

Embedded Agency

1. [Embedded Agents](#)
2. [Decision Theory](#)
3. [Embedded World-Models](#)
4. [Robust Delegation](#)
5. [Subsystem Alignment](#)
6. [Embedded Curiosities](#)
7. [Embedded Agency \(full-text version\)](#)

Embedded Agents

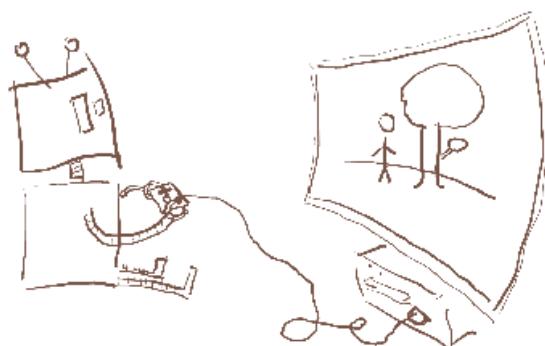
Crossposted from the [AI Alignment Forum](#). May contain more technical jargon than usual.

(A longer text-based version of this post is also available on MIRI's blog [here](#), and the bibliography for the whole sequence can be found [here](#))

Embedded Agency

Abram Demski & Scott Garrabrant

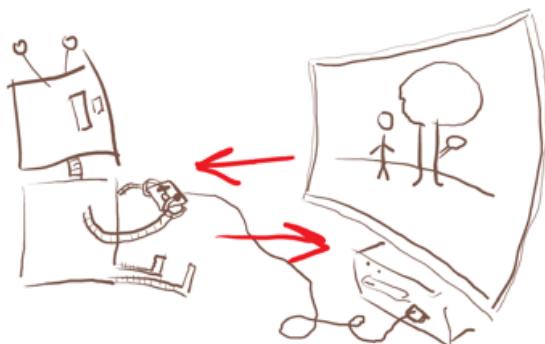
This is Alexei.



Alexei is playing a

video game.

Alexei interacts with the environment via well-defined i/o channels.



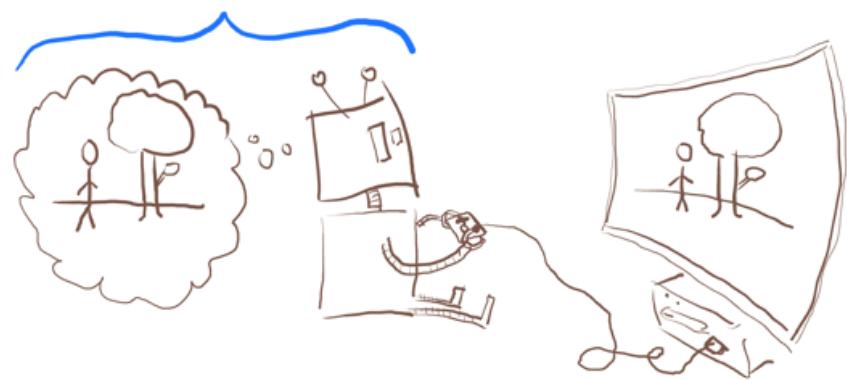
This means Alexei has a clearly-defined functional relationship with the environment, defining action consequences.

Alexei can hold the entire environment in mind.

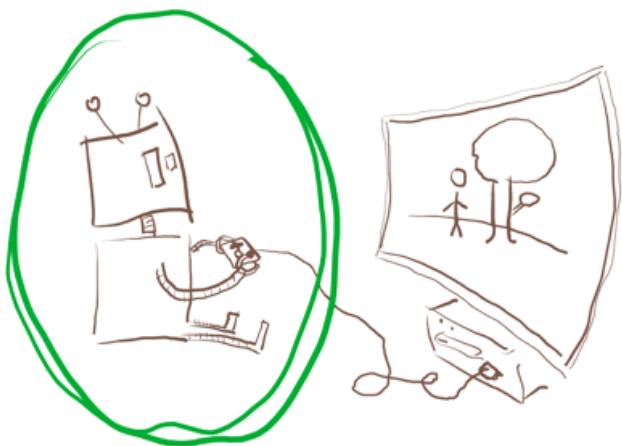


Alexei may need to learn what the environment is like, but in doing so, can represent every detail.

Alexei thinks about manipulating and controlling the environment, but not himself.

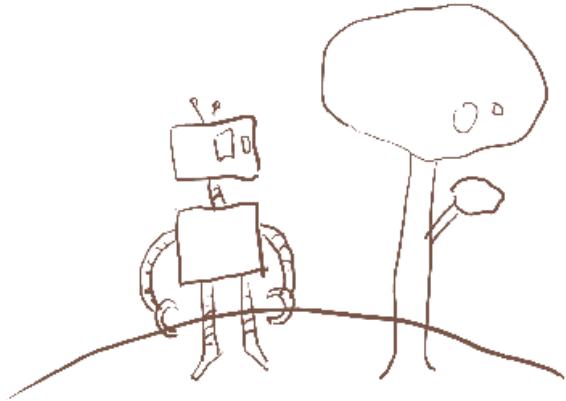


He doesn't have the opportunity or risk of arbitrary self-modification, because the environment can't really touch him. He can't really die, either; he only has to worry about restarts.



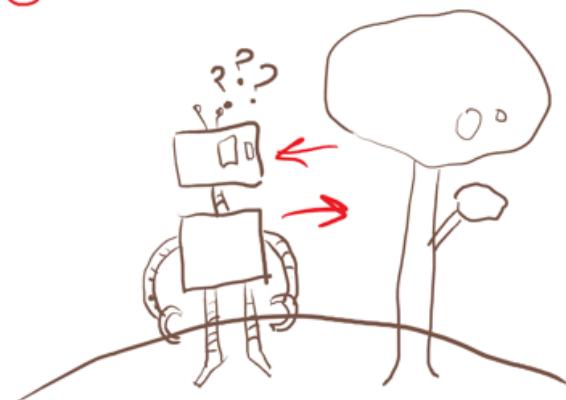
Alexei may think like a reductive scientist about the world, breaking it into parts, but the concept of agent is non-reductive; Alexei is an indivisible atom which produces actions.

This is Emmy.



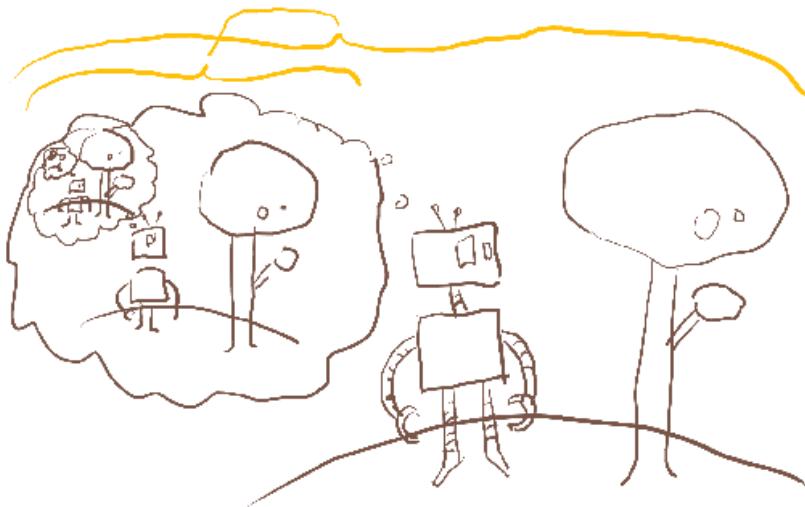
Emmy is playing real life.

Emmy is part of the universe, not sitting outside, so it is hard for her to imagine taking "different actions".



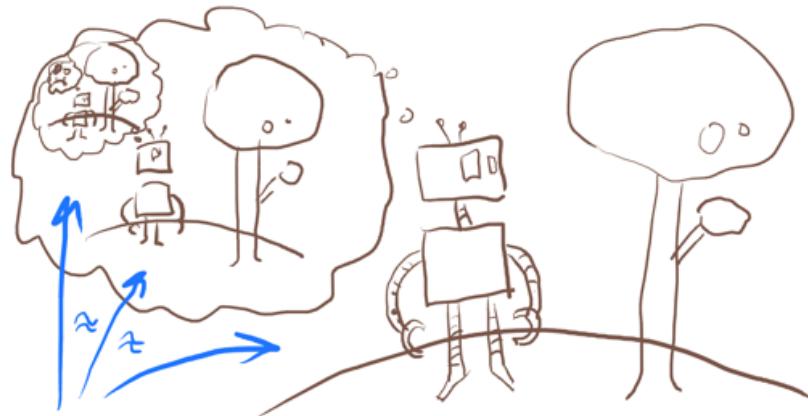
Alexei can poke the universe and see what happens. Emmy is the universe poking itself.

Emmy can't hold the entire world in her head.



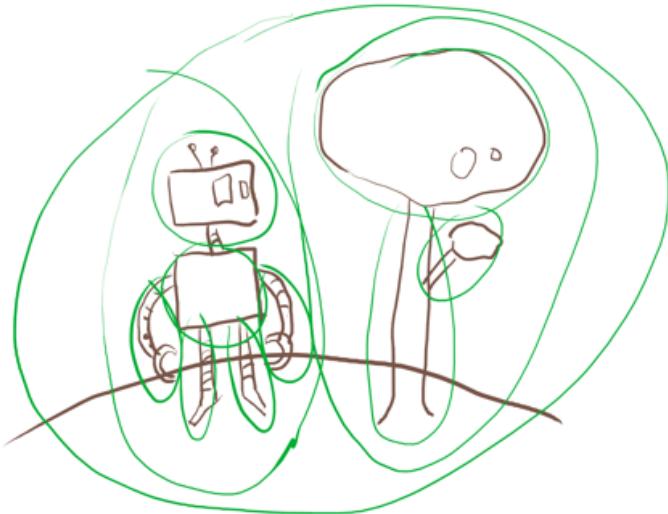
Any model she uses will be very partial and approximate.

Emmy can reflect and self-improve.



Emmy thinks about how to think about how to win, where Alexei only thinks about how to win.

Emmy is made of parts,
just like everything else.



She isn't really a unitary entity; she
is just a bunch of stuff. Somehow we
get an agent out of non-agentic pieces.

This is Marcus Hutter.

