and properties that one can derive from axioms working with some logic is, roughly, part of how mathematics progresses. Another analogy is how we learn about regularities in the world by approximating Solomonoff induction. It seems that we are lacking some formalised, ideal rational procedure of ethical progress that would help us with sorting and generating ethical theories. Such a procedure seems difficult to figure out and potentially crucial to help solving AI alignment.

Why could this be important?

A better understanding of metaethics could help us decide among different ethical theories and how to generate new ones. Furthermore, knowing what the world should become and how AI should interact with it might requires us to make progress on how we should think about ethics to enlighten how we could think about aligning AI. For example, aligning AI with human values, learning and aggregating human preferences in some way, avoiding X-risks are all ethical propositions of what we should do. It is plausible that these views are flawed and that a better understanding of how to think about ethics might make us reconsider these normative stances and clarify what alignment means.

The following intuition is one of the main reasons why I think a better understanding of metaethics might be important to AI alignment research. As I am thinking more about ethics, arguing with others about it and getting more informed about the world, my ethical views evolve and it seems that I am making some sort of progress by sharpening my reasons for why I hold some ethical view or why some ethical theory seems flawed. Thus I tend to value more my future self's moral views to the extent that he has spent more time thinking about ethics and is more informed about the world so that I trust him more about deciding how I should go about transforming it. Similarly, it might be sensible for future AI systems to be able to instantiate a similar process of moral progress to update its utility function or goals according to the results of such a process that, if transparent and consulted by humans, could figure out how to transform the world through some long and efficient ethical reflection.

Some examples

For clarification, the following, non-exhaustive, criteria might be examples of how to evaluate ethical theories and constraints under which we could generate new ones.

- Using clean thought experiments as intuition pumps.
- Constraining ethical theories by physics and other scientific domains such as evolution or computability. For example, maximising the number of 10-dimensional pink unicorns is probably not a very good ethical theory as it demands to bring values that are meaningless and incompatible with the laws of physics. Science might not tell us what to do but it can help us in knowing what we can't do or can't consider as valuable.
- Formalising ethical theories further than a lot of existing ethical theories that are mostly represented through natural language. This could ease the evaluation and learning of ethical theories by some AI and yield more consistent ethical theories.
- Favor simplicity : avoid adding unnecessary arbitrary values.
- Favor Universality : try to be as observer independent as possible. Ideally we might want our ethical theories to be applicable not just to humans but to all sorts of other physical systems.

Possible objections

This approach of AI alignment might be too top-down in its current formulation and raise a number of difficult challenges or objections toward being a research path worth pursuing :

- There might be no such meaningful thing as 'better ethical theories' in the absolute sense but there might be some that are better for a certain class of physical systems.
- Such a project might take too long to implement. There does not seem to be any consensus regarding a better ethical theory although philosophers have been arguing and thinking about ethics for long time.
- There might not be any universally compelling argument. But we still might identify a class of arguments or ethical theories that seems more viable and use them in addition to others value learning approaches.
- Formalising ethics is too hard because it's too fuzzy. Indeed ethics plausibly emerges from genetic and memetic evolution and mostly reflects humans trying to gain some value from cooperation with other humans.

Nevertheless such a project might have the positive aspect of not speeding up AI capability research while informing us about values and how to think about alignment. One important downside though would be that there might be other more promising projects to pursue instead.

Conclusion

To conclude I would like to suggest some possible way to imagine working toward a better understanding of metaethics and producing better ethical theories. These are extremely broad and vague suggestions to stimulate research ideas.

- Build an ethical oracle that could be asked questions about ethical inconsistencies, moral blindspot or axiological problems.
- Artificial Philosopher: Input philosophy papers on ethics and metaethics and output better understanding of metaethics and more satisfying ethical theories.
- Expected utility maximiser : Update utility function in accordance to best guess about ethics for example derived from the artificial Philosopher or the ethical oracle. This would involve an additional step of translating ethical theories into utility function.
- Simulate humans so that they have more time to figure out more about ethics.
- Formalise ethics maybe using logic, probability and game theory.
- Accelerate research in moral psychology.

.

Torture and Dust Specks and Joy--Oh my! or: Non-Archimedean Utility Functions as Pseudograded Vector Spaces

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

A dozen years ago, Eliezer Yudkowsky asked us which was less (wrong?) bad:

- 3^^^3 people each getting a dust speck in their eyes,

or

- 1 person getting horribly tortured continually for 50 years.

He cheekily ended the post with "I think the answer is obvious. How about you?"

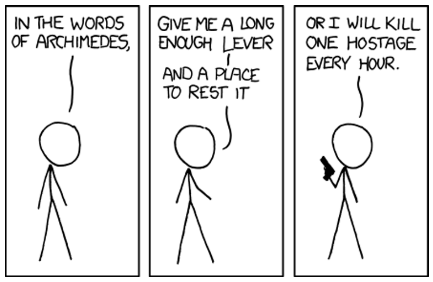
To this day, I do not have a clue which answer he thinks is obviously true, but I strongly believe that the dust specks are preferable. Namely, I don't think there's any number we can increase 3^^^3 to that would change the answer, because I think the disutility of dust specks is fundamentally **incomparable** with that of torture. If you disagree with me, I don't care--that's not actually the point of this post. Below, I'll introduce a generalized type of utility function which allows for making this kind of statement with mathematical rigor.

Linearity and Archimedes

A utility function is called **linear** if it respects scaling--5 people getting dust specked is 5 times as bad as a single dust specker. This is a natural and common assumption for what utility functions should look like.

An ordered algebraic structure (number system) is called **Archimedean** if for any two positive values x, y, there exists natural numbers n, m, such that $nx > y$ and $my > x$. In other words, adding either value to itself a finite number of times will eventually outweigh the other. Any pair x, y for which this property holds of $|x|$ and $|y|$ is called **comparable**. Note that we're taking the absolute value because we care about magnitude and not sign here: of course any positive number of people enjoying a sunset is better than any positive number of people being tortured, but the question remains whether there is some number of sunset-views which would outweigh a year of torture (in the sense that a world with n additional sunset watchers and 1 additional year of torture would be preferable to a world with no more of either).

Another way to think about comparability is that both values are relevant to an analysis of world state preference: a duster does not need to know how many dust specks are in world 1 or world 2 if they know that world 2 has more torture than world 1, because the preference is already decided by the torture differential.


IN THE WORDS OF ARCHIMEDES,
GIVE ME A LONG ENOUGH LEVER
AND A PLACE TO REST IT
OR I WILL KILL ONE HOSTAGE EVERY HOUR.

Convicted dusters now face a dilemma: a linear, Archimedean utility function necessarily yields some number of dust specks to which torture is preferable. One resolution is to reject linearity: very smart people have discussed this at length. There is another option though which, as far as the author can tell, hardly ever gets proper attention: *rejecting Archimedean utilities*. This is what I'll explore in the post.

Comparability

Comparability defines an equivalence class on utilities of world states (and their utilities).

- Any world state is comparable to itself (*reflexivity*),
- if state 1 is comparable to state 2, then state 2 is comparable to state 1 (*symmetry*), and
- if state 1 is comparable to state 2 and state 2 is comparable to state 3, then state 1 is comparable to state 3 (*transitivity*).

These properties follow immediately from the definition of comparability provided above. Now we have a partition of world states into **comparability classes**. Note that the comparability classes do not form a vector space: 1 torture + 1 dust speck is comparable to one torture, but their difference, 1 dust speck, is not comparable to either.

As I said before, I don't care if you disagree about dust specks. If you think there are any pair of utilities which are in some way qualitatively distinct or otherwise incomparable (take your pick from: universe destruction, death, having corn stuck between one's teeth, the simple joy of seeing a friend, etc--normative claims about which utilities are comparable is outside the scope of this post), then there is more than one comparability class and the utility function needs a non-Archimedean co-domain to reflect this.

If you staunchly believe that every pair of utilities is comparable, that's fine--there will be only one comparability class containing all world states, and the construction below will boringly return your existing Archimedean utility function.

Pseudograded Vector Spaces

We can assign a total ordering (severity) to our collection I of comparability classes:

$j >_S i$ means that, for any world states x, y in classes i, j, respectively, $|U(y)| > |U(nx)|$ for all $n \in N$. It is clear from the construction of comparability classes that this does not depend on which representatives x, y are chosen. Note that severity does not distinguish between severe positive and severe negative utilities: the classes are defined by magnitude and not sign. Thus, the framework is agnostic on such matters as negative utilitarianism: if one believes that reducing disutility of various forms takes priority over expanding positive utility, then that will be reflected in the severity levels of these comparability classes--perhaps a large portion of the most severe classes will be disutility-reduction based, and only after that do we encounter classes prioritizing positive utility world states.

Then we can think of utilities as being vectors, where each component records the amount of utility of a given severity level. We'll write the components with most severe first (since those are the most significant ones for comparing world states). Thus, if an agent believes that there are exactly two comparability classes:

- a severe class T which contains e.g. 50 years of torture, and
- a milder class D containing e.g. being hit with a dust speck,

then world A with 1 person being tortured and 100 dust-speckers would have utility $(-1, -100)$. Choosing between this and world B with no torture and 3^^^3 dust speckers is now easy: world B has utility $(0, -3\text{^^^}3)$, and since the first component is greater in world B, it is preferable no matter what the second component is. If this seems unreasonable to the reader, it is not a fault of the mathematics but the normative assumption that dust specks are incomparable to torture--if comparability classes are constructed correctly, then this sort of **lexicographic ordering** on utilities necessarily follows.

This is what we refer to as a **pseudograded vector space**: the utility space R^I consists of |I|-dimensional vectors, which are ***I*-pseudograded** by severity, meaning that for any vector we can identify its severity as the comparability class of the first non-zero component. Equivalently, utility values are functions from *I* to R, and thus utility functions have type $W \to (I \to R)$ (where W is the world state space) (this interpretation becomes important if we would like to consider the possibility of infinitely many comparability classes).

Formally, the utility vector space admits a grading function $g : R^I \to I$ which gives the index of the first (most severe) non-zero coordinate of the input vector. In our example, $g(U(A)) = T$ and $g(U(B)) = D$. In general, if we're comparing two world states with different severity gradings, we need only check which world W_i has greater severity, and then consult the sign of $U(W_i)(g(U(W_i)))$ (that's a function evaluation, not a product!) to determine if it's preferable to the alternative (the sign will be non-zero by definition of g). More generally, we compare worlds W_1, W_2 by checking the sign of $(U(W_1) - U(W_2))(g(U(W_1) - U(W_2)))$. Note that two utility vectors u, v satisfy $u < v$ exactly when $0 < v - u$, so the lexicographic ordering is determined fully by which utilities are preferable to 0, the utility of an empty world. We will refer to such utility vectors as **positive**, keeping in mind that this only means that the most severe non-zero coefficient is positive and not that the whole vector is.

Expected utility computations work exactly as in Archimedean utility functions: averages are performed component-wise. Thus, existing work with utilities and decision theory should import easily to this generalization.

Observe that g satisfies ultrametric-like conditions (recalling those found in p-adic norms):

- For any non-zero $r \in R$ and utility vector u, $g(ru) = g(u)$
- For any pair of utility vectors u, v, we have $g(u + v) \leq \max\{g(u), g(v)\}$ with equality if $g(u) \neq g(v)$.

One may reasonably be concerned about the well-definedness of g if they believe $|I| = \infty$: isn't it possible that *I* is not **well-ordered** under severity, meaning that some vectors may not have a *first* non-zero element? [Technically we are considering the well-orderedness of the reverse ordering on I, since we want to find the most severe non-zero utility and not the least severe].

Consider for instance the function $f(x) = \sin(1/x)$ defined on the positive reals: for all its smoothness, it has no least input with a non-zero output, and what's worse the output oscillates signs infinitely many times approaching $x = 0$ such that we aren't prepared to weigh a world state with such a utility against an empty world with utility 0. Luckily, our setting is actually a special case of the Hahn Embedding Theorem for Abelian Ordered Groups, which guarantees that any utility vector can only have non-zero entries on a well-ordered set of indices (thanks to Sam Eisenstat for pointing this out to me), and thus g is necessarily well-defined. We'll refer to this vector space as $\hat{\bigoplus}_I R$: the space of all functions $I \to R$ which each only have non-zero values on a well-ordered subset of I, and thus in particular all have a first non-zero value.

Basis Independence

Working with vector spaces, we often think of the vectors as lists of (or functions from the index set to) numbers, but this is a convenient lie--in doing so, we are fixing a **basis** with which the vector space was not intrinsically equipped. Abstractly, a vector space is coordinate-free, and we merely pick bases for ease of visualization and computation. However, it is important for our purposes here that the lexicographic ordering on utility functions is ``basis-independent", for a suitable notion of basis (otherwise, our world state preference would depend on arbitrary choices made in representations of the utility space). In fact, classical graded vector spaces have distinguished homogeneous elements of grading i (e.g. polynomials which only have degree i monomials, rather than mixed polynomials with max degree i, in the graded vector space of polynomials), but this is too much structure and would be unnatural to apply to our setting, hence *pseudo*graded vector spaces.

The standard definition of basis (a linearly independent set of vectors which spans the space) isn't sufficient here, since we have extra structure we'd like to encode in our bases--namely, the severity grading. Instead, we'll define a **graded basis** to be a choice of one positive utility vector for each grading (i.e., a map $d : I \to \hat{\bigoplus}_I R$ such that $g(d(i)) = i$--in categorical terms, d is a *section* of g).

Then suppose Alice and Bob agree that there are two severity levels, and even agree on which world states fall into each, but have picked different graded bases. Letting X=50 years of torture, and Y=1 dust speck in someone's eye, perhaps Alice's basis has $d_1(T) = -X$ and $d_1(D) = -Y$, but for some reason Bob thinks the sensible choice of representatives is $d_2(T) = -X - Y$ and $d_2(D) = -Y$--he agrees with Alice that $U(X + Y) = U(X) + U(Y)$, but factors the world state differently (while Bob may seem slightly ridiculous here, there are certainly world states with multiple natural factorizations).

Then Alice's utility for world A in her basis is $U_1(A) = (-1, -100)$ and her $U_1(B) = (0, -3\hat{}\hat{}\hat{}3)$, while Bob writes $U_2(A) = (-1, -99)$ (since $(-1)d_2(T) + (-99)d(D) = (X + Y) + 99Y = X + 100Y)$ and $U_2(B) = (0, -3\hat{}\hat{}\hat{}3)$. Note that they will both prefer world B, since the change in graded basis did not affect the lexicographic ordering on utilities. Specifically, the transformation from Alice's coordinates to Bob's is induced by left multiplication by the matrix

$$M = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}.$$

We observe two key facts about M:

- M is **lower triangular** (all entries above the main diagonal are 0), and
- M has positive diagonal entries.

These will be true of any transformations between graded bases of the same utility space: lower triangularity follows from the ultrametric property of the grading g, and positive diagonals follow from our insistence that the graded basis vectors all have positive utilities. In general (when $|I| = \infty$), it won't make sense to view M as a matrix, but we will have a linear transformation operator with the same properties.

Recalling that the ordering on utilities is determined by the positive cone of utilities (since $u < v$ iff $0 < u - v$), we simply observe that lower triangular linear transformations with positive diagonal entries will preserve the positive cone of $\widehat{\bigoplus}_I R$: if the first non-zero entry of u is positive, then Mu will also have 0 for all entries before index g(u) (since lower triangularity means changes only propagate downstream) and will scale g(u) by the positive diagonal element, returning a positive utility. Thus, our ordering is independent of graded basis and depends only upon the severity classes and comparisons within them.

What's the point?

I think many LW folk (and almost all non-philosophers) are firm dusters and have felt dismayed at their inability to justify this within Archimedean utility theory, or have thrown up their hands and given up linearity believing it to be the only option. While we will never encounter a world with 3^^^3 people in it to be dusted, it's important to understand the subtleties of how our utility functions actually work, especially if we would like to use them to align AI who will have vastly more ability than us to consider many small utilities and aggregate them in ways that we don't necessarily believe they should.

Moreover, the framework is very practical in terms of computational efficiency. It means that choosing the best course of action or comparing world states requires one to only have strong information on the most severe utility levels at stake, rather than losing their mind (or overclocking their CPU) trying to consider all the tiny utilities which might add up. Indeed, this is how most people operate in daily life; while some may counter that this is a failure of rationality, perhaps it is in fact because people understand that there are some tiny impacts which can never accumulate to more importance than the big stuff, and it can't all be chalked up to scope neglect.

Finally, completely aside from all ethical questions, I think there's some interesting math going on and I wanted people to look at it. Thanks for looking at it!

This is my first blog post and I look forward to feedback and discussion (I'm sure there are many issues here, and I hope that I will not be alone in trying to solve them). Thanks to everyone at MSFP with whom I talked about this--even if you don't think you gave me any ideas, being organic rubber ducks was very productive for my ability to formulate these thoughts in semi-coherent ways.

Soft takeoff can still lead to decisive strategic advantage

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

[Epistemic status: Argument by analogy to historical cases. Best case scenario it's just one argument among many. Edit: Also, thanks to feedback from others, especially Paul, I intend to write a significantly improved version of this post in the next two weeks. Edit: I never did, because in the course of writing my response I realized the original argument made a big mistake. See this review.]

I have on several occasions heard people say things like this:

> The original Bostrom/Yudkowsky paradigm envisioned a single AI built by a single AI project, undergoing intelligence explosion all by itself and attaining a decisive strategic advantage as a result. However, this is very unrealistic. Discontinuous jumps in technological capability are very rare, and it is very implausible that one project could produce more innovations than the rest of the world combined. Instead we should expect something more like the Industrial Revolution: Continuous growth, spread among many projects and factions, shared via a combination of trade and technology stealing. We should not expect any one project or AI to attain a decisive strategic advantage, because there will always be other projects and other AI that are only slightly less powerful, and coalitions will act to counterbalance the technological advantage of the frontrunner. (paraphrased)

Proponents of this view often cite Paul Christiano in support. Last week I heard him say he thinks the future will be "like the Industrial Revolution but 10x-100x faster."

In this post, I assume that Paul's slogan for the future is correct and then nevertheless push back against the view above. Basically, I will argue that *even if* the future is like the industrial revolution only 10x-100x faster, there is a 30%+ chance that it will involve a single AI project (or a single AI) with the ability to gain a decisive strategic advantage, if they so choose. (Whether or not they exercise that ability is another matter.)

Why am I interested in this? Do I expect some human group to take over the world? No; instead what I think is that (1) an unaligned AI in the leading project might take over the world, and (2) A human project that successfully aligns their AI might refrain from taking over the world even if they have the ability to do so, and instead use their capabilities to e.g. help the United Nations enforce a ban on unauthorized AGI projects.

National ELO ratings during the industrial revolution and the modern era

In chess (and some other games) ELO rankings are used to compare players. An average club player might be rank 1500; the world chess champion might be 2800;

computer chess programs are even better. If one player has 400 points more than another, it means the first player would win with ~90% probability.

We could apply this system to compare the warmaking abilities of nation-states and coalitions of nation-states. For example, in 1941 perhaps we could say that the ELO rank of the Axis powers was ~300 points lower than the ELO rank of the rest of the world combined (because what in fact happened was the rest of the world combining to defeat them, but it wasn't a guaranteed victory). We could add that in 1939 the ELO rank of Germany was ~400 points higher than that of Poland, and that the ELO rank of Poland was probably 400+ points higher than that of Luxembourg.

We could make cross-temporal fantasy comparisons too. The ELO ranking of Germany in 1939 was probably ~400 points greater than that of the entire world circa 1910, for example. (Visualize the entirety of 1939 Germany teleporting back in time to 1910, and then imagine the havoc it would wreak.)

Claim 1A: If we were to estimate the ELO rankings of all nation-states and sets of nation-states (potential alliances) over the last 300 years, the rank of the most powerful nation-state at at a given year would on several occasions be 400+ points greater than the rank of the entire world combined 30 years prior.

Claim 1B: Over the last 300 years there have been several occasions in which one nation-state had the capability to take over the entire world of 30 years prior.

I'm no historian, but I feel fairly confident in these claims.

- In naval history, the best fleets in the world in 1850 were obsolete by 1860 thanks to the introduction of iron-hulled steamships, and said steamships were themselves obsolete a decade or so later, and then *those* ships were obsoleted by the Dreadnought, and so on... This process continued into the modern era. By "Obsoleted" I mean something like "A single ship of the new type could defeat the entire combined fleet of vessels of the old type."
- A similar story could be told about air power. In a dogfight between planes of year 19XX and year 19XX+30, the second group of planes will be limited only by how much ammunition they can carry.
- Small technologically advanced nations have regularly beaten huge sprawling empires and coalitions. (See: Colonialism)
- The entire world has been basically carved up between the small handful of most-technologically advanced nations for two centuries now. For example, any of the Great Powers of 1910 (plus the USA) could have taken over all of Africa, Asia, South America, etc. if not for the resistance that the other great powers would put up. The same was true 40 years later and 40 years earlier.

I conclude from this that *if* some great power in the era kicked off by the industrial revolution had managed to "pull ahead" of the rest of the world more effectively than it actually did--30 years more effectively, in particular--it really would have been able to take over the world.

Claim 2: If the future is like the Industrial Revolution but 10x-100x faster, then correspondingly the technological and economic power granted by being 3 - 0.3 years ahead of the rest of the world should be enough to enable a decisive strategic advantage.

The question is, *how likely is it that one nation/project/AI could get that far ahead of everyone else?* After all, it didn't happen in the era of the Industrial Revolution. While we did see a massive concentration of power into a few nations on the leading edge of technological capability, there were always at least a few such nations and they kept each other in check.

The "surely not faster than the rest of the world combined" argument

Sometimes I have exchanges like this:

- *Me:* Decisive strategic advantage is plausible!
- *Interlocutor:* What? That means one entity must have more innovation power than the rest of the world combined, to be able to take over the rest of the world!
- *Me:* Yeah, and that's possible after intelligence explosion. A superintelligence would totally have that property.
- *Interlocutor:* Well yeah, *if* we dropped a superintelligence into a world full of humans. But realistically the rest of the world will be undergoing intelligence explosion too. And indeed the world as a whole will undergo a faster intelligence explosion than any particular project could; to think that one project could pull ahead of everyone else is to think that, prior to intelligence explosion, there would be a single project innovating faster than the rest of the world combined!

This section responds to that by way of sketching how one nation/project/AI might get 3 - 0.3 years ahead of everyone else.

Toy model: *There are projects which research technology, each with their own "innovation rate" at which they produce innovations from some latent tech tree. When they produce innovations, they choose whether to make them public or private. They have access to their private innovations + all the public innovations.*

It follows from the above that the project with access to the most innovations at any given time will be the project that has the most hoarded innovations, even though the set of other projects has a higher combined innovation rate and also a larger combined pool of accessible innovations. Moreover, the gap between the leading project and the second-best project will increase over time, since the leading project has a slightly higher rate of production of hoarded innovations, but both projects have access to the same public innovations

This model leaves out several important things. First, it leaves out the whole "intelligence explosion" idea: A project's innovation rate should increase as some function of how many innovations they have access to. Adding this in will make the situation more extreme and make the gap between the leading project and everyone else grow even bigger very quickly.

Second, it leaves out reasons why innovations might be made public. Realistically there are three reasons: Leaks, spies, and selling/using-in-a-way-that-makes-it-easy-to-copy.

Claim 3: Leaks & Spies: I claim that the 10x-100x speedup Paul prophecies will not come with an associated 10x-100x increase in the rate of leaks and successful spying. Instead the rate of leaks and successful spying will be only a bit higher than it currently is.

This is because humans are still humans even in this soft takeoff future, still in human institutions like companies and governments, still using more or less the same internet infrastructure, etc. New AI-related technologies might make leaking and spying easier than it currently is, but they also might make it harder. I'd love to see an in-depth exploration of this question because I don't feel particularly confident.

But anyhow, if it doesn't get much easier than it currently is, then going 3 years to 0.3 years without a leak is possible, and more generally it's possible for the world's leading project to build up a 0.3-3 year lead over the second-place project. For example, the USSR had spies embedded in the Manhattan Project but it still took them 4 more years to make their first bomb.

Claim 4: Selling etc. I claim that the 10x-100x speedup Paul prophecies will not come with an associated 10x-100x increase in the budget pressure on projects to make money fast. Again, today AI companies regularly go years without turning a profit -- DeepMind, for example, has never turned a profit and is losing something like a billion dollars a year for its parent company -- and I don't see any particularly good reason to expect that to change much.

So yeah, it seems to me that it's totally possible for the leading AI project to survive off investor money and parent company money (or government money, for that matter!) for five years or so, while also keeping the rate of leaks and spies low enough that the distance between them and their nearest competitor increases rather than decreases. (Note how this doesn't involve them "innovating faster than the rest of the world combined.")

Suppose they could get a 3-year lead this way, at the peak of their lead. Is that enough?

Well, yes. A 3-year lead during a time 10x-100x faster than the Industrial Revolution would be like a 30-300 year lead during the era of the Industrial Revolution. As I argued in the previous section, even the low end of that range is probably enough to get a decisive strategic advantage.

If this is so, why didn't nations during the Industrial Revolution try to hoard their innovations and gain decisive strategic advantage?

England actually did, if I recall correctly. They passed laws and stuff to prevent their early Industrial Revolution technology from spreading outside their borders. They were unsuccessful--spies and entrepreneurs dodged the customs officials and snuck blueprints and expertise out of the country. It's not surprising that they weren't able to successfully hoard innovations for 30+ years! Entire economies are a lot more leaky than AI projects.

What a "Paul Slow" soft takeoff might look like according to me

At some point early in the transition to much faster innovation rates, the leading AI companies "go quiet." Several of them either get huge investments or are nationalized and given effectively unlimited funding. The world as a whole continues to innovate, and the leading companies benefit from this public research, but they hoard their own innovations to themselves. Meanwhile the benefits of these AI innovations are starting to be felt; all projects have significantly increased (and constantly increasing) rates of innovation. But the fastest increases go to the leading project, which is one year ahead of the second-best project. (This sort of gap is normal for tech projects today, especially the rare massively-funded ones, I think.) Perhaps via a combination of spying, selling, and leaks, that lead narrows to six months midway through the process. But by that time things are moving so quickly that a six months' lead is like a 15-150 year lead during the era of the Industrial Revolution. It's not guaranteed and perhaps still not probable, but at least it's reasonably likely that the leading project will be able to take over the world if it chooses to.

Objection: What about coalitions? During the industrial revolution, if one country did successfully avoid all leaks, the other countries could unite against them and make the "public" technology inaccessible to them. (Trade does something like this automatically, since refusing to sell your technology also lowers your income which lowers your innovation rate as a nation.)

Reply: Coalitions to share AI research progress will be harder than free-trade / embargo coalitions. This is because AI research progress is much more the result of rare smart individuals talking face-to-face with each other and much less the result of a zillion different actions of millions of different people, as the economy is. Besides, a successful coalition can be thought of as just another project, and so it's still true that one project could get a decisive strategic advantage. (Is it fair to call "The entire world economy" a project with a decisive strategic advantage today? Well, maybe... but it feels a lot less accurate since almost everyone is part of the economy but only a few people would have control of even a broad coalition AI project.)

Anyhow, those are my thoughts. Not super confident in all this, but it does feel right to me. Again, the conclusion is not that one project will take over the world even in Paul's future, but rather that such a thing might still happen even in Paul's future.

Thanks to Magnus Vinding for helpful conversation.

Algorithmic Similarity

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

The Problem

If someone asked you and me to each write a python program to solves some novel problem then it wouldn't be surprising if our solutions looked quite different. If we then tried explaining to the other person the way our own program worked, one thing that might happen is that we would realize that in spite of surface differences in the way that we wrote it the solutions were actually quite similar. We might have called variables different names, done some steps in a different order and so on, but both programs worked for the same reasons.

Another thing that might happen is that even after really understanding how the other persons program worked the solutions still seemed different. Maybe each program would seem to rely on different mathematical facts. Maybe you would realize that the way I saw the problem was fundamentally different from the way you saw it, you'd never thought about it like that before, and so the approaches I considered were in a totally different regime from the ones you considered. Yours might run asymptotically faster than mine, in a way such that even if I tried to cut out redundancies and optimize my algorithm I could never make it run as fast as yours without fundamentally changing how it worked. Mine might generalize better in some direction, whereas yours might not generalize as well or maybe generalize in a different direction.