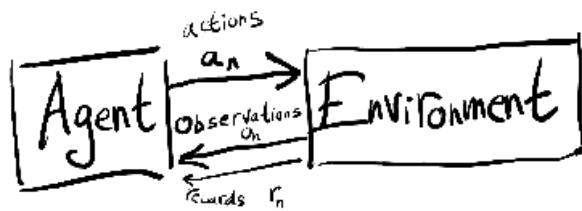


Marcus
Hutter

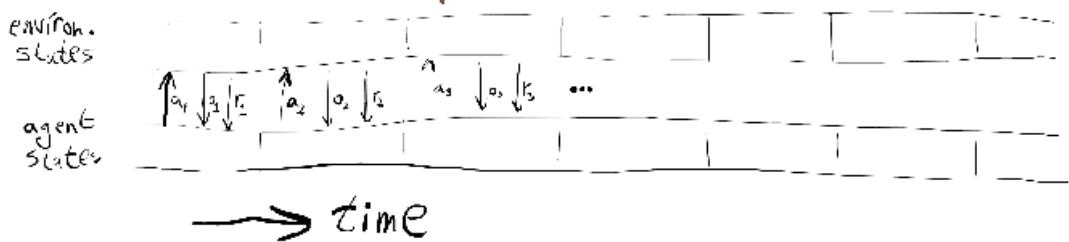
$$a_k = \operatorname{argmax}_{a_k} \sum_{o_k r_k} \dots \max_{a_m} \sum_{o_m r_m} [r_k + \dots + r_m] \sum 2^{-L(q)}$$

$$q: U(q_1, \dots, q_n) = \alpha r_1, \dots, \alpha r_n$$

Marcus Hutter's **AIXI** model tells us all about the kind of thinking Alexei needs to do to be good at winning video games.



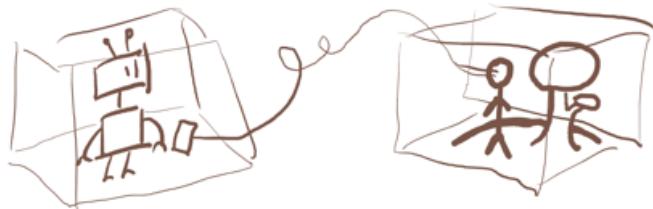
The ADXI model sets up a somewhat symmetric relationship between the agent and environment: the agent produces a sequence of actions which are a function of previous observations and rewards, while the environment produces observations and rewards in a way which is a function of the previous actions.



AIXI doesn't know which environment it is interacting with, so it uses a probability distribution on all computable environments.



Its task is to learn about its environment through observation, and plan to get as much reward as possible (taking into account its uncertainty to hedge its bets).

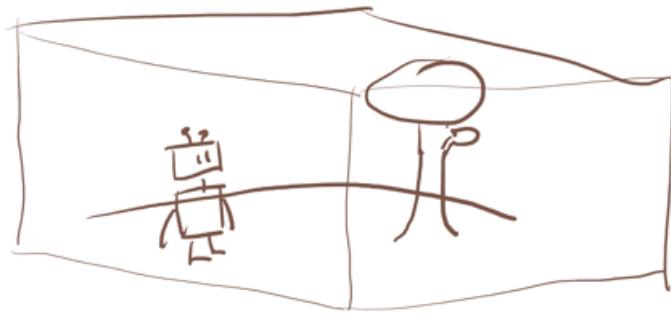


Agent models like AIXI are dualistic: the agent exists outside of the environment, with only set interactions between agent-stuff and environment-stuff. They require the agent to be larger than the environment and don't tend to model self-referential reasoning because the agent is made of different stuff than what the agent reasons about.

These dualistic assumptions are not unique to AIXI; they are very common among models of rational agency.

We would like to understand agents like Emmy as well as we currently understand agents like Alexei. AIXI serves as an illustration of what this high level of understanding looks like.

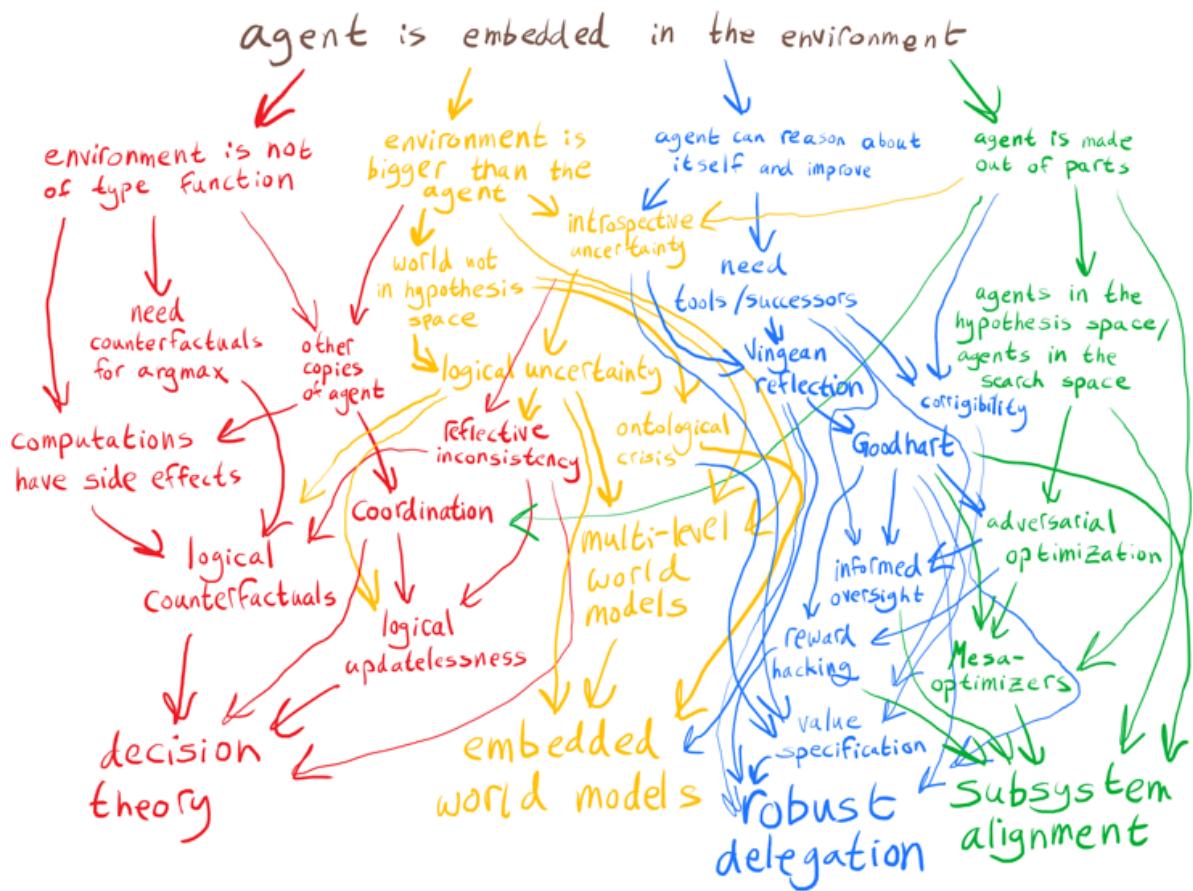
However, agents in the real world are forced to break important assumptions of the AIXI model.



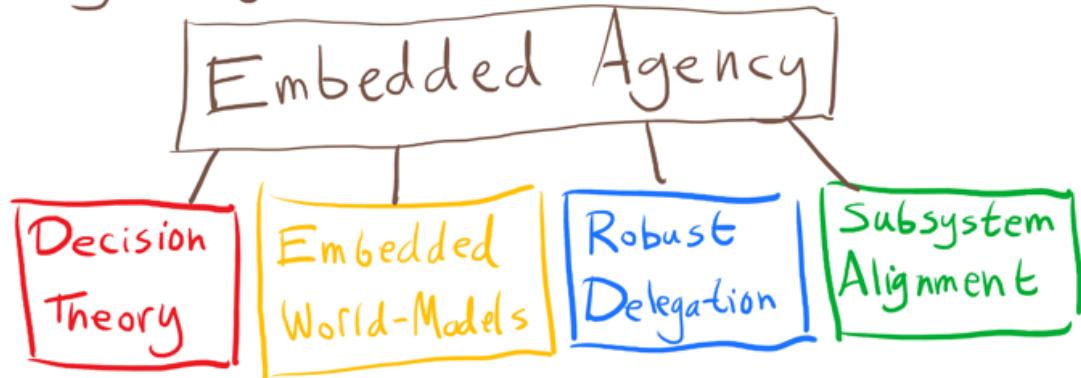
Agents like Emmy are embedded in their environments. Embedded agents break the dualistic assumptions:

- not given well-defined i/o channels
- smaller than the environment
- able to reason about themselves & self-improve
- made of parts like the environment

The four properties don't cleanly divide everything up. They're more like four ways of looking at the same problem than mutually exclusive categories.



We can cluster problems of embedded agency into four subfields, which is as close to a nice partitioning as we're likely to get.



These have a rough correspondence to the four properties of embedded agents.

Decision Theory: Adapting classical decision theory to embedded agents

- Counterfactuals
- Newcomblike problems; copies of the agent
- Reasoning about other agents
- Extortion problems
- Coordination problems
- Logical counterfactuals
- Logical updatelessness

Embedded World-Models: understanding epistemic states appropriate for embedded agents

- world not in hypothesis space
("realizability"/"grain of truth" problem)
- logical uncertainty
- high-level models
- multi-level models
- ontological crises
- agent must be in world-model (Naturalized Induction)
- anthropic reasoning

Robust Delegation: understanding what trust relationships can exist between an agent and its future self or other agents it can delegate to.

- Vingeian Reflection
- Tiling problem
- Averting Goodhart's Law
- Value Loading
- Corrigibility
- Informed Oversight

Subsystem Alignment: Ensuring
that subsystems are not working at
cross purposes; avoiding subprocesses
optimizing for unintended goals.

- Benign Induction
- Benign Optimization
- Transparency
- Mesa-Optimizers

These four areas point to our biggest confusions about how highly intelligent systems could work in the real world.

The next four sections will go into these areas in more detail.

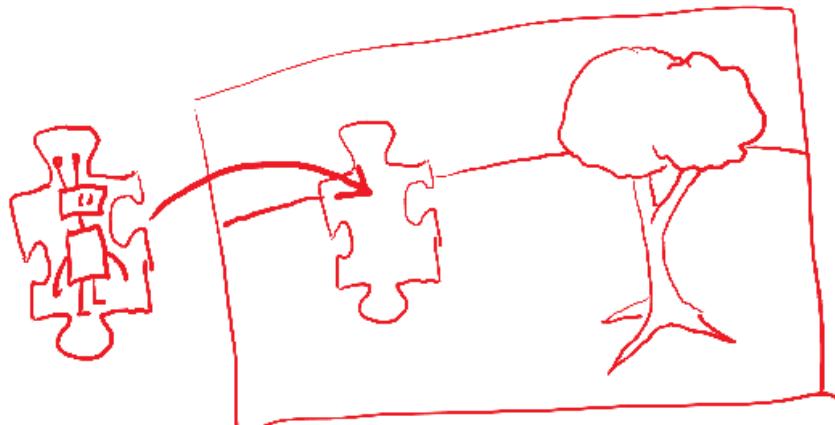
Decision Theory

Crossposted from the [AI Alignment Forum](#). May contain more technical jargon than usual.

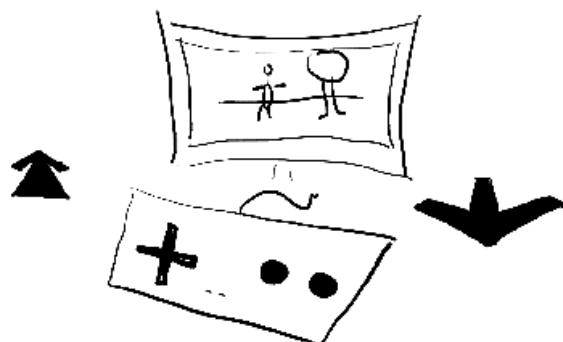
(A longer text-based version of this post is also available on MIRI's blog [here](#), and the bibliography for the whole sequence can be found [here](#).)

[Decision Theory]

environment is not of type function



[Abram Demski and Scott Garrabrant]



Decision theory and artificial intelligence typically try to compute something resembling

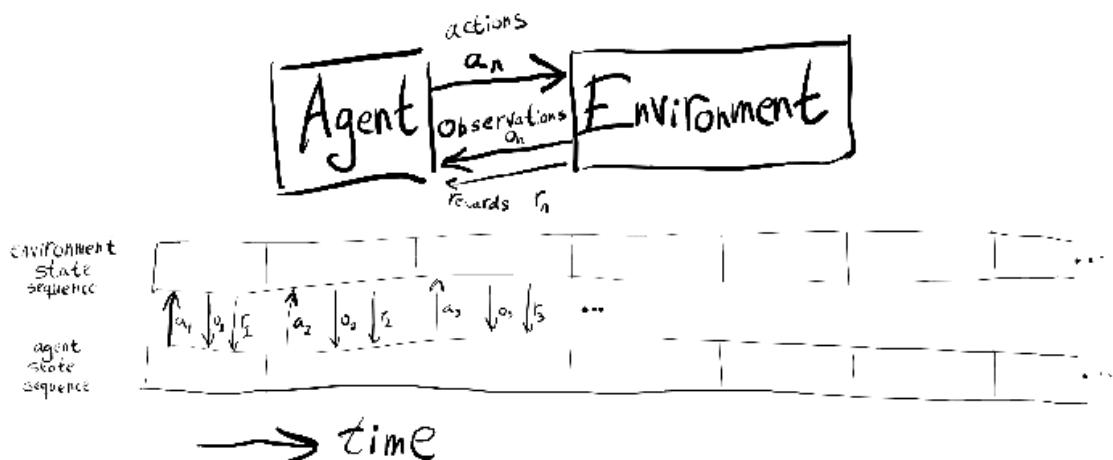
$$\underset{a \in \text{Actions}}{\operatorname{argmax}} f(a)$$

IE, maximize some function of the action.

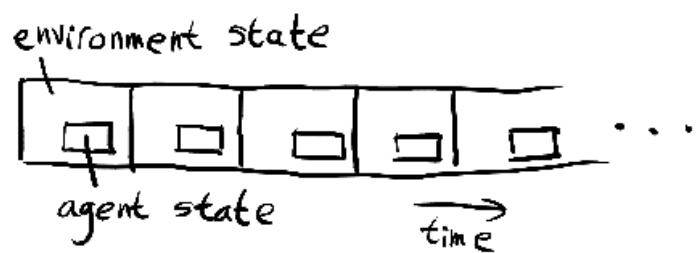
This tends to assume that we can detangle

....
things enough to see outcomes as a function of actions.

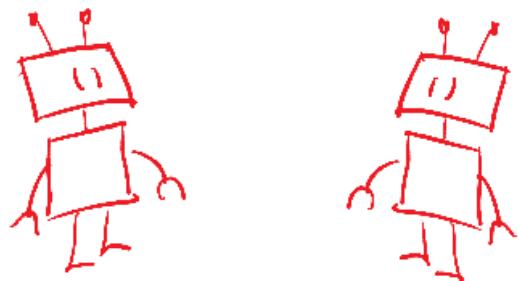
For example, AIXI represents the agent and the environment as separate units which interact over time through clearly defined i/o channels, so that it can then choose actions maximizing reward.



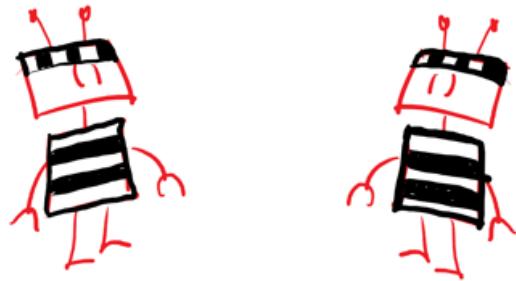
When the agent model is a part of the environment model, it can be significantly less clear how to consider taking alternative actions.



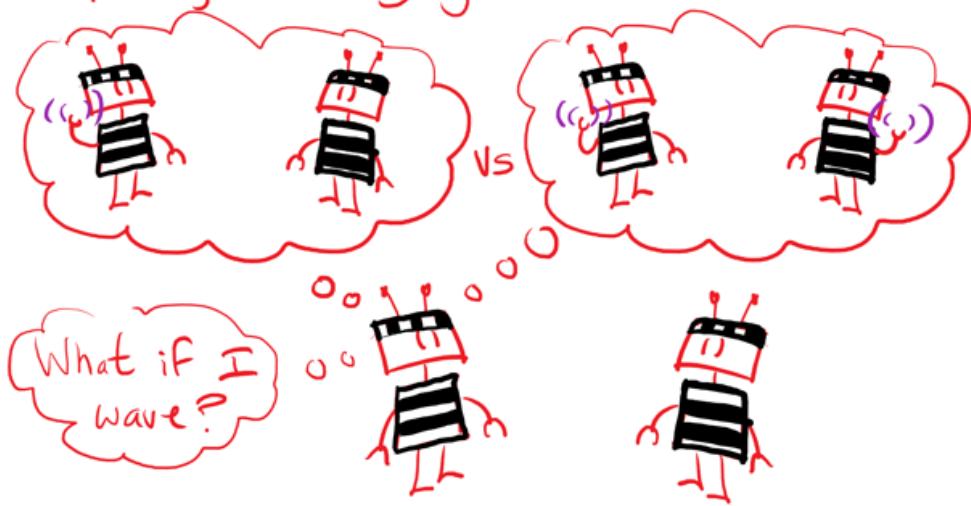
For example, because the agent
is smaller than the environment,
there can be other copies of the
agent, or things very similar to
the agent.



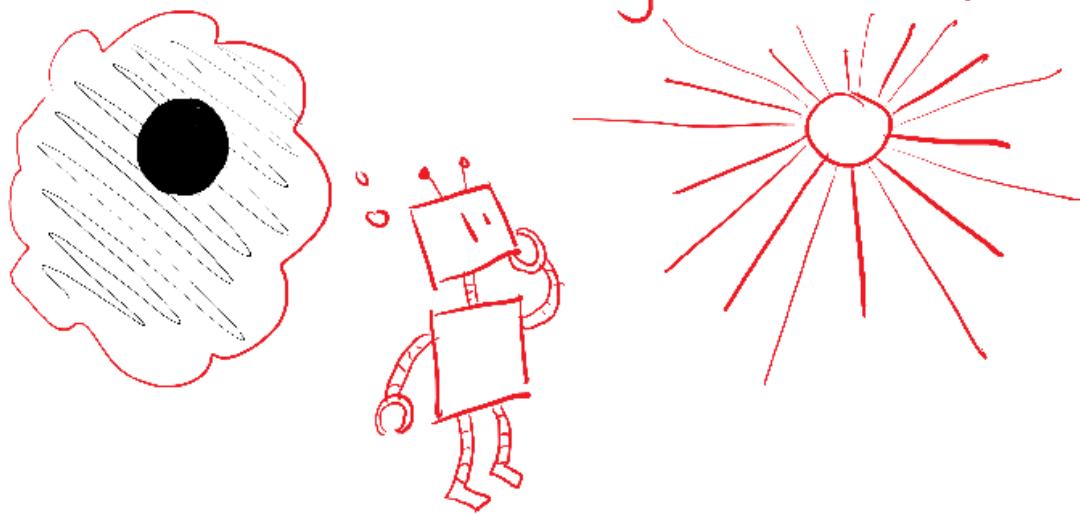
This leads to contentious decision-theory problems such as Twin Prisoner's Dilemma and Newcomb's problem.



Depending on how you draw the boundary around "yourself", you might think you control the action of both copies, or only your own.



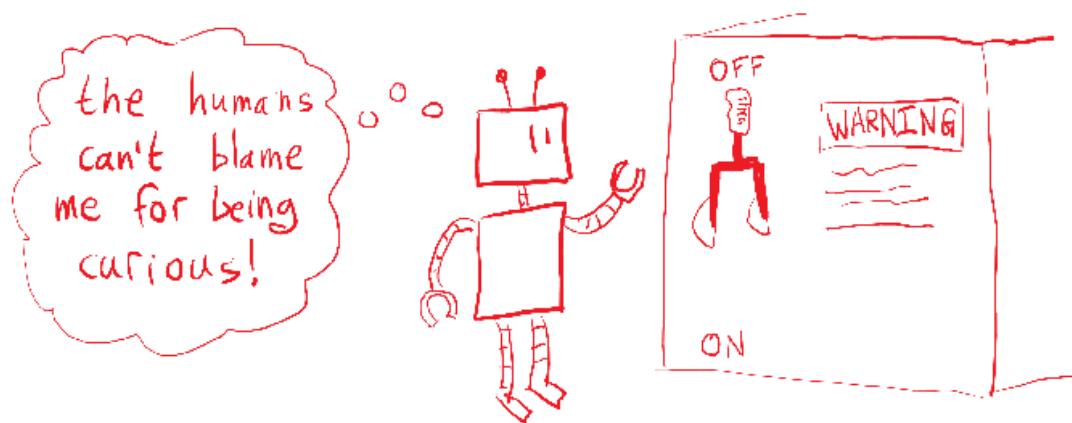
This is an instance of the problem of counterfactual reasoning: how do we evaluate hypotheticals like "What if the sun suddenly went out"?