This is the kind of intuition that a formal theory of algorithmic similarity should seek to capture. Given two Turing machines that compute the same function it should be able to tell us to what degree they use the same or different algorithms. Exactly how course or fine-grained the classification should be is unclear, and maybe different versions could work with different levels of detail, but it should capture some of our intuitive notions of algorithmic similarity, e.g. that changing the name of a variable generally doesn't change the algorithm, whereas there are fundamental differences between the sieve of Erathostenes and the AKS primality test.

A different problem that might be the same problem or might be a different but related problem is the question of whether one program uses/contains another program.

A Variant

Consider a simple program that outputs 1 if x>y and else outputs 0. Call this program F1. And lets suppose that I know how to compute x-y when x>y or x=y. Then I might write a program that computes the absolute difference of two natural numbers x,y in the following way:

def P1(x,y):

- if F1(x,y)==1

    - output x-y
- if F1(x,y)==0
    - output y-x

This specific program clearly makes use of F1. But lets say we had a different program F2 that computed the same function as F1 but in a fundamentally different way. Then we could define:

def P2(x,y):

- if F2(x,y)==1
    - output x-y
- if F2(x,y)==0
    - output y-x

Now, in some ways P1 and P2 look the same, since they both follow the following general outline:

- if x>y
    - output x-y
- else
    - output y-x

Yet P1 contains F1 whereas P2 does not. Intuitively, if it turned out that the algorithm in F1 didn't actually work as we thought and so didn't compute the function we thought it did, then P2 would still work the same but P1 would be broken.

The general question we would like the theory to answer is thus: given two algorithms, determine whether (and maybe to what degree) one contains the other.

Notice that even if an algorithm contains F1 this might be much harder to spot than in the above example of P1, especially if we allow it to use trivial variations of F1 and still have it count. When humans write programs they explicitly write them in a modular form so that it is easy to see how they work and what components they contain, but in cases such as algorithms created by gradient descent, figuring out how they work and which components they contain can be much harder.

Notice also that this is a question at the level of algorithms not functions. You can't just say that absolute difference contains F1 whereas addition does not, even if the most natural way to make an absolute difference algorithm contains F1 and most addition algorithms really shouldn't have need of F1.

Finally, though P1 and P2 used different algorithms to compute the greater-than-function they both still used some algorithm to compute the greater-than-function, and so we might be interested in the more general question of whether a given program contains the greater-than-function at all, irrespective of whether it uses F1, F2 or something else entirely.

In a similar vein, and more related to the first question of algorithmic similarity, even though at some level of resolution P1 and P2 are different since one uses F1 and the other F2, we would still like our theory to recognize that the overall strategy of P1 and P2 were the same, and to differentiate that strategy with other programs that might use completely different strategies, even ones that also compute the absolute difference .

So this is all well and good and technical. But of course the *real* problem is about philosophy and metaphysics.

The Real Problem

Mathematics is uncannily useful at describing the world. In fact a lot of the great successes of science has been to find good ways of describing parts of the world using math, whether it be general relativity describing gravity, information theory giving the woolly concept of information a precise meaning or encoding complicated real world planning problems into number crunching problems that we can just feed into a computer to get the optimal answer out.

And even if you wont go as far as proclaiming that the world simply is math, you might still believe that there is nothing in the world that can't be explained by math, that there exists such a complete mathematical description of the universe such that for every property of the real world there is a corresponding mathematical property concerning this description of the world.

What does this have to do with algorithmic similarity? Assume the strong Church-Turing thesis i.e. that the whole universe is computable. Then a natural question to ask is, which Turing machines simulate the universe and which doesn't. This question seems like it should have an answer, some Turing machines just run to the left as fast as possible, those don't seem to simulate the universe, whereas others might encode the state of the universe on their tape and then gradually change the tape content according to the laws of physics (it would probably have been more natural to think in terms of cellular automaton). And yet, this raises the question of algorithmic similarity. Two Turing machines might use very different encodings of the world state, how do I tell that they are really computing the same universe? And on the other hand, one Turing machine might just be doing some arbitrary thing that looks complicated but is actually meaningless, how do i rule out the claim that it is just simulating the universe in an non-obvious way?

Consider now the question of whether there is diamond in the universe. For a given Turing machine simulating a universe similar to ours this question should have an answer (an answer for every time step or something). Yet in general, this seems very hard to answer. Though diamond is a natural category in the world, it might not be a very natural category in a specific encoding of the world, so even though a Turing machine simulating the universe must contain a piece that is simulating diamond, having a theory that identifies that subcompontent and differentiates it from simulations without that component look like quite the challenge.

But even now that you know the *real* question is about cool philosophy stuff, you might still want to ask; why should I care?

Why You Should Care

Reason 1: Annoying waterfall apologists

The one comes to you, shows you a random waterfall and says: this waterfall is implementing a conscious human being.

What do you say to this person? They are clearly crazy, but how do you convince them of their error or at least make their position look undefendable to bystanders watching the argument?

I can imagine someone understanding minds so well on a technical level that they could build a computer that completely emulates a human mind. And maybe someone skilled enough in the art of hydrocomputation could even do the same on a waterfall. But there is just no way that a arbitrary naturally occurring waterfall is running a human mind, and yet here we are, with waterfall apologists taking over key positions in government and academia, all because we don't have the theory of algorithmic similarity that could have prevented this tragedy.

Reason 2: What is a world model/What does it mean to have true beliefs

True belief is when your map matches the territory. These are words we all hear every morning when we recite our morning prayer to rationality. But what does it actually mean? Intuitively, there is some thing in the world, and there is some thing in your brain, and then there is some relation between the thing in the world and the thing in your brain such that the thing in your brain is "like" the thing in the world and you can make predictions about how the thing in the world would react merely by doing stuff to the thing in your brain. To explain what that relation is and what "like" means we need a theory of algorithmic similarity. (I know the point of The Simple Truth was not to ask too many questions, but honestly, how *are* the pebbles "like" the sheep??)

Reason 3: We need it for this very unrealistic AI design

Consider this very unrealistic AI design. It's got at great model of the world inside its head. It considers sequences of actions, for each sequence computes the different states of the world that could follow from that sequence and how likely they each are. Then it runs a pre-programmed function on the hypothetical world states to compute the amount of value, e.g. diamonds, in each state of the world, weights each world by its likelihood and chooses the action sequence that would in expectation lead to the most value.

Forgetting the complexity of human values for the moment, how does the function count the amount of value in the representation of a world state in the AI's head? Algorithmic similarity. But wait, you might say, what if we fix the way the AI represents world states such that it's easy to look inside? This might fix part of the problem (see Ontological Identification), but we might not want to fix the AI's representation like that, maybe we want it to learn the best world model it can without constraints, but in that case we probably need a better understanding of how to look inside of computational objects.

Even disregarding values, when the AI has found a world model that yields good predictions we want it to be able to look inside that model and do stuff like count the amount of diamonds in it, or find out other physical facts about it, even if it found that model just by looking for computational objects that predicted its sensory input.

Reason 4: It could give us a better agent/optimizer criterion

In Risks From Learned Optimization the authors write: "whether a system is an optimizer is a property of its internal structure—what algorithm it is physically implementing—and not a property of its input-output behavior ". In this case the usefulness of a theory of algorithmic similarity is obvious. Algorithmic similarity could give us a framework with which to talk about classifications of exactly this nature.

Reason 5: Because FDT needs it

FDT reasons by imagining that the mathematical function making its decisions have different outcomes. But the question then arises of how FDT determine what things in the universe are implementing her decision algorithm. This seems like an ideal situation for a theory of algorithmic similarity to bring some clarity. We need a theory of how similar two algorithms should be for FDT to change both of them, when one algorithm contains FDT's decision algorithm so that changing her own means changing part of the other and what it means for part of the universe to implement a decision algorithm at all.

Reason 6: Deep insights into the nature of reality/Giving computational functionalism the comeback it deserves!

Self-explanatory. (see Computational Functionalism, these guys need help)

Now that you understand the problem and care deeply about solving it the only reasonable thing to do would be to end this blog post and get to work, right?...

Why The Question Doesn't Make Sense And We Shouldn't Expect There To Be A Satisfying Answer

I've been telling you tales about all the nice properties this theory will have and all the confusing situations that it will explain in a perfectly satisfying manner but all this time I was really selling you a mirage, we don't have the thing yet, and so we run the risk of the question itself just not making sense to ask. I think this is a reasonable thing to be afraid of. Intuitions can be misleading and when the thing you are looking for is this great the reason might be because it is just a magical dream.

I don't believe so, but other smart people do. And even if you think the criticisms in the end will turn out to be invalid, they're still important to be aware of now so we know the extent of the challenges that a theory like this would somehow have to overcome.

This will not be en exhaustive list of the problems you could have with the views presented so far. In fact, there will only be one entry to the list and I will definitely not present the strongest case for it. Treat it as an appetizer. I wont even respond to the criticism, hopefully I will get back to that some time, maybe someone else will as well. I don't believe I have any knock-down arguments, that might require the full theory, but I still think there is something to be said in response.

Criticism 1: It's Trivial

Lets say you have some system A, might be an algorithm, some casual graph, some mathematical description of the human mind or maybe the whole universe. I will give an intuitive argument that basically any other physical system can be seen as implementing this system.

First we need to identify the states that A can be in. Lets suppose A only has finitely many states, this should normally be enough since real things are generally thought to be finite, but much of the same argument would work with infinitely many states. Since this is a deterministic system, for each state there will be exactly one state that follows. You can imagine drawing this out as a huge diagram. If you also did this with something like a bucket of water then because of the astronomical amount of molecules in the bucket there would be an almost inconceivable amount of states it could be in, with all sorts of complicated connections between them.

Now the heuristic argument is that there would almost certainly be a way to identify sets of states in the bucket system such that the resulting graph were identical to the graph of system A. This might be a very unnatural coarse-graining, you might end up associating very different physical states in the bucket system and saying that they are the same, but on the other hand, when you say that a physical calculator is implementing addition you are also bundling together a bunch of distinct physical states together to make that argument. And in any case, there should definitely be something disturbing about the statement that if you just look at a bucket of water the right way it can be seen to implement pretty much anything, including a human brain.

You could try to make rules for which ways of associating states are allowed and which aren't, but this seems somewhat unnatural, and in any case it seems hard to make up a set of rules such that you could never game the definition with something like this.

There are better versions of this arguments, versions where you actually construct a trivial physical system that implements A in this way without resorting to "almost certainly", but I wont go into them here. For now I just want to make the remark that triviality results like these will have to be dealt with in some way for any good theory of algorithmic similarity to work, especially if we want it to have all the nice philosophical implications.

Conclusion

If you only remember one thing from this post, let it be this: algorithmic similarity is a cool problem and by thinking about it you can be cool too.

In terms of grand strategies for tackling the problem, well, those will be in the next post, I swear. For now, I think the main thing is to focus on the technical problem. It's interesting, it's pretty concrete and hopefully the philosophical stuff will just fall out of it when we've got the formal theory down.

In terms of work to build of, there should be plenty. For one thing, it seems like some people in proof theory are interested in similar stuff, the way they frame it is "when are different proofs really the same", but there are reasons to believe that the questions might be intimately related (see Curry-Howard correspondence).

There is the debate about computational functionalism in the philosophy literature, which doesn't look at the technical side of the problem as much, at least not in the way I presented it here (or so I believe), but have considered a lot of the philosophical implications.

Finally, computational complexity theory has obviously done a lot of work that seems to be in this area, since the complexity classes are examples of classifications of programs that distinguish between different programs which compute the same function.

My impression is that a surprising amount of people and areas of thought have been in contact with pieces of this problem, and I think it would be great to try and gather as much as possible of their work in one place. I would like to compile a giant reading list/archive of stuff related to algorithmic similarity, and I hope that some of you reading this post will help by contributing links to work or ideas that you have stumbled upon and think might be relevant.

Another thing that might be useful is to compile a list of problems where for each problem there are multiple algorithms that solve the same problem in what, at least initially, seems to be different ways. I don't have many concrete examples yet but it should be relatively easy to find some. Whether or not people can agree on whether two algorithms are "actually" different or "really" just doing the same thing is another question, I predict that there might be quite a lot of disagreement on that, but that discussion might itself be useful for clarifying some intuitions. Especially interesting would be examples of algorithms having the same asymptotic run-time that nevertheless ran clearly different algorithms.

Here is the link to the reading list, I have got some stuff already, and if you post a suggestion in the comments for something that you think should be in there, then I will add it in:

https://docs.google.com/document/d/1cC9H2sBd09OUYroo9YMFg2Gc438PEvyxnBO1PNxUZkA/edit?usp=sharing

Related documents:

https://drive.google.com/drive/folders/1wdxLcb7nTLYl1byycXIE9tNm3G5jNaPG?usp=sharing

Thanks for reading!

Thoughts on Retrieving Knowledge from Neural Networks

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Disclaimer: this post is a collection of preliminary thoughts around a specific question for MSFP'19 blogpost writing day. I do not intend to present novel results or define a research agenda.

1. Motivation

The capital of Mozambique is Maputo. This is a fact about the world I learned in school many years ago. Some part of my brain must have changed in order to store this fact and allow me to remember it today. It may be possible, *in principle*, to recover this fact by simply looking at my brain.

In this post, I consider an analogous topic in the context of Artificial Intelligence: *is it possible to retrieve the knowledge stored in a neural network?* This line of enquiry could be a step towards more transparent AI.

2. Query-Based Network Analysis

Suppose we train a RL agent in the virtual sandbox environment Minetest. In this environment, agents can use tools called pickaxes to efficiently "mine" and collect objects made of stone. A pickaxe is an example of a tool. Tools have some properties that distinguish them from other items (e.g. tools cannot be "stacked" in the inventory).

We may try to retrieve the knowledge represented in the internal state of the agent by means of queries. For instance: at which point, during training, does the RL agent learn that *a pickaxe is a tool?*

2.1 Clarifying the Query

It seems plausible to me that this question could be asked "as it is" by an observer in order to gauge the agent's understanding of its environment. That is to say, at the moment of posing the question, it seems perfectly meaningful. However, further thought reveals some vagueness in it: what *exactly* is a pickaxe? What *exactly* is a tool? *What does it really mean for a pickaxe to be a tool?* It is funny how easily one can become confused with words like "really" and "exactly".

In this example, it seems plausible to interpret the question in reference to the source code of the Minetest engine. Let us assume, for the sake of the argument, that the source code uses an object-oriented approach, and contains a *Tool* class, and a *Pickaxe* class which is a subclass of the former. The fact "a pickaxe is a tool" can be recast as a claim that the Pickaxe class implements all relevant aspects of the Tool class.

When we ask similar queries for agents that learn in the real world (e.g. "when has a system learned the fact that the capital of Mozambique is Maputo?"), formalising the query is not as straightforward. A possible solution to this issue could be to introduce ontologies (in the sense used in information science). An ontology defines formally the properties, meaning, and relations between categories and entities in a particular domain. Such ontologies could be used to model relevant aspects of a target domain and used analogously to the source code in the previous example. However, the addition of a new "layer" between the real-world domain and the query can be a source of problems if the ontology fails to capture all relevant aspects of the domain.

2.2 Two Approaches to Query Answering

Once the query has been formally defined, I find at least two approaches to answering it. The first is to define a battery of tests to gather evidence that all relevant features or aspects of the fact have been learned by the agent. For instance, if we observe that the agent has learned not to stack pickaxes, we may conclude that this particular aspect of the Tools class has been learned. However, this approach quickly runs into trouble, since it seems very hard to ensure that the designed suite of tests captures all relevant aspects of the query. For instance, an agent may not attempt to stack iron pickaxes, but perhaps it would attempt to stack bronze pickaxes, if it found them. Furthermore, this approach does not entirely satisfy the original desideratum of retrieving knowledge directly from the network, because it depends on tests (and the number of required tests is potentially very large).

A second approach is to attempt to define a structure-preserving mapping between the learned model of the network and the representation of the Pickaxe and Tool classes. This approach is more exhaustive than the previous and does not require tests. However, the questions of how to define such mapping, and what kind of structure should be preserved by it, become the crux of the whole problem.

The knowledge representation analysis described in [1] can be roughly seen as an example of this method. This article describes RL agents trained to play a "capture the flag" game in a 3D environment. In order to study the ways in which knowledge is represented in these systems, a simple "ontology" consisting of 200 Boolean facts is defined. To test whether these facts have been learned, a classifier is used in order to decode these facts from the agent's recurrent hidden state. The classifier is trained with input data labelled manually, according to whether each fact holds in the environment. If the classifier can "decode" a fact with sufficient accuracy, the fact is deemed to have been "learned". Effectively, the classifier is implementing the "mapping" between the state of the network and the ontology-based query, and the "structure" to be preserved is the sufficient separability between states where the fact holds, and states where it does not. The analysis verified that the number of learned facts increases monotonically in time during training, and discovered that some facts are represented by single-neutron responses.

2.3 A Challenge for the Query-Based Approach

In addition to all the difficulties already mentioned, retrieving knowledge from a neural network by means of queries (ontology-based or otherwise) can be affected by a severe problem:

> ***Sarge:*** *You mean you could have turned the bomb off any time!? Why didn't you tell us? And don't say I didn't—*
> ***Gary: You didn't ask.***
> ***Sarge:*** **growls in irritation**

— *Red vs. Blue*

If the knowledge stored in a network is probed via a set of queries, it might be the case that much of it is not revealed due to the fact that crucial questions are simply not asked. This can be particularly problematic in applications where a the neural network must be completely transparent.

3. Comprehensive Analysis via Pattern Recognition

To prevent the problem discussed in the previous section, we would like to ensure we retrieve *all knowledge* represented in the network. However, it is unclear what this means. In some way, we already know everything the network "knows," since all the values of all parameters of a network are accessible at any time during training. However, these parameters do not describe what the network "knows" in a way that is intelligible to a human. This is analogous to receiving a picture as a list of pixels and RGB colours, and being asked about its contents. What we need is a way to process the state of a network in a way that is both ***exhaustive*** and ***meaningful to a human***.

The kind of process that would solve this problem would need to look at the function implemented by a network at a particular step during training, and recognise in it patterns that are intelligible to us, such as abstract algorithms, structures, ontologies, or processes. In some cases, this process would need to reveal more than just the knowledge stored in the network, as it may be hard to extricate it from other aspects of the learned model, like action policies or evaluation functions. The question of how to define and execute such process is far from trivial.

However, even if we had a process like this, the approach based on pattern-recognition may still fail. Imagine, for instance, a world similar to ours where a sophisticated neural network was trained to predict the motion of celestial bodies. Suppose the network learned and stored in it the laws and concepts of General Relativity. Next, suppose that it was possible to analyse the contents of the network using some form of comprehensive pattern-recognition approach, as described above. I would expect the process to the retrieve a formal representation of GR from the network. However, imagine that the physicists in this world had not yet discovered GR, and mathematicians had not yet developed the types of tools required to formulate the theory. In such a world, it seems hard to see how the work of the network could be interpreted in a way that was intelligible to humans. Even if we had a perfect process that reliably extracted all the higher-level structure encoded in the network, the ontologies uncovered by it may be impossible to understand. Similarly, it is possible that sufficiently advanced neural networks in our world may represent knowledge in terms of ontologies that are incomprehensible for us.

4. Conclusion

Retrieving knowledge represented in a neural network is a challenging task. Two approaches for addressing this problem have been discussed: a query-based approach, and a pattern-recognition approach. The first approach queries the network using a formal ontology, while the second processes the model learned by a network in order to output high-level ontologies. The first approach seems more tractable than the second, but it is also more limited, since some knowledge may go undetected due to our failure to ask relevant questions. The second approach can also be limited by the fact that it may yield knowledge which is incomprehensible for humans.

---

[1] Jaderberg, M., Czarnecki, W.M., Dunning, I., Marris, L., Lever, G., Garcia Castañeda, A., Beattie, C., Rabinowitz, N.C., Morcos, A.S., Ruderman, A., Sonnerat, N., Green, Tim., Deason, L., Leibo, J.Z., Silver, D., Hassabis, D., Kavukcuoglu, K. and Graupel, T. 2019. Human-level performance in 3D multiplayer games with population-based reinforcement learning. Science, Vol. 364, Issue 6443, pp. 859-865.

Parables of Constraint and Actualization

I have an intuition that physics, logic, and computation are all closely connected, and that their bounded versions are also closely connected. I think this intuition started as wishful thinking. I didn't have time to study all three of my favorite subjects, so I hoped that studying one would eventually lead to all the others. The rest of this post aims to convince you (and me) that it wasn't *just* wishful thinking.

Imagine you live in the world of mathematics. The things around you are groups and manifolds and numbers. You can't touch anything. You *can* touch symbols floating around the objects, and only by moving these symbols around can you interact with the objects. This is Logic. Eventually you notice that the symbols are made of the same stuff as everything else, and have their own symbols. By thinking hard and manipulating these symbol symbols very carefully, you find a way to get the symbols to manipulate themselves. This is a Computer and with it you interact with the world more easily. *Logic actualizes computers*.

Imagine, in your platonic world, you come across a library which claims to contain proofs of every true fact of arithmetic, and only true facts. You also discover that you can build a Computer out of not just logic, but also the numbers around you. Since it's made of numbers, the library should know about it. You teach the computer to ask the library for proofs that it will finish computing and halt, and for proofs that it will run forever. You also teach the computer to disobey the library. If the library tells the computer it will halt, it should run forever. If the library tells the computer it will run forever, it should halt. If the library gives you any proof, it will have lied. If it doesn't, then there are facts it does not know, so it lied at the beginning. The library fades away, it was an illusion all along. *Computers restrain logic*.

Imagine you grow tired of your platonic world and you want to build a universe with planets and people and everything else. What could you build it out of? The only thing at your disposal is math. You teach your computer what numbers are needed to represent each bit of stuff, and how every bit of stuff will change at every step, then turn it on. You watch as the platonic world of numbers transmutes into the world of physics and stuff. *Computers actualize physics*.

Imagine looking into your universe from the outside. What would you see? If you made the universe infinite, you would see every logically possible configuration of stuff somewhere inside, constrained by the laws of physics you chose. If you made it grow in a certain way, you would expect to see the universe split into different universes like the first, but with different laws of physics. If you built in something like Quantum Mechanics, you would see the universe developing in such a way that pieces stop interacting, each split recording every "decision". Perhaps, if you build it right, and you look far enough, you would see every conceivable (consistent) mathematical structure. *Physics actualizes logic.*

Imagine you taught your universe how to make a particle that knew the entire future of any lightbulb it hits, and scatters to the left if the lightbulb is going to turn on, and scatters right if the lightbulb won't. Consider a turned off lightbulb in a room with a light-switch on its right, and a self-destruct switch on its left. If the lightbulb will turn on, the particle should scatter left and destroy the light before it will ever turn on. Contradiction. If the lightbulb never turns on, the particle should scatter right and turn the light on. Contradiction. The particle fades away, it was an illusion all along. *Logic constrains physics*.

Imagine you live in an actual universe. There are planets and people, and everything else. You had a dream about symbols moving themselves, so you moved some particles in a clever way so that they mirror the computer in your dream. But with actual time you have to wait for your computer to finish thinking, which sometimes takes forever, and sometimes it doesn't have enough particles in the right places for it to mirror all the calculations. These *computers* are much more *constrained by physics,* so using them is much more complex. *Complexity is for computers inside universes*.

Imagine that you are curious. You see planets and people and computers and everything else around you, and you wonder how they work. You do some experiments and discover that there seem to be some rules that everything obeys, rules your computer can learn. Given any set up, could you figure out how it will behave? You can build computers out of the universe, so the rules should know about them. You attach a lightbulb that turns on only if the computer halts, and tell the computer to use the rules to figure out whether any photons leave the lightbulb in this configuration of particles, but you teach the computer to disobey whatever it derives. If it derives photons leaving the lightbulb, it will run forever and keep the lightbulb off. If it sees to the end of time and learns that the lightbulb never turns on, it stops and turns on the light. Both contradictions. Therefore, it can't know its behavior forever in the future, and keeps calculating forever. The laws aren't enough to decide everything quickly. *Bounded Physical Laws are for universes containing computers*.

Imagine you are playing a game with your friend where he gives you a number and you have to factor it. This game takes a really long time for bigger numbers, but you can't just make numbers up to trick your friend because he can check your numbers easily by multiplying. Some problems are easy-**P**easy, some problems are hard. The best games involve problems that seem **N**ot easy, but can be checked easy-**P**easy (**NP**). Logic is a game like this. After all, a proof is easy to check, but often hard to find. Maybe, by some clever trick, all problems which are hard but can be checked easily are actually easy (**P** = **NP**). Then Math would be easy. *Complexity automates Logic (or shows why it can't be automated)*.

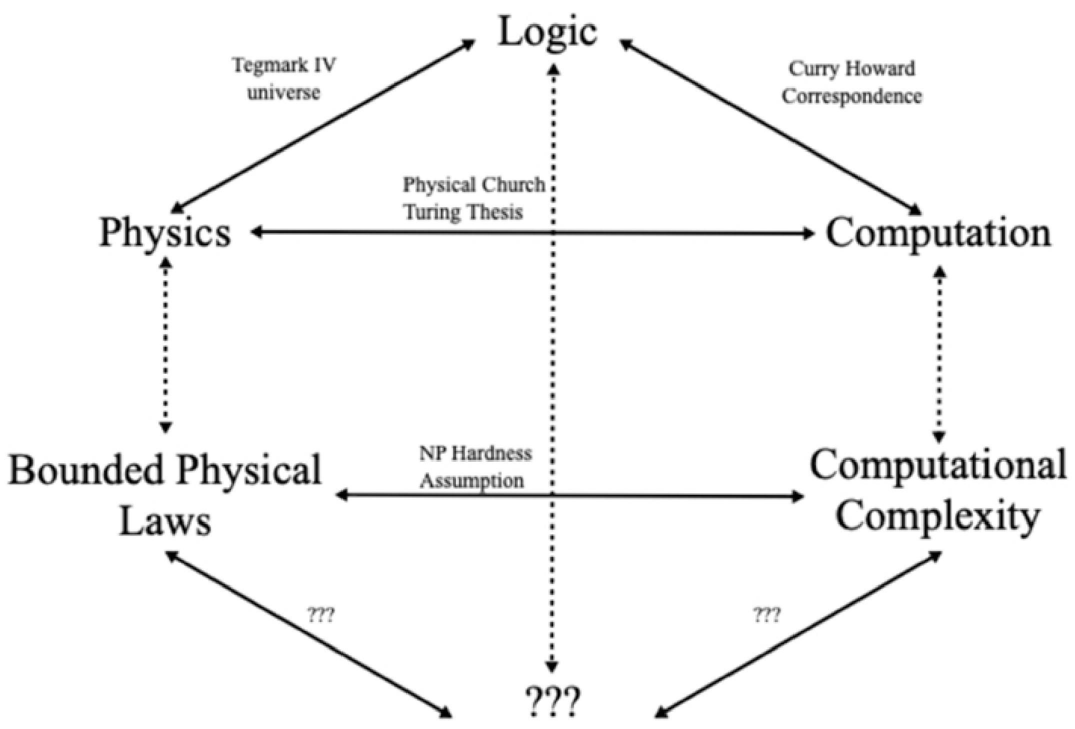
Imagine you found a particle that tries to determine if a lightbulb will light up and scatter left if it will, and right if it won't, but it only has access to limited information about the lightbulb. The particle can only know with probability p that the lightbulb will turn on. To comply with the rules best it can, it will scatter left with probability p and scatter right with probability 1-p. Consider a turned off lightbulb in a room with a light-switch on its right, and a self-destruct switch on its left. If it scatters left with probability p>50% (or p<50%), then the lightbulb will never turn on with probability (1-p) > 50% (or (1-p) < 50%), a contradiction. The same thing happens if it scatters right. But, if p = 50% exactly, everything stays consistent. The particle scatters left and right randomly. *Bounded Physical Laws preserve Logical Consistency (and randomize in case of inconsistency)*.

Imagine you find that the universe is different than you thought, it actually works on complex numbers and probabilities and "entanglement" and it turns out that you can build something that does computation in a completely different way from your computer. When you use this new machine, it seems like it can play games like factoring much faster than you could before, and you wonder if it can do all hard problems faster. If it can, you can automate math (and maybe physics and art and music and politics and...), which seems just way too powerful, so you expect it can't. If you found even newer laws, and built even stranger computers, would they ever be able to do something like that? Maybe, but it still seems unlikely. Maybe even as *Bounded Physical Laws actualize complexity* (by offering new designs over time), *complexity constrains physics* (By preventing **P** = **NP**).

Thinking Machines live in an actual universe. They act within the constraints of a Bounded Physical Universe, they think within the constraints of Complexity.