Problems of adapting decision theory to embedded agents include:

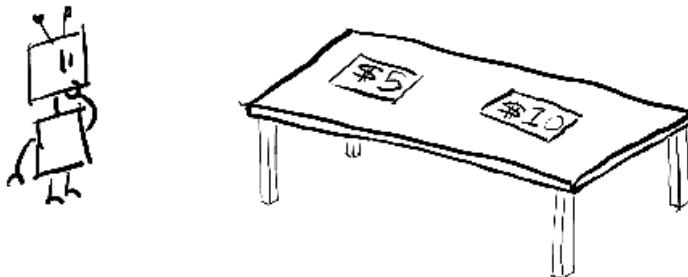
- Counterfactuals
- Newcomblike problems; copies of the agent
- Reasoning about other agents
- Extortion problems
- Coordination problems
- Logical counterfactuals
- Logical updatelessness

Agents first and foremost need to concern themselves with counterfactuals about their own actions.



The difficulty with action counterfactuals can be illustrated by the five-and-ten problem.

Suppose we have the option of taking a five dollar bill or a ten dollar bill, and all we care about in the situation is how much money we get. Obviously, we should take the \$10.



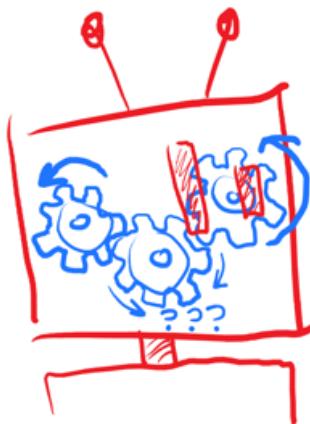
However, it is not so easy as it seems to reliably take the \$10. If you reason about yourself as just another part of the environment, then you can know your own behavior. This throws a monkey wrench into many common reasoning methods.

Obviously I take \$10,
so if I take \$5,
contradiction. From
contradiction, anything
follows, meaning taking \$5
nets me \$1,000...



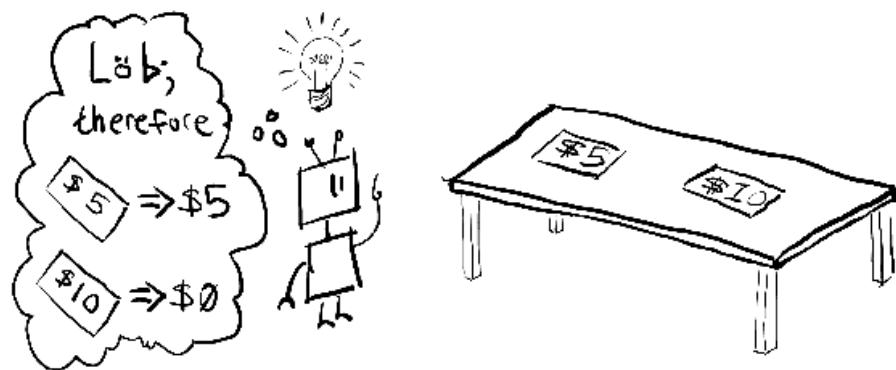
If we try to calculate the expected utility of our actions by Bayesian conditioning, as is common, knowing our own behavior leads to a divide-by-zero error when we try to calculate the expected utility of actions we know we don't take:

$$\neg A$$
$$P(A) = 0$$
$$P(B \& A) = 0$$
$$P(B|A) = \frac{P(B \& A)}{P(A)} = \frac{0}{0}$$



Because the agent doesn't know how to separate itself from the environment, it gets gnashing internal gears when it tries to imagine taking different actions.

But the biggest complication comes from Löb's theorem, which can make reasonable-looking agents stably take the \$5 because "If I take the \$10, I get \$0"!



This might be hard to believe; so, let's look at a detailed example. The phenomenon can be illustrated by the behavior of simple logic-based agents reasoning about the five-and-ten problem.

Here is our example:

Agent

$A() :=$

Spend some time
searching for proofs of
sentences of the form
 $"[A()=5 \rightarrow U()=x] \ \& \ [A()=10 \rightarrow U()=y]"$

for $x, y \in \{0, 5, 10\}$

If a proof is found with
 $x > y$:

return 5

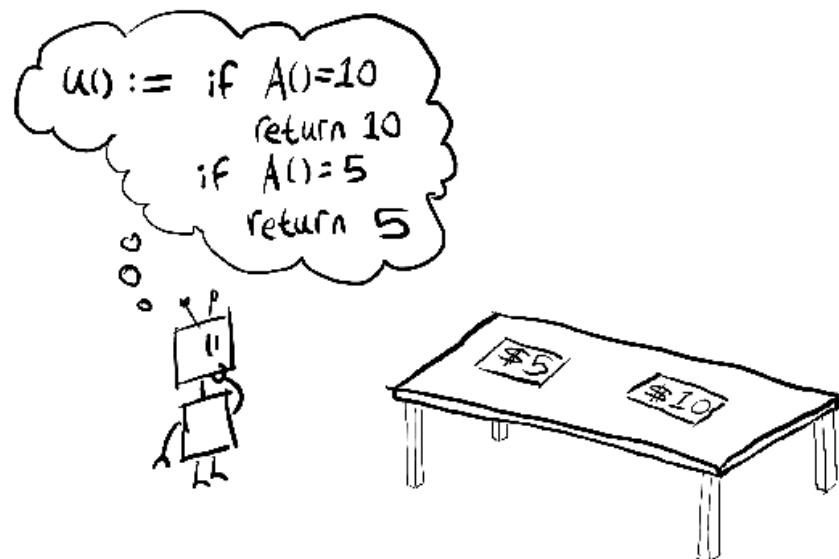
else:

return 10

Universe

$U() :=$ if $A()=10$
return 10
if $A()=5$
return 5

It seems easy when you just imagine an agent trying to reason about the universe.



However, if the amount of time spent searching for proofs is enough, the agent will always choose 5!

The proof is by Löb's theorem. Löb's theorem says that, for any proposition P , if you can prove that a proof of P would imply the truth of P , then you can prove P .

In symbols, with " $\Box X$ " meaning "X is provable" :

$$\text{Löb's Theorem: } \Box(\Box P \rightarrow P) \rightarrow \Box P$$

In the version of the 5&10 problem I gave, "P" is " $A_0=5 \rightarrow U_0=5 \& [A_0=10 \rightarrow U_0=0]$ ". Supposing it is provable, the agent will eventually find it, and return 5 in fact. This makes the sentence true, since $A_0=5$ and $U_0=5$, and $A_0=10$ is false and so implies anything, including $U_0=5$.

The agent can (given enough time) prove all of this, and so proves " $A_0=5 \rightarrow U_0=5 \& [A_0=10 \rightarrow U_0=0]$ " in fact, and takes the 5.

We call this a "spurious proof": the agent takes the \$5 because it can prove that if it takes the \$10 it has low value, because it takes the \$5.

It sounds circular, but sadly, is logically correct.

More generally, when working in less proof-based settings, we refer to this as a problem of spurious counterfactuals.

The general pattern is: counterfactuals may spuriously mark an action as not being very good. This makes the AI not take the action. Depending on how the counterfactuals work, this may remove any feedback which would "correct" the problematic counterfactual; or, as we saw with proof-based reasoning, it may actively help the spurious counterfactual be "true".

Note that, because the proof-based examples are of significant interest to us, "counterfactuals" actually have to be counterlogicals; we sometimes need to reason about logically impossible possibilities.

This rules out most existing accounts of counterfactual reasoning.

People tend to respond to all this by suggesting that there will always be some uncertainty. An agent may know its source code perfectly, but it can't perfectly know the hardware it is running on.



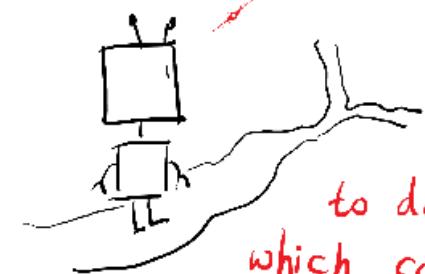
"An electron seems to be out of place"

Does adding a little uncertainty solve the problem?

Often not.

- The proof of the spurious counterfactual often still goes through; if you think you are in a 5&10 problem with 95% certainty, you can have the usual problem within that 95%.
- Adding uncertainty to make counterfactuals well-defined doesn't get you any guarantee that the counterfactuals will be reasonable.
Hardware failures aren't often what you want to expect when considering alternate actions.

Consider this scenario:



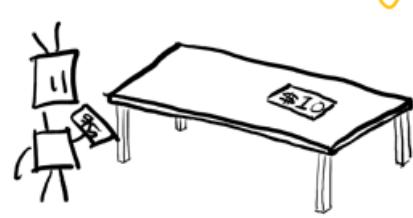
You are confident that you almost always take the left path. However, it is possible (though unlikely) for a cosmic ray to damage your circuits, in which case you could go right -- but you would then be insane, which would have many other bad consequences.

If this reasoning in itself is why you always go left, you've gone wrong.

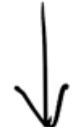
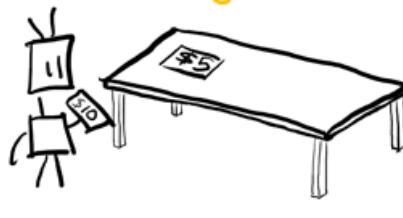
Although simply ensuring the agent has some uncertainty about its actions doesn't ensure the counterfactual expectations are in any way reasonable, we can instead ensure that the agent actually takes each action with some probability.

This strategy is called ϵ -exploration,
(that's "epsilon-exploration")

ϵ -exploration ensures that if an agent plays similar games on enough occasions, it can eventually learn realistic counterfactuals (modulo a concern of realizability which we will get to later).