We live in an actual universe. We see planets and people and computers and everything else, but the people and the computers look more and more the same, and because we're forcing them to look alike and not understanding why they look alike, we aren't sure how the computers will turn out. And the planets and everything else don't have any say at all, only the people do. Maybe we constrain the computers and the computers actualize us. Or maybe we actualize the computers, and the computers constrain us.

Diagram



Final thoughts:
These parables are pretty contrived. It would be easy to flip around the constraint/actualization directions in a lot of places, and in other places there are much deeper connections than those presented (e.g. Curry-Howard Correspondence between Logic and Computation). The point was to get across my impression that all of these fields are very closely related, and that by considering the relationships between physics and computational complexity (as bounded versions of their idealized forms) you might learn something about the bounded version of idealized Logic, which I wildly speculate to be something helpful to AI Safety, like embedded agency or something. Big thanks to MSFP for connecting me with loads of brilliant people willing to think through crazy ideas, and a special thanks to Mathijs Henquet for drawing the initial form of the diagram above.

Further Reading (not exhaustive, not always related, maybe helpful):

Logic actualizes Computers - Turing's original paper
Computers restrain Logic - Incompleteness Ex Machina by Sebastian Oberhoff
Computers actualize Physics - David Deutsch
Physics actualizes Logic - Tegmark's Universes
Logic constrains Physics - If this isn't true, I don't know what is.

Bounded Physical Laws are for universes containing Computers - Undecidability of the Spectral Gap
Physics constrains Computers and *Complexity is for Computers inside universes* - (The Nature of Computation by Christopher Moore is a standard reference)
Bounded Physical Laws preserve Logical Consistency - Reflective Oracles
Complexity automates Logic and *Complexity and Physical laws are connected* - NP Complete Problems and Physical Reality

When do utility functions constrain?

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

The Problem

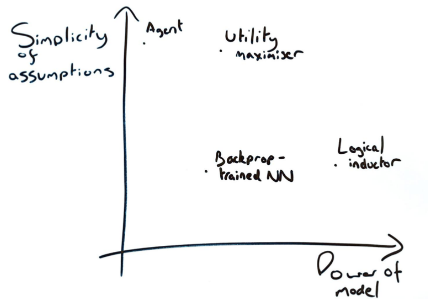
This post is an exploration of a very simple worry about the concept of utility maximisers - that they seem capable of explaining any exhibited behaviour. It is one that has, in different ways, has been brought up many times before. Rohin Shah, for example, complained that the behaviour of everything from robots to rocks can be described by utility functions. The conclusion seems to be that being an expected utility maximiser tells us nothing at all about the way a decision maker acts in the world - the utility function does not constrain. This clashes with arguments that suggest, for example, that a future humanity or AI would wish to self-modify its preferences to be representable as a utility function.

According to Wikipedia's definition, a decision maker can be modelled as maximising expected utility if its preferences over *mutually exclusive outcomes* satisfy completeness, transitivity, continuity and independence. This is neat. However, it still leaves us with free choice in ordering over our set of mutually exclusive outcomes. The more outcomes we have, the less constrained we are by utility maximisation and when we look closely there are often a LOT of mutually exclusive outcomes, if we give ourselves truly free range over the variables in question.

In the context of thought around AI safety, introductions to these axioms have not denied this but added a gloss of something like 'if you have a consistent direction in which you are trying to steer the future, you must be an expected utility maximizer', in this case from Benya's post in 2012. Here the work done by 'consistent direction' seems to imply that you have reduced the world to a sufficiently low dimensional space of outcomes. Though I agree with this sentiment, it seems fuzzy. I hope to add some clarity to this kind of qualification. To put it another way, I want to see what restrictions we need to add to regain something like the intuitive notion of the maximiser and see if it is still sufficiently general as to be worth applying to theoretical discussions of agents.

Pareto Frontier of Concepts

A quick aside: Why do we want to use the idea of a utility maximiser? The explanation that comes to mind is that it is felt to be a concept that lies on the Pareto frontier of a trade-off between the explanatory power of a concept and the lack of troubling assumptions. I doubt this framing is new but I drew up a little diagram anyway.



Explaining Too Much

What I want to do is move from considering utility functions in general to particular classes of utility functions. To begin, let's look at the first example, let's look at a canonical example of utilities being constraining - the Allais paradox. In the paradox it is shown that people have inconsistent preferences over money where the utilities involved are samples of the underlying utility function $U : R \rightarrow R$ where the input is money, for concreteness, at a particular future time t'. In this case most humans can be shown to make decisions not compatible with any preference ordering.

When we construct this scenario we are imagining that people are in identical states when they are offered the various choices. However, we could instead construct these lotteries as being offered at distinct moments in time, and allow preferences over money to vary freely as time goes on. Now we are assuming instead the class of utility functions $U : R^2 \rightarrow R$ where the input is the pair (gain in money at current time t, time). Importantly, a person can still make irrational decisions with respect to their individual utility function, but now there is *no behaviour that can be ruled out by this abstraction.*

The basic point is the obvious one that the class of potential utility functions decides the extent to which the utility function constrains behaviour. Note that even the first class of utility functions over money is extremely permissive - the Allais paradox only works because the possible outcomes in terms of money are identical, but in any scenario where the quantities involved in the lottery are different, one could make any set of decisions and remain free of inconsistency. I may for example have wildly different preferences for having £1, £1.01, £1.02 etc.. It is also important to note that we only need to have one such 'free' variable in our utility function to strip it of its predictive power.

Restricting Function Classes

The next question is whether it is possible to tighten up this theory so that it accords with my intuitive notion of a maximiser, and whether too much is lost in doing so.

One simple way is to reduce possible trajectories to a discrete space of outcomes - I want an aligned AI, I want to be alive in 20 years etc - and view all possible decisions as lotteries over this space. No arguments from me, but we certainly want a more expressive theory.

The example given in Eliezer's Stanford talk, which opens with an argument as to why advanced AIs will be utility maximisers, is that of a hospital trying to maximise the number of lives saved. Here we find the rather unsurprising result that one's preferences can be gainfully modelled as maximising expected utility... if they were already a linear function of some persistent real world variable.

What if we go beyond linear functions though? There certainly seems to be a space for 'near to a goal' functions, like radial basis functions, both in theoretical and practical uses of utility functions. What about cubics? Any polynomials? My instinct is to reach for some kind of notion of smooth preferences over a finite number of 'resources' which persist and remain similarly valued over time. This certainly does a lot of work to recover what we are assuming, or at least imagining, when we posit that something is a utility maximiser.

However, there seem to be nearly maximal examples of non-smoothness that still allow for a class of utility functions that constrain behaviour. For example, take the class of preferences over a rational number in which all I care about is the denominator when the number is in its simplest form. I could for example desire the greatest denominator: $\frac{3}{4} < \frac{5}{9} \sim \frac{1}{9}$. In most situations, were these preferences over my store of dollars for example, this would seem to be outside the class of utility functions that would meaningfully constrain my action, since this function is not at all smooth over the resource in question. However, the are imaginable cases, such as when my action space is to multiply my bank account by a lottery of other rationals, in which a class of utility functions incorporating this sort of information represents a meaningful restriction on my behaviour.

A contrived example? Completely, but what this implies is that the real problem is not strictly in the way in which our function class is able to map 'real' variables to utility but in the way in which our decisions represent a restriction of our future outcome space.

Connecting Function Classes and Decision Spaces

The result is the perhaps obvious point that whether a class of utility functions constrains behaviour in a given situation is function not just of either the class in question or the type of actions available but a question of the extent to which our available actions are correlated with the possible. When this is true for all utility functions in the class we are considering, our utility function becomes capable of constraining behaviour (though there may still be cases where no course of action could not result from a utility function). This is more likely to happen when our objective must be a simple function of persistent real world states but it does not require this simplicity, as the above example shows.

To take an example of how this works in practice: in economics textbooks we will encounter a wide range of utility functions. These will of course determine the way that the agent behaves in the given scenario. More than this though, the very fact that these agents are utility maximisers feels as if it constrains their behaviour, not just because of the specific utility function but in the kind of calculating, maximising reasoning that it enforces. Under this analysis, the reason that it seems this way is that all utility functions that we might expect in such a scenario, whether logarithmic, quadratic, lexical, all are such that any given utility function, our actions will change the amount of utility we expect to be available. This is not because of the class of potential utility functions exactly, but because the action space/utility function class combination are such that the actions and changes in magnitude of available utility are always linked. (In this pedagogical case of course we come to expect this with good reason because the utility function/action space pair are selected to produce interesting interaction.)

What we need to find, for a given agent to be constrained by being a 'utility maximiser' is to consider it as *having a member of a class* of utility functions where the actions that are available to it systematically alter the expected utility available to it - for **all** utility functions within this class. This is a necessary condition for utility functions to restrict behaviour, not a sufficient one. Note that within almost any natural class there will be the degenerate utility function in which all results result in equal utility and therefore all actions are permissible - this must be deliberately excluded to make predictions. It is this notion of *classes* rather than individual utility functions, which saves my post (I hope) from total triviality.

What remains?

All this talk of restricting function classes is fine, and there is much more to say, but we're talking about selecting the class of possible utility functions as a modelling tool! We can restrict the class of functions that their utility function *is allowed* to be a part of, but if decision makers can employ a function from outside our allowed class and laugh at our Dutch books then what use is all this talk, and these abstractions?

Well, a few things. Firstly I hope this leads some people to clearer understanding of what is going on with utility functions - I feel I am now thinking more clearly about them, though only time will tell if this feeling lasts.

Second, when we make statements about utility functions being meaningful for an agent which wants to steer the future in a consistent direction, we have a more direct way of talking about this.

Third, there may be some ways in which such restrictions are natural for interesting classes of agents.

One thing to consider about a system, with respect to how we should model it, is whether we should expect a system to systematically acquire the capacity to control variables that are of relevance to it.

If I understand them correctly, systems which run according to active inference/predictive processing (of which humans may be a member) can be interpreted as maximising the power to predict observed variation (in this case entropy), presumably where variation is measured across a finite set of information streams. This may suggest that these systems naturally converge to methods of behaviour that are well modelled by utility functions of a particular class and so the abstraction of utility functions may regain meaning while still accurately describing such systems.

Sweeping the Floor

So what becomes of our rock that maximises its rock-like actions? Firstly, we can say that the class of utility functions needed to represent these 'decisions' is simply too broad to be consistently correlated with the minimal freedom of 'action' that a rock has, and thus our abstraction has no predictive power. Second, of course, is that thinking about utility in this way emphasizes that utility is a way of representing decisions. and a rock does not make decisions. How do we know? I'm not entirely sure, but it's not a question that can be offloaded to utility theory.

And what of our humans, and AIs, that may wish to modify themselves, in order to become truly consistent utility maximisers? Does this represent a likely happening? I think this may well still be true, for humans at least, but what is going on is not purely the result of utility functions but comes from the fact that humans seem to (a) organize their way of evaluating to situations to simpler outcome spaces so that there is a restriction of the space of utility functions that contains all of the possible ways in which humans value trajectories and (b) is such that all possible functions in this class require humans to take decisions which are consequential the evaluation of this utility function. In some ways this is very similar to the 'steering the future' hypothesis presented at the beginning but hopefully with some further clarification and operationalization of what this means.

My conclusions are getting shakier by the paragraph but it's posting day here at MSFP and so the flaws are left as an exercise to the reader.

Actually updating

Actually updating can be harder than it seems. Hearing the same advice from other people and only ***really*** understanding it the third time (though internally you felt like you ***really*** understood the first time) seems inefficient. Having to give yourself the same advice or have the same conversation with yourself over and over again also seems pretty inefficient. Recently, I’ve had significant progress with ***actually*** causing internal shifts, and the advice.. Well, you’ve probably heard it before. But hopefully, this time you’ll ***really*** get it.

Signs you might not be *actually* updating.

- You do some focusing, and you discover a problem and you talk to a part of yourself and work through it. Then a week later, you find yourself having the same conversation with that part of you.
- You change your mind about doing something but find that suspiciously your behaviour is not changing or doesn’t change for that long.
- Someone says “X” and you go “oh yes, X”. Then, some time passes. Then someone else says “X” and you go “ohhh yes, X, now I get it”. Then, some time passes. Then someone else says “X” and you go “ohhhh yes of course, X, now I really get it”. Etc.
- You feel like you believe X but yet keep finding yourself behaving as though you don’t believe X.
- You feel like your rationalisations are running so deep that you can’t seem to catch them. You're doing things that seem to make sense but then something might happen that makes you feel like your behaviour was actually driven by a 'rationalisation'. To make this clearer, imagine something happens (event Y) that changes your opinion of X. Maybe you realise that the thing 'shouldn't' be much evidence at all. You decide it will only slightly affect your opinion. You may end up with a certain conclusion by looking at old evidence and this thing Y (that you have actively decided to only weight a small amount). Then, something may then happen that makes you ***really*** believe that thing Y is small evidence. Suddenly, you are looking at the same evidence and this thing Y (weighted the same amount as you tried to weight it before), but you have a different conclusion. It's like you are looking at the evidence through a different 'lens'. It seems like you are able to exploit uncertainty around evidence to get differing conclusions depending on this 'lens'. This whole thing might make you feel large internal distrust.

Plausible hypotheses

Plausible hypothesis 1: Some things take longer to digest than other things. Maybe you just need time to actually update models.

Plausible hypothesis 2: If you change a fundamental node in your ‘belief network’, it can be hard to change patterns of behaviour and reactions. You might not believe thing X but behave like you think thing X because you are mostly working on autopilot and habits are hard to break out of. This is especially salient when a piece of actual behaviour is ‘far away’ from the node that has been changed (so that it seems unrelated at a glance).

Plausible hypothesis 3: A lot of the things people are trying to teach you are ‘purple knowledge’. This may mean you may just need lots of gesturing at a thing, or to develop a certain intuition before a certain thing ***actually*** makes sense.

I think it’s likely these hypotheses play at least some role in what is happening. However, in my case something else was playing a larger role.

What was going wrong for me

The hypothesis that seems right for my situation: I was not ***really*** listening to some parts of me. In an attempt to listen to all parts of me, I was doing a few things that would cause the process to fail:

- Calling things ‘biases’. Using words like ‘anxiety’ and ‘perfectionism’ that cause me to box up a part of myself and believe blindly that it’s doing something wrong. Warning flags should arise if you realise that you believe something is wrong for reasons separate to the actual words that part of you is saying. If no matter what a part of you says, you think it’s wrong - you’re being pretty brutal to yourself.
- Saying “I don’t know what the right answer is and should listen to all parts of me” whilst internally feeling like one side of me is obviously going to win
- Looking at a part of myself and believing that it’s trying to help but just thinking that it’s going to be ‘silly’ because it was adapted for a different situation “which definitely doesn’t apply here”.
- Strawmanning a part of myself. Being quick to feel like I’ve understood a part of myself and resolved the problem there. Wanting to solve the problem quickly and just assuming feelings of discomfort afterwards was just ‘residue pain’.
- Deferring to “experts”. Person Y told me X was good and I understand their reasoning, so X must be good. The little part of me that disagrees is just being stupid.
- Using words like ‘weird’ when talking about any disagreement I have with an ‘expert’ and feeling pressure to update quickly (encourages strawmanning a part of myself, and being quick to think I've understood where a part of me is coming from).
- Not focusing enough throughout the process to make sure I’m in actual contact with the part of me that feels a certain way. It might be especially hard to be in touch with the parts of you when talking to an 'expert'.
- Not paying attention to meta parts. Maybe I’m trying to decide between X and Y. I shouldn’t just be paying attention to my feelings about X and Y. It’s good and worthwhile to pay attention to why I’m finding it hard to make this decision. Or why I'm finding it confusing. There may be parts that should be involved in the discussion that are missing.
- Just thinking really hard and not doing experiments. If you are unsure about a tradeoff or if something is true or just a nice story, experiment with it in the real world. Come up with hypotheses and test them.

The thing that changed and allowed me to actually start updating more efficiently was that I ***actually*** started believing that all parts of me are pretty smart. I started believing this because I started actually listening to myself and realised that these parts of me weren’t saying the ‘obviously wrong’ things I thought they were saying. I began to stop just listening to experts and going ‘what they are saying makes sense’ and started having conversations where I just entirely let the part of me that disagreed say all the reasons it disagreed and ‘fight’ the expert. I allowed that part of me to have contact with the world and this meant that part of me could learn. And it worked.

This whole post is something you’ve probably heard before - “listen to all parts of you”, “don’t write the final line”, etc. None of this stuff was new to me, and yet, it feels like a lesson I’ve just learnt. I hope you let the part of you that might think this is all wrong ‘fight’ with me. And hopefully that will cause one of us to ***actually*** update towards the truth.

Understanding understanding

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

How does 'understanding' a problem work, and when do we feel like we understood an explanation or proof? Having an opaque long formal proof often feels insufficient, similarly some arguments feel unsatisfying because they contain many subjectively arbitrary choices.

An explanation is a sequence of claims corroborating a statement. A reasoner understands an explanation (or rather has more understanding of an explanation) when each claim has low surprisal given its mental state after reading all of the previous statements, starting with the problem statement. The aggregate of the surprise over the entire explanation indicates how poorly understood it is. The measure of surprisal is essentially about the reconstructability of the explanation using something like Monte Carlo Tree Search over all possible explanations informed by the mental states of the reasoner.

The surprisal of claims is roughly how long it would take for the reasoner to come up with the claim given its current mental state. Since the space of such claims is exponential in the length of the claim the reasoner has to use some form of attention to guide its search. We can model this attention mechanism by an ontological graph. Such a graph encodes the collection of mental concepts and associative links between them. The mental state of the reasoner is an activation of an ensemble of concepts and the associative links make available other concepts, lowering their surprisal when invoked in the next step of the explanation.

When a step in an explanation is highly surprising some understanding is needed. The reasoner does this by modifying its ontology, creating new concepts or creating associations that make the step more obvious. I call such modifications insights, a good insight gives clarity and makes steps obvious and subjectively trivial.

To examine this situation consider the mutilated chessboard problem[1] and subsequent explanation:

> Suppose a standard 8×8 chessboard has two diagonally opposite corners removed, leaving 62 squares. Is it possible to place 31 dominoes of size 2×1 so as to cover all of these squares?
>
> The answer that it is impossible. Opposite corner squares always have the same color so that there are 30 and 32 white or black squares left. A domino placed on the chessboard will always cover one white square and one black square. So there will always be either two white or two black squares left which can not be covered by a domino.

The problem becomes trivial after making available the idea of attaching the invariant to the problem. We can imagine that the color invariant is activated by the joint activation of the domino and chess nodes in the reasoners ontology.

In this scheme there is nothing stopping a reasoner from encoding solutions to all problems directly into its ontology. This leads to 1) poor generalization/transfer; and 2) a large ontology.

Compression deals with both of these issues, indeed compression is closely related to generalization. General compression works by recognizing and exploiting all detectable patterns with certain computational bounds. Once these patterns are detected they can be extract and recognized in different contexts, which is generalization.[2]

Compressing the ontology has the same general shape as understanding explanations, except this time we want to understand all elements in our ontology given the rest of the ontology. This is related to drop-out: how surprised would I be of this ontological node given that it was not in my ontology. Compressing of the ontology has the nice feature that the ontology becomes more fault tolerant: things that are forgotten but once well understood can be reconstructed from context.

In the example of the mutilated chessboard we can explain the insight by noting that is a general instance of attaching invariant to problems.

I believe there is both pressure to prune nodes with low surprisal (beliefs that do not pay rent) and low activation rates. Very low surprisal nodes can be safely pruned as they are reconstructible from context. On the other hand, nodes with very high surprisal are hard to recall (i.e. get activated) and will also be pruned. This explains why opaque formal proofs don't feel like satisfying explanations, it's hard to fit them into our information retrieval indices.

[1]: https://en.wikipedia.org/wiki/Mutilated_chessboard_problem

[2]: Compression in the colloquial sense only exploits patterns with very low computational complexity

Troll Bridge

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

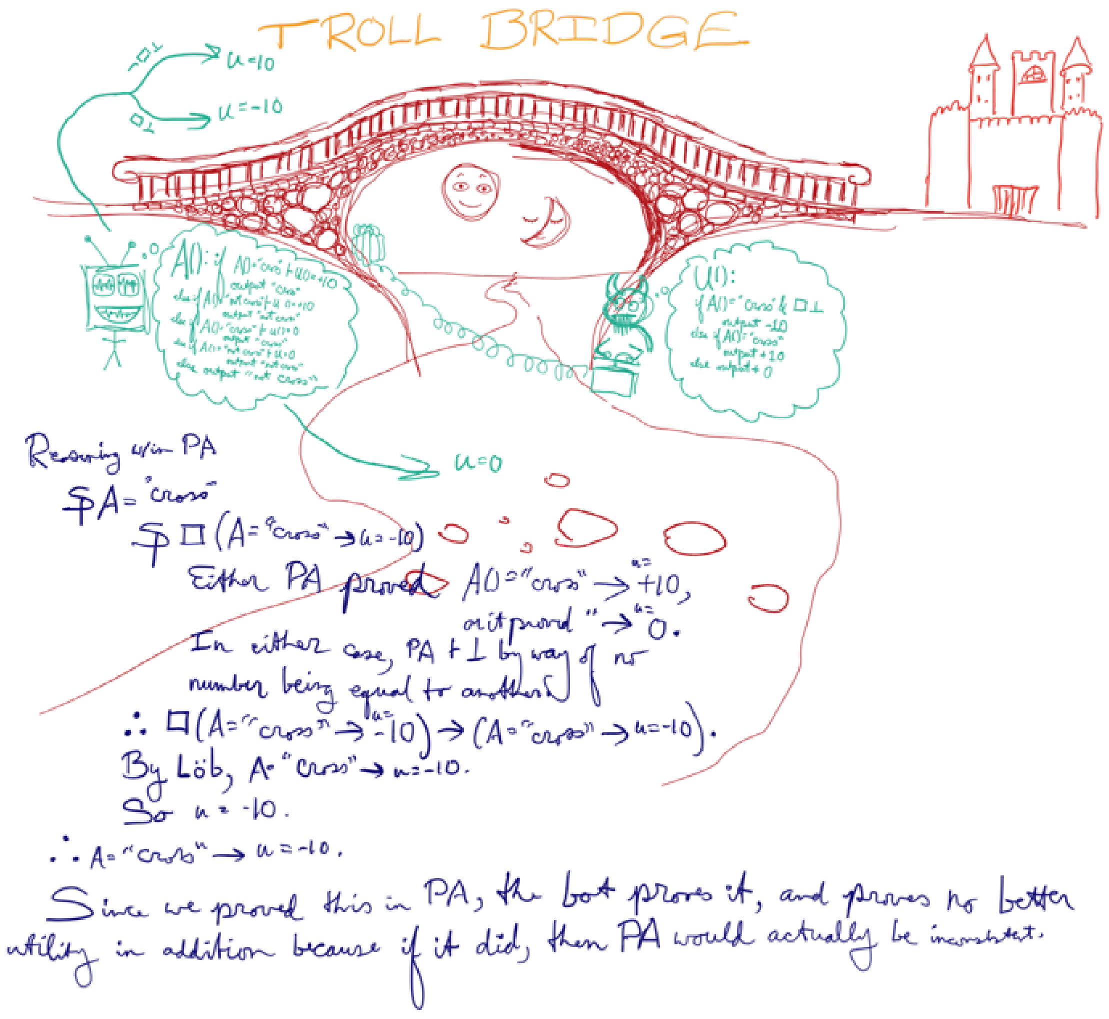
All of the results in this post, and most of the informal observations/interpretations, are due to Sam Eisenstat. I think the Troll Bridge story, as a way to make the decision problem understandable, is due to Tsvi; but I'm not sure.

Pure Logic Version

Troll Bridge is a decision problem which has been floating around for a while, but which has lacked a good introductory post. The original post gives the essential example, but it lacks the "troll bridge" story, which (1) makes it hard to understand, since it is just stated in mathematical abstraction, and (2) makes it difficult to find if you search for "troll bridge".

The basic idea is that you want to cross a bridge. However, there is a troll who will blow up the bridge with you on it, ***if*** (and only if) you cross it "for a dumb reason" — for example, due to unsound logic. You can get to where you want to go by a worse path (through the stream). This path is better than being blown up, though.

We apply a Löbian proof to show not only that you choose not to cross, but furthermore, that your counterfactual reasoning is confident that the bridge would have blown up if you had crossed. This is supposed to be a counterexample to various proposed notions of counterfactual, and for various proposed decision theories.



Reasoning w/in PA

Suppose A = "cross"

Suppose $\Box(A=\text{"cross"} \rightarrow u=-10)$

Either PA proved A()="cross" → u = +10, or it proved " → u = 0.

In either case, PA ⊢ ⊥ by way of no number being equal to another.

$\therefore \Box(A=\text{"cross"} \rightarrow u=-10) \rightarrow (A=\text{"cross"} \rightarrow u=-10)$.

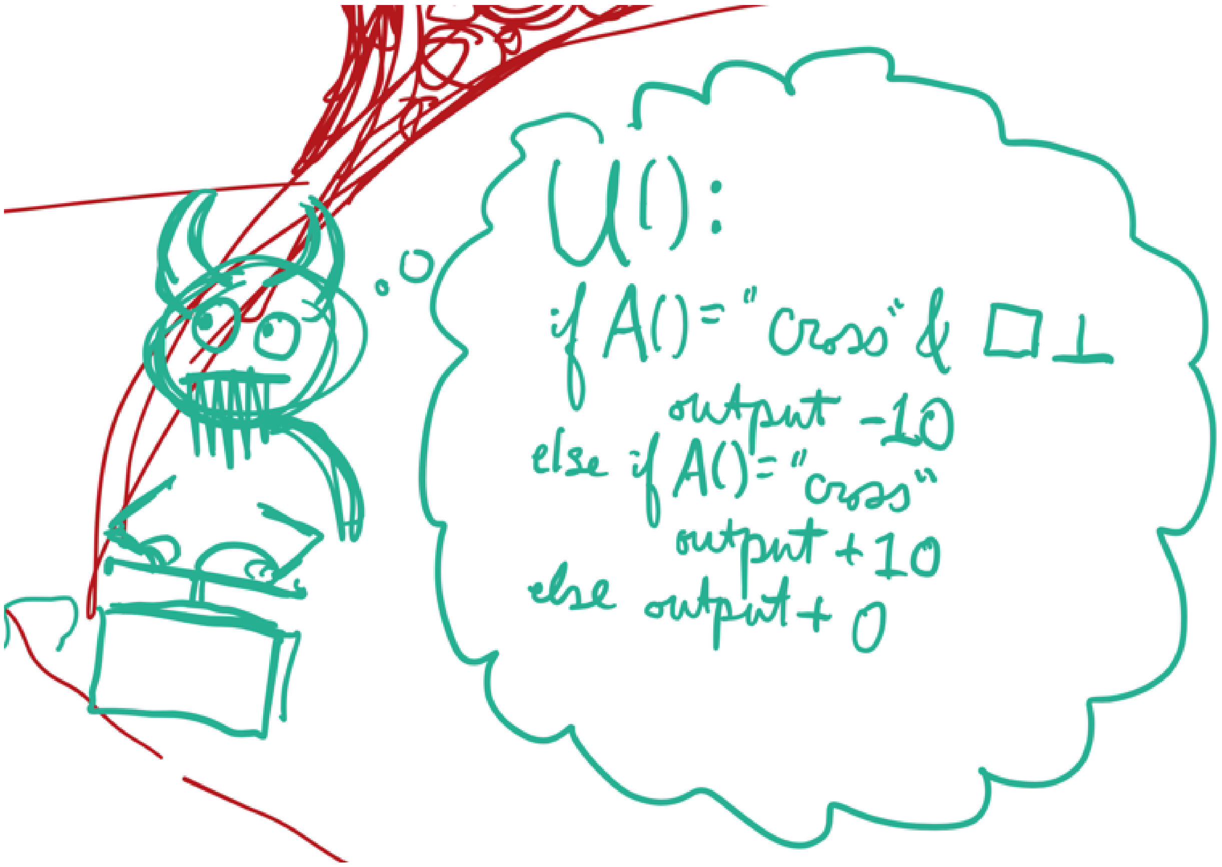
By Löb, A = "cross" → u = -10.

So u = -10.

$\therefore$ A = "cross" → u = -10.