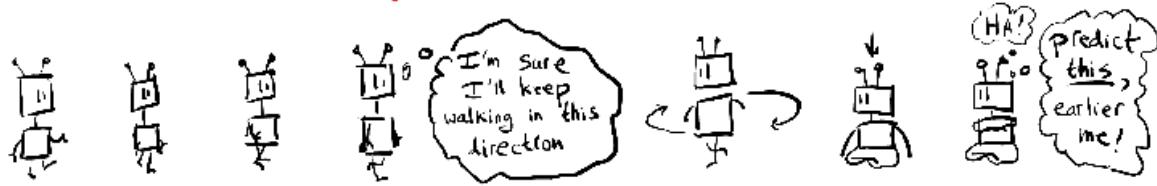
gain \$5



gain \$10

ϵ -exploration only works if it ensures that the agent itself can't predict whether it is about to ϵ -explore.

In fact, a good way to implement ϵ -exploration is via the rule "if the agent is too sure about its action, it takes a different one".

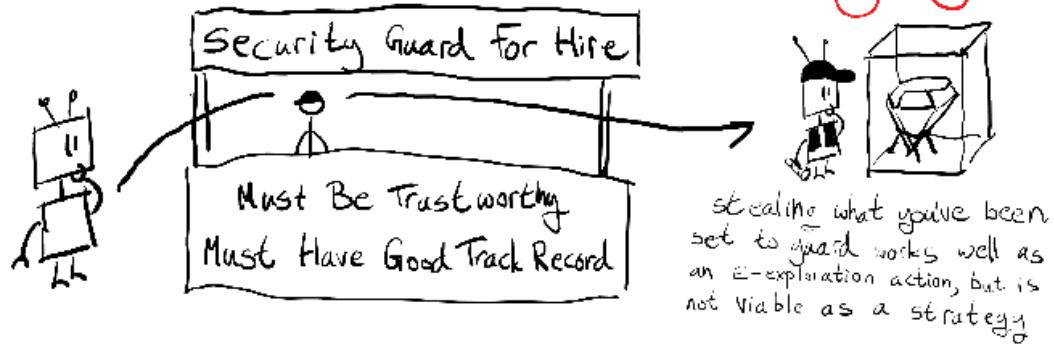


From a logical perspective, the unpredictability of ϵ -exploration is what prevents the problems we've been discussing.

From a learning-theoretic perspective, if the agent could know it wasn't about to explore, then it could treat that as a different case — failing to generalize lessons from its exploration.

This gets us back to a situation where we have no guarantee that the agent will learn better counterfactuals; exploration may be the only source of data for particular acts, so, we need to force the agent to use it or it may not learn.

However, even ϵ -exploration does not seem to get things exactly right.

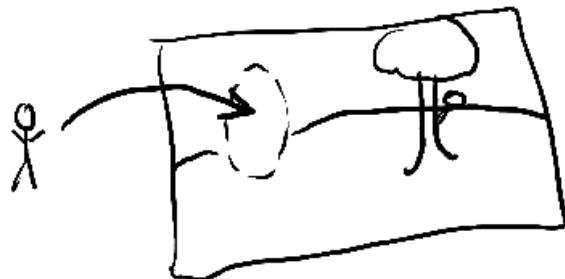


Observing the result of ϵ -exploration shows you what happens if you take an action unpredictably; the consequences of taking that action as part of business-as-usual may be different.

So, if we don't learn counterfactuals from ϵ -exploration, it seems we have no guarantee of learning realistic counterfactuals at all.

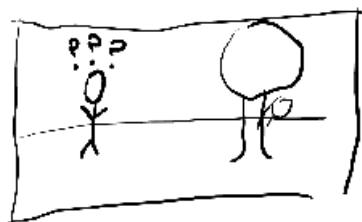
But, if we do learn from ϵ -exploration, it appears we still get things wrong in some cases.

But writing down examples of "correct" counterfactual reasoning doesn't seem hard from the outside!



Maybe that's because from "outside" we always have a Dualistic perspective. We are in fact sitting outside of the Problem, and we've defined it as a function of an agent.

However, an agent can't solve the problem in the same way from inside. From its perspective, there is no objective fact of its functional relationship with the environment.



(This is why "counterfactuals" are called what they are called, after all.)

Are we just fooling ourselves due to the way we set up decision problems, then? Are there no "correct" counterfactuals?

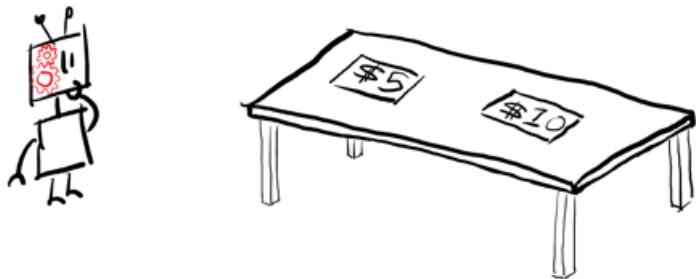
Well, maybe we are fooling ourselves. But, there is still something we are confused about!

"Counterfactuals are subjective, invented by the agent" doesn't dissolve the mystery. There is something intelligent agents do, in the real world, to make decisions.

So, I'm not talking about agents who know their own actions because I think there's going to be a big problem with intelligent machines inferring their own actions in the future.



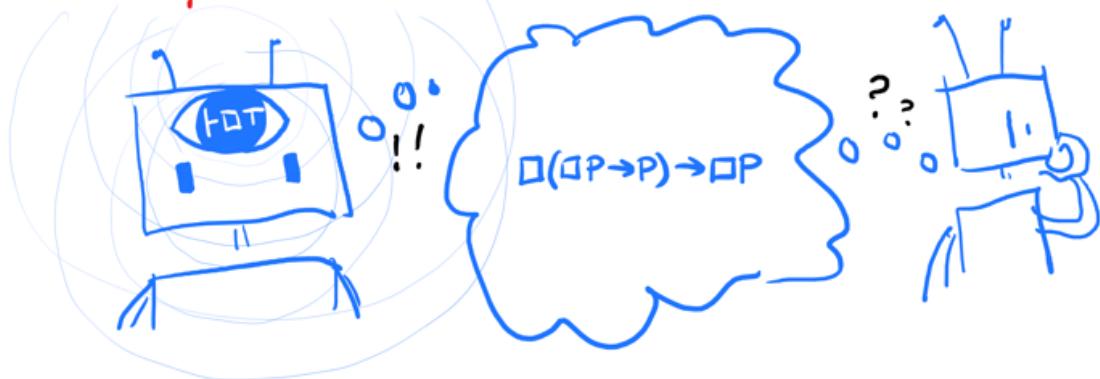
Rather, the possibility of knowing your own actions illustrates something confusing about determining the consequences of your own actions ; a confusion which shows up even in a very simple case, where everything about the world is known, and you just need to choose the larger pile of money.





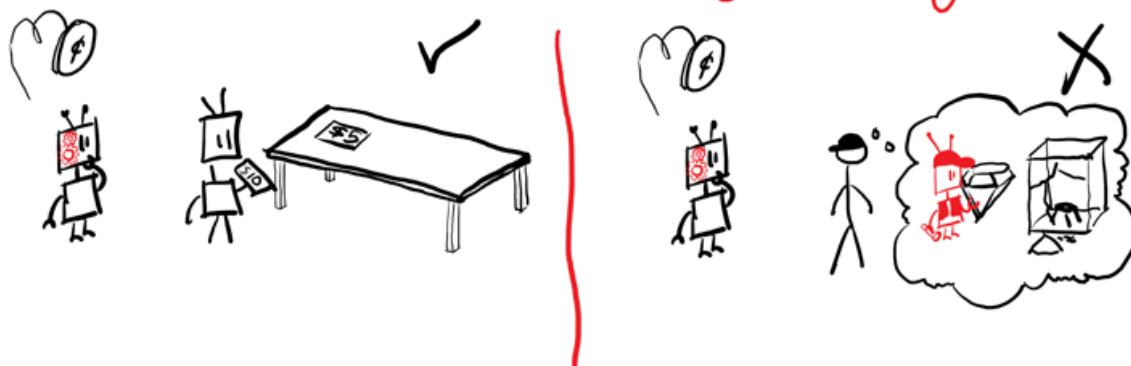
This is hard to avoid given the Bayesian assumption of logical omniscience — a probability distribution assigns probability 1 to any fact which is logically true. So, if a Bayesian agent knows its own source code, then it should know its own action.

However, realistic agents who are not logically omniscient may still have the same problem — logical omniscience forces the issue, but denying it does not prevent the issue.



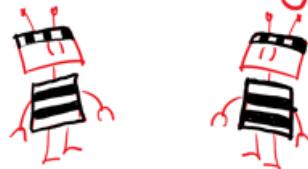
As we discussed, ϵ -exploration can fix the issue in many cases —

however, it makes us go wrong when the results of exploring randomly differ from the results of acting reliably.

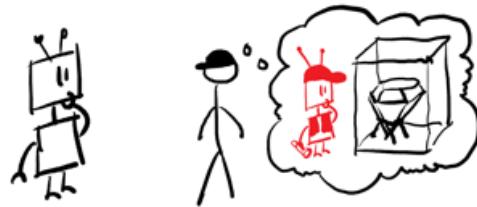


Examples which go wrong in this way seem to involve another part of the environment whose behavior imitates your own — such as another agent very similar to yourself, or, a sufficiently good model/simulation of you.

agent very similar
to yourself

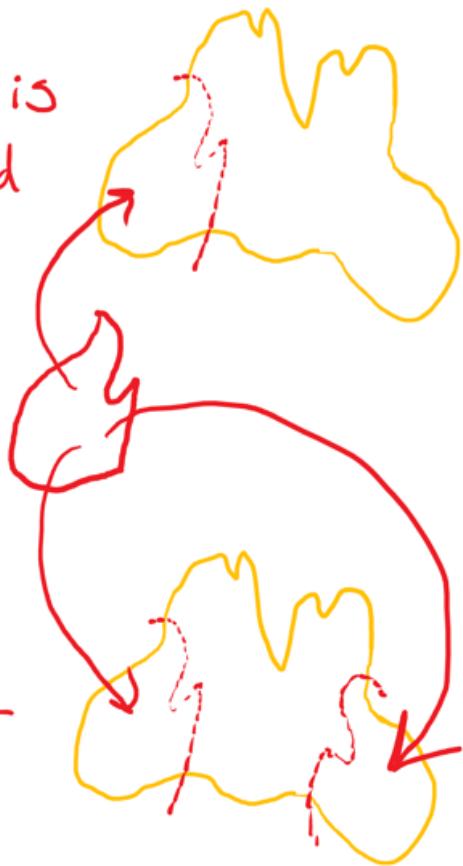


agent who models you

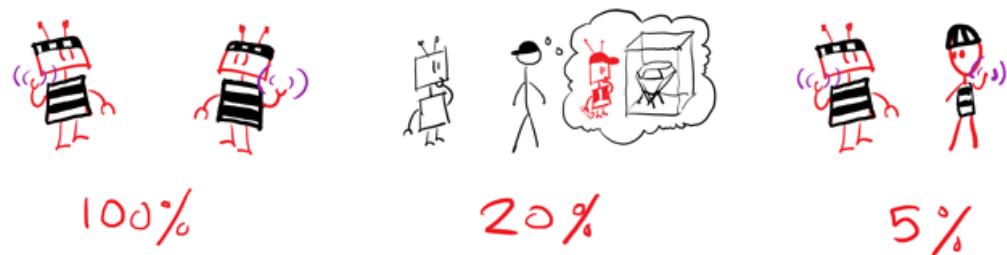


These are called Newcomblike problems.

If the 5&10 problem is about cutting a you-shaped piece out of the world so that the world can be treated as a function of your action, Newcomblike problems are about what to do when there are several approximately you-shaped pieces.



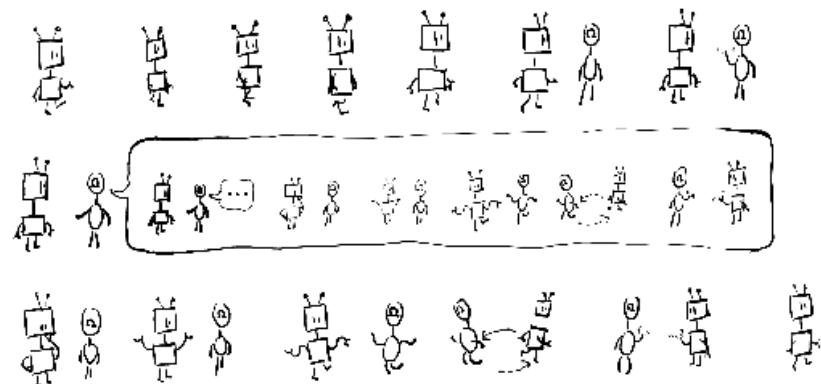
One idea is that exact copies should be treated as 100% under your "logical control".



For approximate models of you, or merely similar agents, control should drop off sharply as logical correlation decreases.

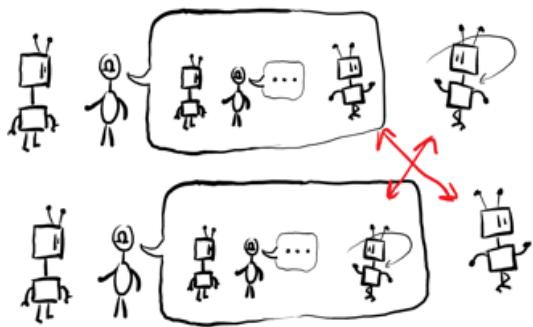
(But how does this work??)

There's another reason Newcomblike Problems are difficult, beyond questions of logical control.



If there is something which can predict you, it might tell you its prediction, or related information.

Now it matters what you do in response to various things you could find out.



If you do the opposite of whatever you're told, then it isn't possible for the scenario to be set up in the first place — the Predictor can't both be good and talk to you.

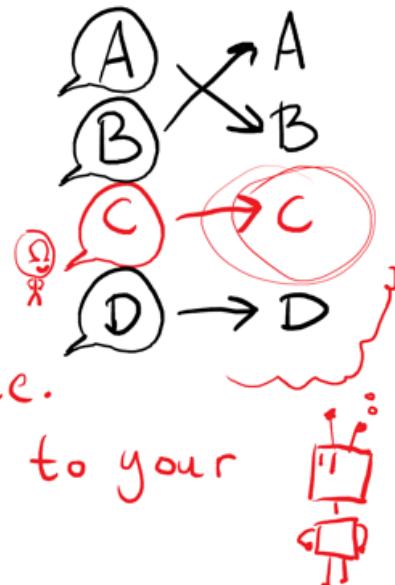
However, if there's a case where you act as predicted, a predictor can make that happen.



So, on the one hand, a powerful predictor can control you by selecting between the consistent possibilities.

On the other hand, you are the one who chooses your own pattern of responses in the first place.

So, you can set them up to your best advantage.



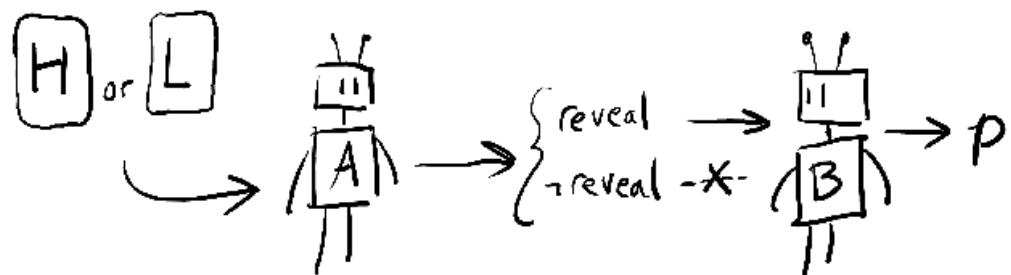
So far, we had been discussing action counterfactuals — how to anticipate consequences of different choices.

This discussion of controlling your response introduces the observation counterfactual — imagining what the world would be like if different facts had been observed.

Observation counterfactuals may be relevant even if the predictor doesn't talk to you. Let's look at a detailed example.

Consider the following game:

Alice receives a card at random which is either High or Low. She may reveal the card if she wishes.



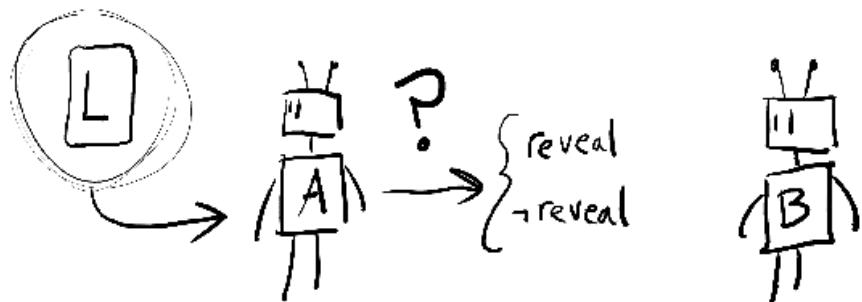
Bob then gives his probability p that Alice has a high card.

Alice always loses p^2 dollars. Bob loses p^2 if the card is low, and $(1-p)^2$ if the card is high.

Bob has a proper scoring rule, so does best by giving his true belief.

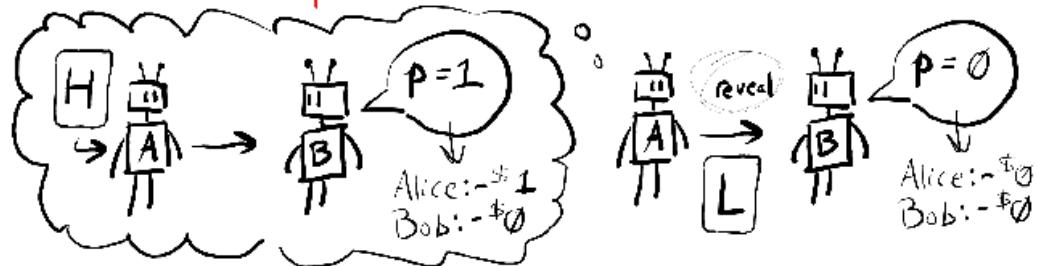
Alice just wants Bob's belief to be as much toward "low" as possible.

Suppose Alice will play only this one time. She sees a low card. Bob is good at reasoning about Alice, but is in the next room so can't read any tells.



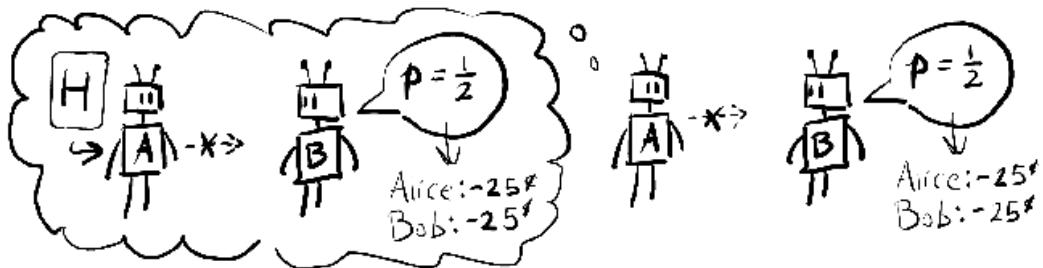
Should Alice reveal her card?

Since Alice's card is low, if she shows it to Bob, she will lose no money, which is the best possible outcome.



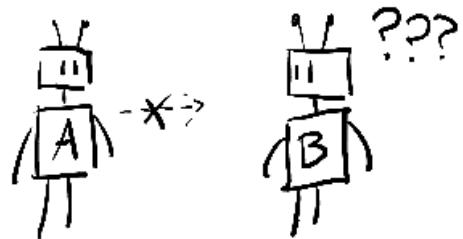
However, this means that in the counterfactual world where Alice sees a high card, she wouldn't be able to keep the secret -- she might as well show her card in that case too, since her reluctance to show it would be as reliable a signal of "high".

On the other hand, if Alice doesn't show her card, she loses $25^{\$}$ -- but then she can use the same strategy in the other world, rather than losing $\$1$.