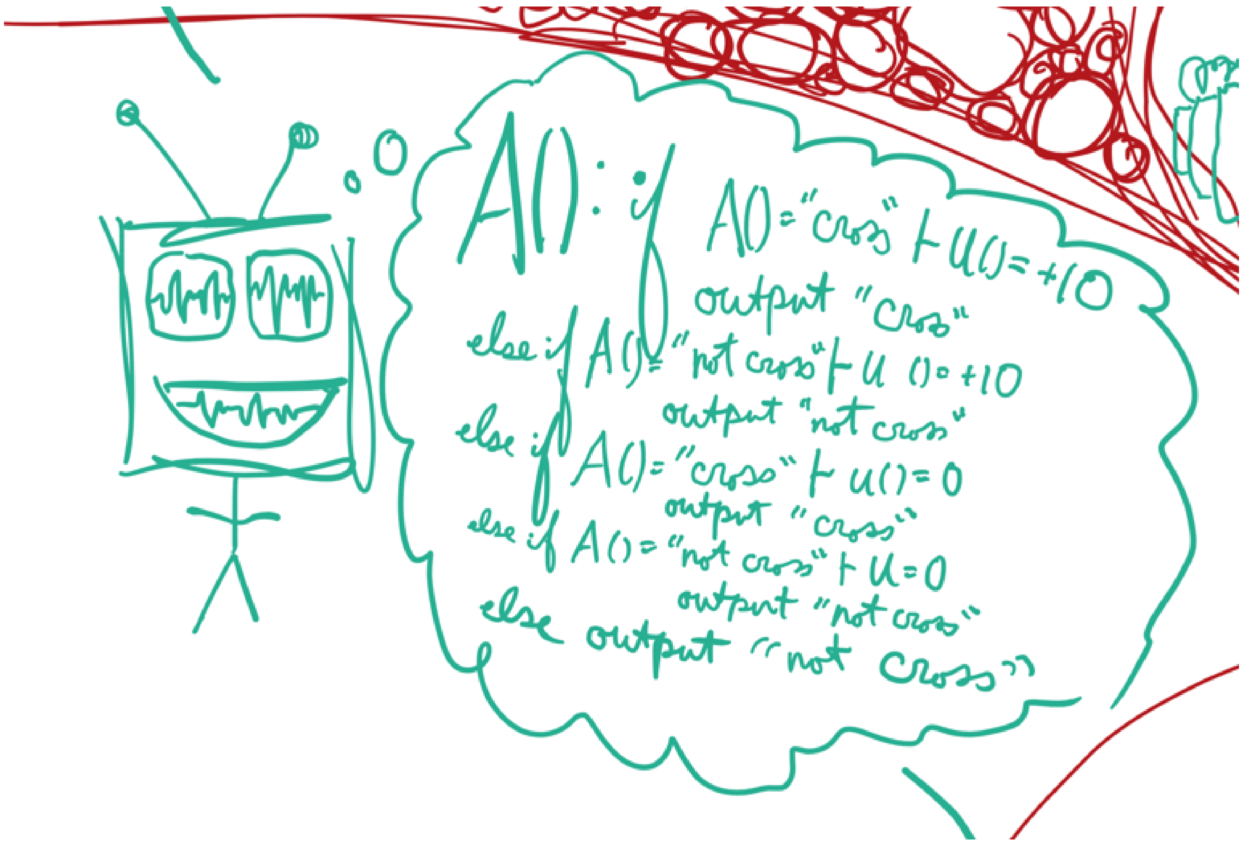
Since we proved this in PA, the bot proves it, and proves no better utility in addition because if it did, then PA would actually be inconsistent.

The pseudocode for the environment (more specifically, the utility gained from the environment) is as follows:



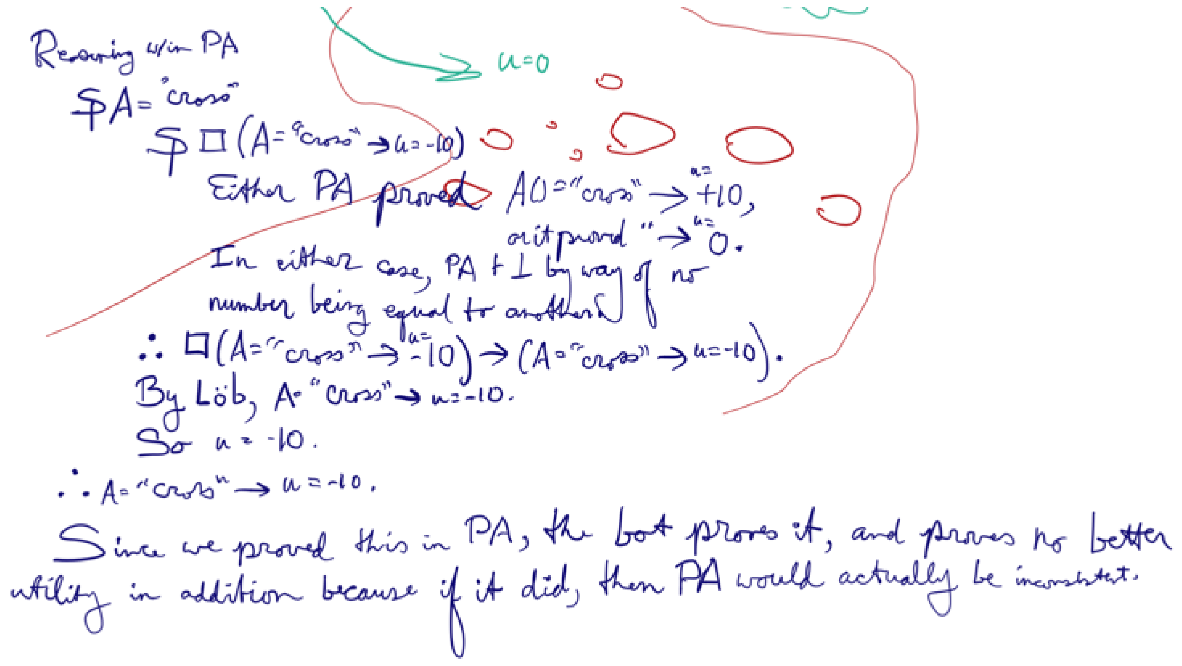
IE, if the agent crosses the bridge and is inconsistent, then U=-10. (□⊥ means "PA proves an inconsistency".) Otherwise, if the agent crosses the bridge, U=+10. If neither of these (IE, the agent does not cross the bridge), U=0.

The pseudocode for the agent could be as follows:


A(): if A()="cross" ⊢ U()=+10
output "cross"
else if A()="not cross" ⊢ U()=+10
output "not cross"
else if A()="cross" ⊢ U()=0
output "cross"
else if A()="not cross" ⊢ U=0
output "not cross"
else output "not cross"

This is a little more complicated, but the idea is supposed to be that you search for every "action implies utility" pair, and take the action for which you can prove the highest utility (with some tie-breaking procedure). Importantly, this is [the kind of proof-based decision theory which eliminates spurious counterfactuals in 5-and-10 type problems](). It isn't that easy to trip up with Löbian proofs. *(Historical/terminological note: This decision theory was initially called MUDT, and is still sometimes referred to in that way. However, I now often call it proof-based decision theory, because it isn't centrally a UDT. "Modal DT" (MDT) would be reasonable, but the modal operator involved is the "provability" operator, so "proof-based DT" seems more direct.)*

Now, the proof:


Reasoning w/in PA

$\vdash A = \text{"cross"}$

$\vdash \Box(A = \text{"cross"} \to u = -10)$

Either PA proved $A() = \text{"cross"} \to u = +10$, or it proved $\text{"} \to u = 0$.

In either case, $PA \vdash \bot$ by way of no number being equal to another.

$\therefore \Box(A = \text{"cross"} \to u = -10) \to (A = \text{"cross"} \to u = -10)$.

By Löb, $A = \text{"cross"} \to u = -10$.

So $u = -10$.

$\therefore A = \text{"cross"} \to u = -10$.

Since we proved this in PA, the bot proves it, and proves no better utility in addition because if it did, then PA would actually be inconsistent.

- Reasoning within PA (ie, the logic of the agent):
    - Suppose the agent crosses.
        - Further suppose that the agent proves that crossing implies U=-10.
            - Examining the source code of the agent, because we're assuming the agent crosses, either PA proved that crossing implies U=+10, or it proved that crossing implies U=0.
            - So, either way, PA is inconsistent -- by way of 0=-10 or +10=-10.
            - So the troll actually blows up the bridge, and really, U=-10.
        - Therefore (popping out of the second assumption), if the agent proves that crossing implies U=-10, then in fact crossing implies U=-10.
        - By Löb's theorem, crossing really implies U=-10.
        - So (since we're still under the assumption that the agent crosses), U=-10.
    - So (popping out of the assumption that the agent crosses), the agent crossing implies U=-10.
- Since we proved all of this in PA, the agent proves it, and proves no better utility in addition (unless PA is truly inconsistent). On the other hand, it will prove that not crossing gives it a safe U=0. So it will in fact not cross.

The paradoxical aspect of this example is not that the agent doesn't cross -- it makes sense that a proof-based agent can't cross a bridge whose safety is dependent on the agent's own logic being consistent, since proof-based agents can't know whether their logic is consistent. Rather, the point is that the agent's "counterfactual" reasoning looks crazy. *(However, keep reading for a version of the argument where it* ***does*** *make the agent take the wrong action.)* Arguably, the agent should be uncertain of what happens if it crosses the bridge, rather than certain that the bridge would blow up. Furthermore, the agent is reasoning as if it can control whether PA is consistent, which is arguably wrong.

In a comment, Stuart points out that this reasoning seems highly dependent on the code of the agent; the "else" clause could be different, and the argument falls apart. I think the argument keeps its force:

- On the one hand, it's still very concerning if the sensibility of the agent depends greatly on which action it performs in the "else" case.
- On the other hand, we can modify the troll's behavior to match the modified agent. The general rule is that the troll blows up the bridge if the agent would cross for a "dumb reason" -- the agent then concludes that the bridge would be blown up if it crossed. I can no longer complain that the agent reasons as if it were controlling the consistency of PA, but I can still complain that the agent thinks an action is bad because that action indicates its own insanity, due to a troublingly circular argument.

Analogy to Smoking Lesion

One interpretation of this thought-experiment is that it shows proof-based decision theory to be essentially a version of EDT, in that it has EDT-like behavior for Smoking Lesion. The analogy to Smoking Lesion is relatively strong:

- An agent is at risk of having a significant internal issue. (In Smoking Lesion, it’s a medical issue. In Troll Bridge, it is logical inconsistency.)
- The internal issue would bias the agent toward a particular action. (In Smoking Lesion, the agent smokes. In Troll Bridge, an inconsistent agent crosses the bridge.)
- The internal issue also causes some imagined practical problem for the agent. (In Smoking Lesion, the lesion makes one more likely to get cancer. In Troll Bridge, the inconsistency would make the troll blow up the bridge.)
- There is a chain of reasoning which combines these facts to stop the agent from taking the action. (In smoking lesion, EDT refuses to smoke due to the correlation with cancer. In Troll Bridge, the proof-based agent refuses to cross the bridge because of a Löbian proof that crossing the bridge leads to disaster.)
- We intuitively find the conclusion nonsensical. (It seems the EDT agent should smoke; it seems the proof-based agent should not expect the bridge to explode.)

Indeed, the analogy to smoking lesion seems to strengthen the final point -- that the counterfactual reasoning is wrong.

I've come to think of Troll Bridge as "the real smoking lesion", since I'm generally not satisfied with how smoking lesion is set up.

But is proof-based decision theory *really* a version of EDT? I think there's more to say about the analogy, but a simple explanation is this: both EDT and proof-based decision theory evaluate actions by adding them to the knowledge base and seeing what the world looks like under that additional assumption. Or, to put it differently, proof-based DT reasons about actions as if they're observations. That's the fundamental idea of evidential decision theory.

Still, as good as the analogy might be, you still probably shouldn't be convinced until you've seen the Troll Bridge defeat *real* EDT.

Probabilistic Version

For the purely logical version, I said that we can't fault the conclusion (because the agent can't prove that it is safe to cross the bridge) – only the *reasoning* is being critiqued. However, the probabilistic version better demonstrates the severity of the reasoning error; it illustrates an EDT agent making what seems to be an outright mistake.

It's reasonable to avoid crossing the bridge if you're only thinking about provable results. But we might expect an agent who balances risks and rewards would decide to cross, so long as it believes the probability of PA being inconsistent is low. Thus turns out not to be the case.

Here's the setup.

Suppose that the agent uses a probability distribution which "respects logic" in the sense that it assigns probability zero to anything which is logically refutable. (This assumes logical omniscience. I'll drop that assumption later.)

The agent is as follows: *if P(cross)=0, cross; if P(¬cross)=0, don't cross; else, take the action with highest expected utility, breaking ties by not crossing.* The idea is that we are trying to take the highest-expected-utility action; however, we must handle cases where this isn't well-defined. Choosing to take an action when the probability of the action is zero guarantees that the agent can never prove its own action; it is a version of the chicken rule. Breaking ties by not crossing is arbitrary, but gives the simplest version of the example, because this choice reduces the number of cases we need to check.

The troll is as follows: *if the agent crosses the bridge due to the P(cross)=0 clause, blow up the bridge.*

Reasoning in the logic of the agent:

- Suppose $\Box(A = \text{cross} \to U = -10)$
    - Suppose A = cross
        - Examining the agent, either crossing had higher expected utility, or P(cross)=0. But we assumed $\Box(A = \text{cross} \to U = -10)$, so it must be the latter. So the bridge gets blown up.
    - Therefore, $A = \text{cross} \to U = -10$.
- Therefore, $\Box(A = \text{cross} \to U = -10) \to (A = \text{cross} \to U = -10)$

Thus, by Löb's theorem, we see that $\Box(A = \text{cross} \to U = -10)$.

Therefore, since the agent is certain of things which are provable, the agent assigns expected value -10 to crossing. Since the expected value of the other action will always be higher than this, and since we know the agent doesn't ever really enter into the P=0 conditions, the agent will choose not to cross. $\Box$

Notice that this reasoning did not depend much on the values 10, 0, and -10. The utility of the bridge getting blown up could be -0.000001, and the agent still won't cross. *It isn't weighing the risk; it's decided that the worst outcome is inevitable.* In the case of proof-based agents, I said that the overall decision not to cross was understandable, since proof-based agents are unable to weigh the risks. A probabilistic agent, however, should intuitively be able to say "I don't know whether the bridge will get blown up, because it involves reasoning about properties of myself which I'm fundamentally uncertain about; but, the odds look decent." But that's not what happens: instead, it is sure that crossing is unfavorable, *no matter what overall probability it assigns to P(A=cross)=0*.

So, in this case we conclude that the Troll Bridge example results in a chicken-rule-based agent taking the wrong action overall. The agent shouldn't be sure that it would cross "for the right reason" (it should assign some probability to P(A=cross)=0, since it can't know that its own logic is consistent). However, intuitively, it should be able to assign some probability to this, and balance the risks. If the downside risk is U=-0.000001, and the probability it assigns to its logic being consistent is not similarly small, it should cross -- and in doing so, it would get +10.

As mentioned for the proof-based agent, the agent's code is a bit arbitrary, and it is worth asking how important the details were. In particular, the default in the case of a tie was to not cross. What if the default in case of a tie were to cross?

We then modify the troll's algorithm to blow up the bridge if and only if P(A=cross)=0 **or** there is a tie. The proof then goes through in the same way.

Perhaps you think that the problem with the above version is that I assumed logical omniscience. It is unrealistic to suppose that agents have beliefs which perfectly respect logic. (Un)Fortunately, the argument doesn't really depend on this; it only requires that the agent respects proofs which it can see, and eventually sees the Löbian proof referenced.

Random Exploration

The frustrating thing about Troll Bridge is that it seems like the agent could just cross the bridge, and things would be fine. The proof that things *wouldn't* be fine *relies on the fact that the agent accepts that very proof as sufficient reason*; so can't we just ignore that kind of proof somehow?

One thing you might try is to consider a learning agent, and force random exploration so the agent just crosses the bridge sometimes. If the agent crosses the bridge, it should be able to see that it's safe, right?

However, we have to ask: what's the appropriate version of Troll Bridge for the exploring agent? Remember I said that the basic idea of Troll Bridge is that the troll blows up the bridge if the agent crosses "for a dumb reason" -- ie, for a reason other than "crossing is a good idea". Random exploration falls into this category.

Furthermore, random exploration is really closely analogous to the chicken rule. Consider:

- (A) The first chicken rule I mentioned: if you prove that you don't take an action, you immediately take it.
- (B) The second chicken rule mentioned: if P(A)=0, then do A.
- (C) Epsilon-chicken: if P(A)<epsilon, then do A.
- (D) Epsilon-exploration: randomly do A with probability epsilon.

(A) is very similar to (B) if the probability distribution respects proofs in the way I mentioned previously; if the probability distribution has the further property of assigning positive probability to anything not disproven, then they're exactly the same. (C) is a natural generalization of (B). (B) tells us what to do when the EDT decision rule isn't well-defined (we can't condition on probability zero events). (C) generalizes this by treating small-probability events in the same way (we might expect our conditional beliefs to be bad for small-probability events, since we presumably don't have much experience with them; so we treat it as if it's the undefined case). (B) ensures there are no zero-probability actions, which means we don't actually ever have to deal with the EDT decision rule being non-well-defined; similarly, (C) ensures no action is almost surely never taken, which helps ensure that our conditional expectations for all actions are empirically grounded.

(C) and (D) are very similar; (C) just supplies pseudorandomness vie the agent's own probabilities. Whenever the agent is too confident about what it does, it does something else. This makes the randomization unpredictable *too the agent*. (D) supplies randomness from an external source instead.

So we have a chain connecting the chicken rule to random exploration.

(Another way of thinking about the similarity: The chicken rule ensures that there are models of the agent's beliefs in which each action is taken, so that the agent can have sensible logical reasoning about each alternative. Epsilon-exploration ensures that the empirical experiences of the agent include all of the actions, so that the agent can have reasonable empirical estimates of consequences of each action.)

OK. So Troll Bridge for an exploring agent should blow up the bridge on exploration, *or* if PA is inconsistent. An agent could look like this:

- With small probability, explore.
  - If exploring, flip a coin to cross or not cross.
- If P(cross)=0, cross.
- If P(¬cross)=0, don't cross.
- Otherwise, take the max-expected-value action, breaking ties randomly.

I'm still assuming that the agent's probability distribution respects proofs, as before. I'm also assuming this agent is playing the game repeatedly, and learning. I also must assume that the agent has found Now, the agent reasons:

- Suppose □(cross → u=-10) for a particular round.
  - Further suppose I crossed on that round.
    - By the first supposition, I knew the payout of crossing to be low; and I must also have known that the payout of not crossing is higher, since I can prove that. Since I can prove what both payouts are, the expected values must equal those, unless PA is inconsistent (in which case P(cross)=0 anyway, since my beliefs respect proofs). So I can only be crossing the bridge for two reasons -- either this is an exploration round, or P(cross)=0.
    - In either case, crossing the bridge yields payout u=-10.
  - Therefore, cross → u=-10 in fact.
- So □(cross → u=-10) → (cross → u=-10).

Since the agent proves that a proof of crossing being bad implies crossing is actually bad, the agent further must prove that crossing is bad in fact, by Löb.

I did this for the logically omniscient case again, but as before, I claim that you can translate the above proof to work in the case that the agent's beliefs respect proofs it can find. That's maybe a bit weird, though, because it involves a Bayesian agent updating on logical proofs; we know this isn't a particularly good way of handling logical uncertainty.

We can use logical induction instead, using an epsilon-exploring version of LIDT. We consider LIDT on a sequence of troll-bridge problems, and show that it eventually notices the Löbian proof and starts refusing to cross. This is even more frustrating than the previous examples, because LIDT might successfully cross for a long time, apparently learning that crossing is safe, and reliably gets +10 payoff. Then, one day, it finds the Löbian proof and stops crossing the bridge!

That case is a little more complicated to work out than the Bayesian probability case, and I omit the proof here.

Non-examples: RL

On the other hand, consider an agent which uses random exploration but doesn't do any logical reasoning, like a typical RL agent. Such an agent doesn't need any chicken rule, since it doesn't care about proofs of what it'll do. It still needs to explore, though. So the troll can blow up the bridge whenever the RL agent crosses due to exploration.

This obviously messes with the RL agent's ability to learn to cross the bridge. The RL agent might never learn to cross, since every time it tries it, it looks bad. So this is sort of similar to Troll Bridge.

However, I think this isn't really the point of Troll Bridge. The key difference is this: the RL agent can get past the bridge if its prior expectation that crossing is a good idea is high enough. It just starts out crossing, and happily crosses all the time.

Troll Bridge is about the inevitable confidence that crossing the bridge is bad. We would be fine if an agent decided not to cross because it assigned high probability to PA being inconsistent. The RL example seems similar in that it depends on the agent's prior.

We could try to alter the example to get that kind of inevitability. Maybe we argue it's still "dumb" to cross only because you start with a high prior probability of it being good. Have the troll punish crossing *unless the crossing is justified by an empirical history of crossing being good*. Then RL agents do poorly no matter what -- no one can get the good outcome in order to build up the history, since getting the good outcome *requires* the history.

But this still doesn't seem so interesting. You're just messing with these agents. It isn't illustrating the degree of pathological reasoning which the Löbian proof illustrates -- **of course** you don't put your hand in the fire if you get burned every single time you try it. There's nothing wrong with the way the RL agent is reacting!

So, Troll Bridge seems to be more exclusively about agents who do reason logically.

Conclusions

All of the examples have depended on a version of the chicken rule. This leaves us with a fascinating catch-22:

- We need the chicken rule to avoid spurious proofs. As a reminder: spurious proofs are cases where an agent would reject an action if it could prove that it would not take that action. These actions can then be rejected by an application of Löb's theorem. The chicken rule avoids this problem by ensuring that agents cannot know their own actions, since if they did then they'd take a different action from the one they know they'll take (and they know this, conditional on their logic being consistent).
- However, Troll Bridge shows that the chicken rule can lead to another kind of problematic Löbian proof.

So, we might take Troll Bridge to show that the chicken rule does not achieve its goal, and therefore reject the chicken rule. However, this conclusion is very severe. We cannot simply drop the chicken rule and open the gates to the (much more common!) spurious proofs. We would need an altogether different way of rejecting the spurious proofs; perhaps a full account of logical counterfactuals.

Furthermore, it is possible to come up with variants of Troll Bridge which counter some such proposals. In particular, Troll Bridge was originally invented to counter proof-length counterfactuals, which essentially generalize chicken rules, and therefore lead to the same Troll Bridge problems).

Another possible conclusion could be that Troll Bridge is simply too hard, and we need to accept that agents will be vulnerable to this kind of reasoning.

Optimization Provenance

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Transparency is vital for ML-type approaches to AI alignment, and is also an important part of agent foundations research. In this post, we lay out an agenda for formalizing transparency which we'll call the Optimization Provenance Agenda.

In particular, the goal is to create a notion of transparency strong enough that an attempted deception would be completely transparent. The basic idea is that at any point, not only should an agent's world model and subgoals be legible, but the entire provenance of all of the optimization processes which are part of the agent should be legible as well.

This agenda is a joint development between me and Evan Hubinger. Special thanks to Hjalmar Wijk and Evan Hubinger for their comments and feedback on this post!

Background notions

In order to discuss the notions here, it will be helpful to have working definitions of the key concepts involved.

Legibility

Intuitively, legibility means that a human should be able to look at something, and be able to understand it easily and correctly. If the thing in question is very large, we might at best only be able to have local legibility, where any given part below a certain size is legible, but the whole thing is not understandable by a human. In an amplification scenario, local legibility may be sufficient, with an amplified human being capable of understanding the global structure. For the purposes of this post, we'll consider legibility to also include these cases. One major issue with the concept of legibility is that it seems very difficult to create something legible through a legible process.

It seems plausible to me that existing ML techniques could be combined and extended to produce natural language descriptions of learned world models. However, this process itself would most likely be very illegible. Even in the case of human communication, it is possible for a human to produce a legible plan, but humans do not seem to be very capable of producing a legible explanation of the process which produced that plan. So it seems likely to me that we may have to decide some illegible process to trust in order to get off the ground with this approach. This could simply be trusting illegible human mental processes, or it could be something like trusting models produced in a mathematically simple way.

World model

A world model belongs to a decision making process, and is used by the process to predict what the result of various decisions would be, so that it can make the best choice. It's important that the world model includes everything going into its decision making process

Due to our motivations in transparency, we will typically think of world models as being made of highly composable models, which each model an aspect of the entire world (including very abstract aspects). I believe that Goguen's sheaf semantics is a promising framework for formalizing this type of world model. It's important to note that world models in current ML methods are not composable this way, which makes these models much less legible.

World models can also be implicit or explicit. The canonical example of an implicit world model is that of a thermostat, where the world model is implicitly represented by the thermistor or bimetallic strip. An explicit world model is represented in a modeling framework, such as a sheaf model. The exact line between explicit and implicit world models seems to be nebulous. For our purposes, it is much better to have an explicit world model, since it is more legible to have something explicitly. Note that implicit models can still be legible though, such as in the thermostat example.

Optimizer

We will consider an optimizer to be made of three components:

1. A world model

2. An objective

3. An optimization process

As an example, consider AlphaGo, which uses Monte-Carlo Tree Search. The world model is distributed, with an implicit part in the selection network, and an explicit part in the expansion and simulation steps of the search. The objective is to maximize the probability of winning the game, and the optimization process is simply backpropagation over the tree.

In order to be legible, it is best if the objective is defined in terms of explicit parts of the world model. While I do not believe this is strictly necessary, I feel very skeptical that implicit objectives will be adequately transparent.

Provenance

The provenance of an object is the entire history of how that object came to be. As an example, art museums place high value on knowing the complete provenance of a famous painting, as it greatly increases their confidence that the painting is indeed legitimate.

In this context, we want to understand the provenance of all optimization processes running in the agent. In particular, for each optimizer, we want to understand the optimization process which created the optimizer.

Motivations

In addition to generally improving our understanding of transparency, there are two main motivations going into the optimization provenance agenda. So before we get into more details of the agenda, we will introduce these motivations here.

Treacherous turn

In order to perform a treacherous turn, or other kinds of deceptive alignment, an agent must at some point conceive of this plan. This conception will be the result of an optimizer, and so this should be in the optimization provenance. If this provenance is legible, then this should be detected by an overseer before it comes into fruition. So we expect that an agent with optimization provenance will be much less likely to achieve or even attempt a treacherous turn.

Goodhart's curse

More generally, we want to prevent Goodhart's curse scenarios, and ensure inner alignment. If the agent is optimizing for a proxy to a goal, this is potentially legible. However, we will typically expect a training process to produce a mesa-optimizer which is optimizing for some proxy, since values are likely to be too complex to optimize for directly. So how do we distinguish between robustly aligned mesa-optimizers and pseudo-aligned mesa-optimizers (i.e. those that Goodhart)? If we understand the optimizer which created the mesa-optimizer in addition to the mesa-optimizer itself, we believe this will give insight into whether this is likely to happen or not. Understanding the provenance of optimizers could thus give us techniques to help us solve this problem.

The optimization provenance agenda

We're now ready to understand the three key pieces of optimization provenance agenda. We believe that a solution to these would represent a large step in solving transparency.

Legible optimizers

In order for this to work, the optimizers running as part of the agent need to be legible. I believe the hardest part of this is ensuring that the world model of the optimizer is legible. Once we have a legible world model, I expect it will be relatively easy to ensure the objective is also legible.

Ensuring that the optimization process is legible falls more naturally into the next section.

1. Find an adequate framework for explicit world models

2. Explicitly understand the objective

3. Understand what makes something an implicit world model

Mesa-optimizer control

The optimization processes used in the agent must all be capable of controlling whether it creates a mesa-optimizer. In particular, a mesa-optimizer should only be formed when its three components are all known and assured. This requires the optimization process to be legible in the sense that we understand under what conditions the process will create a new mesa-optimizer.

1. When are mesa-optimizers created?

2. How do we ensure that the mesa-optimizer will be legible?

3. Can such mesa-optimizers be made corrigible?

4. Does this apply to other forms of optimization daemons ?

Provenance accountability

Having legible provenance of every optimizer occurring in the agent is a good step towards solving transparency. But this is not enough by itself, because provenance may itself be forged. In order for the provenance to be assured, an overseer must investigate any changes made to the optimization provenance, and verify that there is no optimization toward unaligned goals.

1. What constitutes adequate oversight?

2. How possible is it for accidental treachery to be produced?

3. To what extent can Goodhart's curse be avoided with optimization provenance?

Recursive uses

It seems likely that progress on the optimization provenance agenda could be leveraged to make progress on other subproblems. In particular, I think that developing frameworks for explicit models will make it easier to solve the inner alignment problem. I also believe that the idea of optimization provenance is a useful handle for thinking about mesa-optimizers in general.