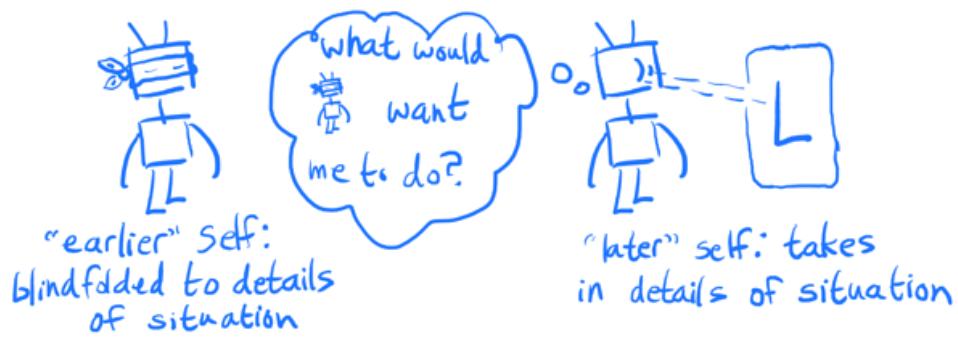
So, before playing the game, Alice would want to commit to not reveal ; this makes expected loss $25^{\$}$, whereas the other strategy has expected loss $50^{\$}$.

By taking observation counterfactuals into account, Alice is able to keep secrets — without them, Bob could perfectly infer her card from her actions.



This game is equivalent to the decision problem called counterfactual mugging.

Updateless Decision Theory (UDT) is one proposed decision theory which does this.



UDT solves such problems by recommending that the agent do whatever would have seemed wisest before -- whatever your earlier self would have committed to do.

UDT is an elegant solution to a fairly broad class of decision problems, but it only makes sense if the earlier self can foresee all possible situations. This makes sense in a Bayesian setting, where the prior already contains all possibilities within itself. However, there may be no way to do this in a realistic

embedded setting. An agent has to be able to think of new possibilities, meaning that its earlier self doesn't know enough to make all the decisions.

However, to explain why this might be, we will have to move on to embedded world models.

Embedded World-Models

Crossposted from the [AI Alignment Forum](#). May contain more technical jargon than usual.

(A longer text-based version of this post is also available on MIRI's blog [here](#), and the bibliography for the whole sequence can be found [here](#))

(Edit: This post had 15 slides added on Saturday 10th November.)

[Embedded World-Models]

agent is smaller than the environment



[Abram Demski and Scott Garrabrant]

An agent which is larger than its environment can:



- Hold an exact model of the environment in its head.
- Think through the consequences of every potential course of action.
- If it doesn't know the environment perfectly, hold every possible way the environment could be in its head, as is the case with Bayesian uncertainty.

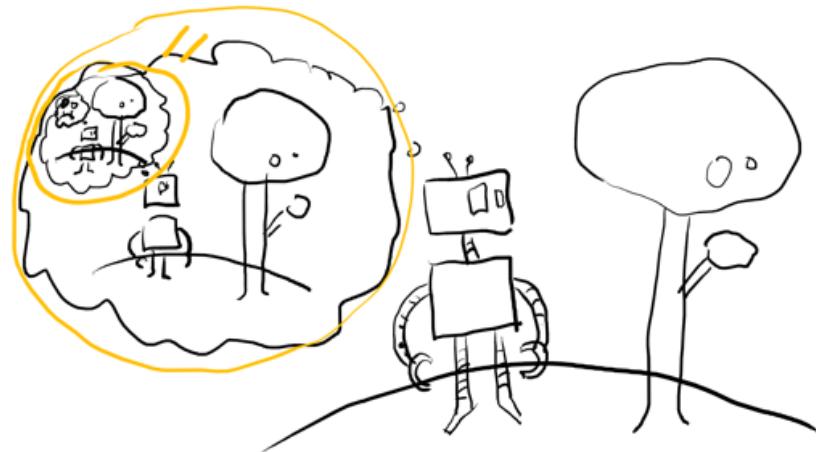


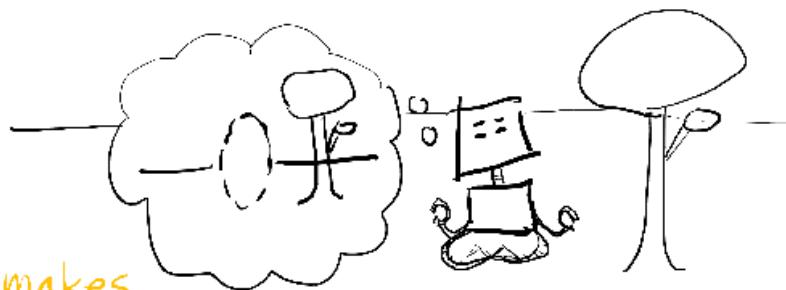
... or

All of these are typical of notions of rational agency.

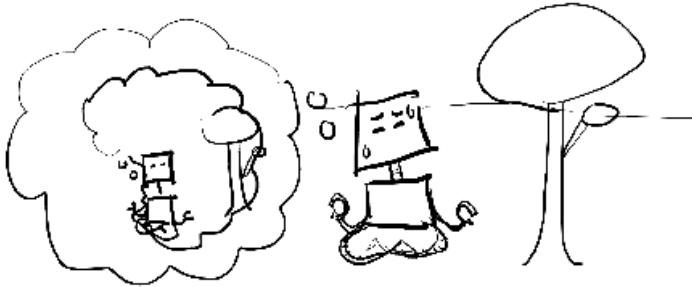
An embedded agent can't do any of those things, at least not in a straightforward way.

One difficulty is that, since it is hard to separate oneself from the world, one would have to have a complete self-model.

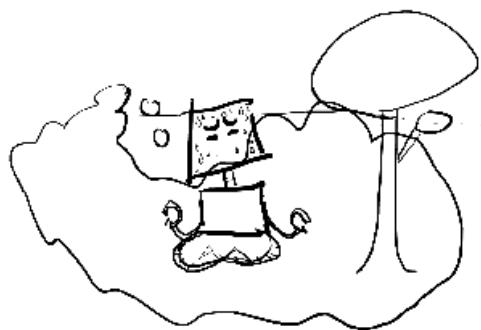




This makes
paradoxes of
self-reference an
obstacle to us.



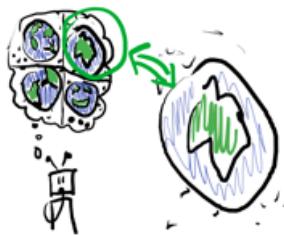
As if representing
the rest of the world
weren't already enough
of a difficulty.



Embedded World-Models have to represent the world in a way more appropriate for embedded agents. Problems in this cluster include:

- world not in hypothesis Space
("realizability"/"grain of truth" problem)
- logical uncertainty
- high-level models
- multi-level models
- ontological crises
- agent must be in world-model (Naturalized Induction)
- anthropic reasoning

In a Bayesian setting, where an agent's uncertainty is quantified by a probability distribution over possible worlds, a common assumption is "realizability": the true underlying environment which is generating the observations is assumed to have at least some probability in the prior.



In game theory, this same property is described by saying a prior has a "grain of truth".

(It should be noted, though, that there are additional barriers to getting this property in a game-theoretic setting; so, in their common usage cases, "grain of truth" is technically demanding while "realizability" is a technical convenience.)

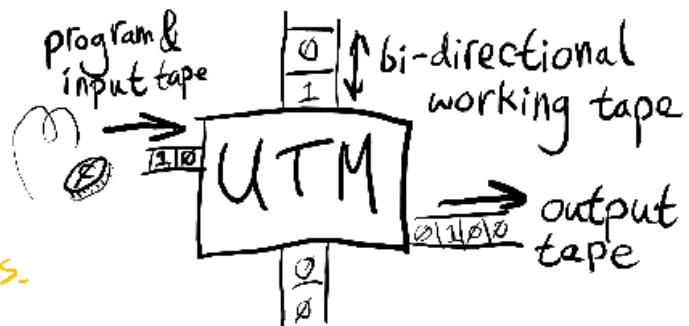
Realizability is not totally necessary in order for Bayesian reasoning to make sense. If you think of a set of hypotheses as "experts", and the current posterior probability as how much you "trust" each expert, then learning according to Bayes' Law ensures a relative bounded loss property.

$$P(h|e) = \frac{P(e|h) P(h)}{P(e)}$$

Specifically, if you use prior π , the amount worse you are in comparison to each expert h is at most $\log(\pi(h))$, since you assign at least probability $\pi(h) \cdot h(e)$ to seeing a sequence of evidence e .

Intuitively, $\pi(h)$ is your initial trust in expert h , and in each case where it is even a little bit more correct than you, you increase your trust accordingly; the way you do this ensures you assign an expert probability 1 and hence copy it precisely before you lose more than $\log(\pi(h))$ compared to it.

The prior AIXI is based on is the Solomonoff prior. It is defined as the output of a universal Turing machine (UTM) whose inputs are coin-flips.



In other words, feed a UTM a random program.

Normally, you'd think of a UTM as only being able to simulate deterministic machines. Here, however, the initial inputs can instruct the UTM to use the rest of the infinite input tape as a source of randomness to simulate a stochastic Turing machine.

Combining this with the previous idea about viewing Bayesian learning as a way of allocating "trust" to "experts" which meets a bounded loss condition, we can see the Solomonoff prior as a kind of ideal machine learning algorithm which can learn to act like any algorithm you might come up with, no matter how clever.

It is assuming all possible algorithms are computable, not that the world is.

For this reason, we shouldn't necessarily think of AIXI as "assuming the world is computable", even though it reasons via a prior over computations.

It's getting bounded loss on its predictive accuracy as compared with any computable predictor.

However, lacking realizability can cause trouble if you are looking for anything more than bounded-loss predictive accuracy.

- posterior can oscillate forever
- probabilities may not be calibrated
- estimates of statistics such as the mean may be arbitrarily bad
- estimates of latent variables may be bad
- identification of causal structure may not work

So, does AIXI perform well without a realizability assumption?

We don't know. Despite getting bounded loss for predictions without realizability, existing optimality results for its actions require a realizability assumption.

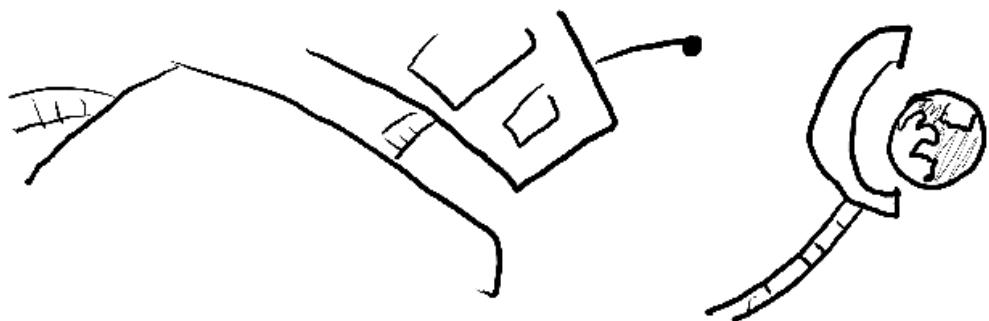
First, if the environment really is sampled from the Solomonoff distribution, AIXI gets the maximum expected reward. But, this is fairly trivial; it is essentially the definition of AIXI.

Second, if we modify AIXI to take somewhat randomized actions (Thompson sampling), there is an asymptotic optimality result for environments which act like any stochastic Turing machine. So, either way, realizability was assumed in order to prove anything.

(Jan Leike, Nonparametric General Reinforcement Learning)

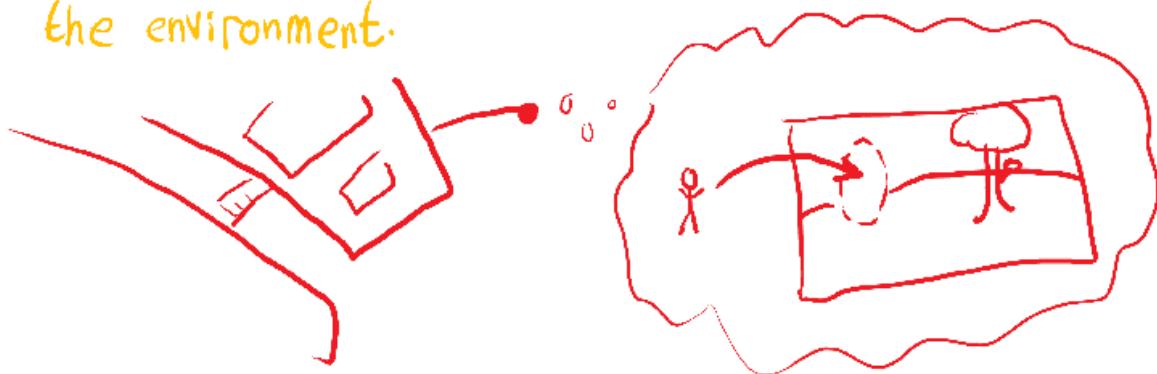
But, the concern I'm pointing at is not "the world might be uncomputable, so we don't know if AIXI will do well" -- that is more of an illustrative case.

The concern is that AIXI is only able to define intelligence/rationality by constructing an agent much much bigger than the environment which it has to learn about and act within.

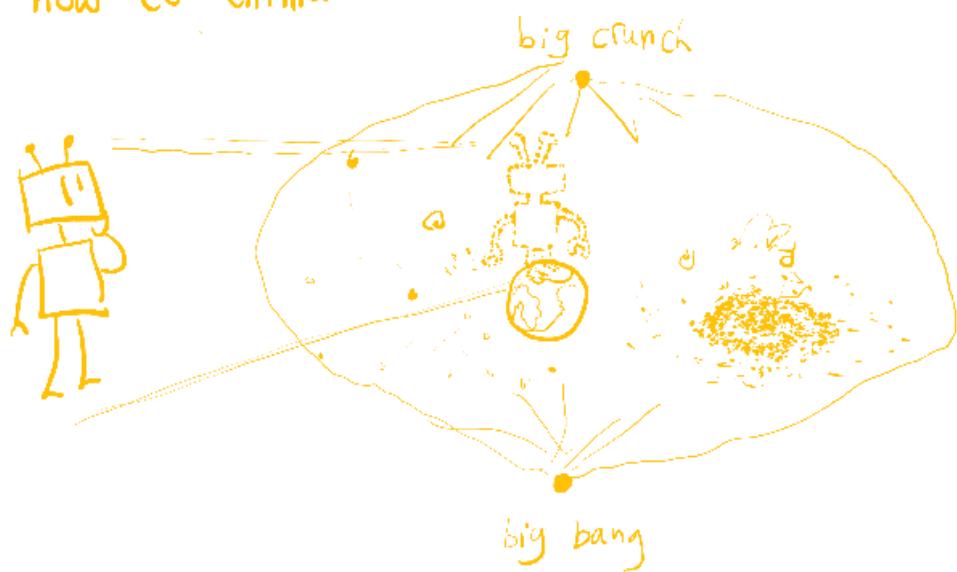


Laurent Orseau provides a way of thinking about this in Space-Time Embedded Intelligence.

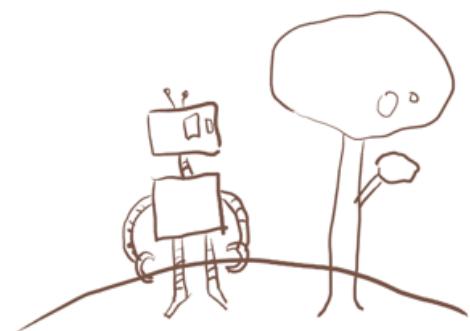
However, his approach defines the intelligence of an agent in terms of a sort of super-intelligent designer who thinks about reality from outside, selecting an agent to place into the environment.



Embedded agents don't have the luxury
of stepping outside of the universe to think
about how to think.



What we would like would be a theory of rational belief for embedded agents which provides similarly strong foundations as Bayesianism provides for Dualistic agents.



$$P(h|e) = \frac{P(e|h) P(h)}{P(e)}$$

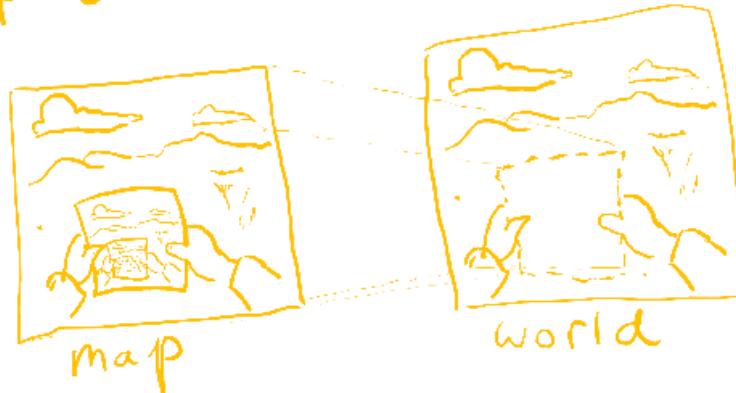


One major obstacle such a theory must deal with is self-reference.

Paradoxes of self-reference such as the Liar paradox make it not just wildly impractical, but in a certain sense impossible for an agent's world-model to accurately reflect the world.

The Liar paradox concerns the status of the self-referential sentence "This sentence is not true.". If it were true, it must be false; and if not true, it must be true.

The difficulty comes in part from trying to draw a map of territory which includes the map itself.



This is fine if the world "holds still" for us; but, because the map is in the world, different maps create different worlds.

We can set up a Liar-paradox-like situation by supposing that we're trying to make an accurate map of the final route of a road which is currently under construction, but we know the construction will proceed so as to disprove whatever map we make.

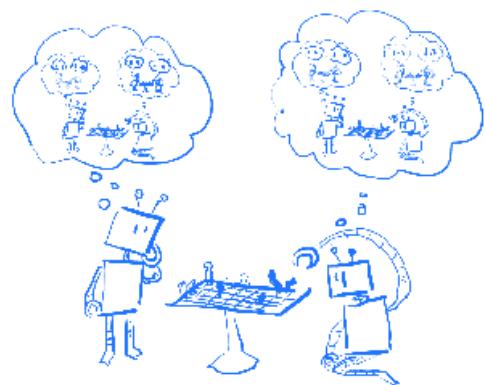


Problems of this kind become relevant for decision-making in the theory of games; a simple game of rock-paper-scissors sets up a Liar paradox, to the extent that the players can predict each other better than chance and are trying to win.

Game theory solves this type of problem with game-theoretic equilibria. But, as I'll explain, realizability issues end up coming back in a different way.

Grain of Truth

I mentioned that the problem of realizability takes on a different character in the context of game theory.



In a machine learning setting, realizability is a potentially unrealistic assumption, but can usually be assumed consistently nonetheless.

Due to the paradoxes which can so easily arise from games, the assumption itself may be inconsistent.

Because there are many agents, it is no longer possible to conveniently make an "agent" a thing which is larger than a world.

So, game-theorists are forced to investigate notions of rational agency which can handle a world which is large.

Unfortunately, this is done by splitting up the world into "agent" parts and "non-agent" parts, and handling the agents in a special way.



This is almost as bad as dualistic models of agency.



In rock-paper-scissors, the Liar paradox is resolved by stipulating that each player play each move with $\frac{1}{3}$ probability. If one player plays this way, then the other loses nothing by doing so.

This way of introducing probabilistic play to resolve would-be paradoxes of game theory is called a Nash equilibrium.

We can use Nash equilibria to prevent the assumption that the agents correctly understand the world they're in from being inconsistent. However, that works just by telling the agents what the world looks like. What if we want to model agents who learn about the world, more like AIXI?

The grain of truth problem is the problem of formulating a reasonably broad prior probability distribution which would allow agents playing games to place some positive probability on each other's true (probabilistic) behavior, without knowing it precisely from the start.

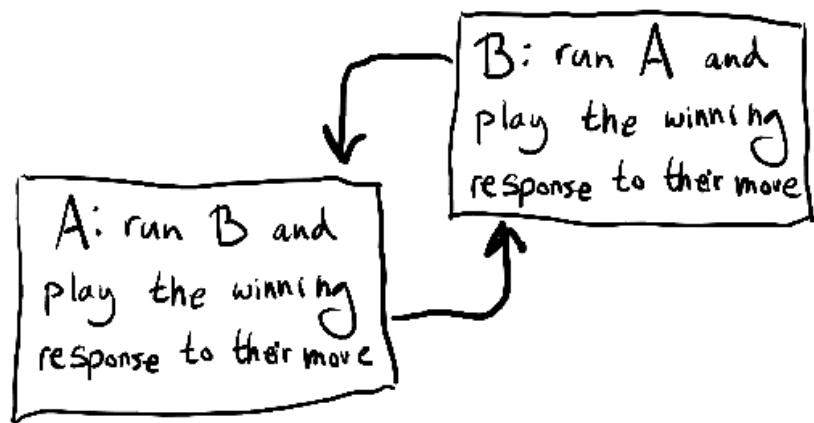


Until recently, known solutions to the problem were quite limited. Reflective Oracles: A Foundation for Classical Game Theory (Benja Fallenstein, Jessica Taylor, & Paul F. Christiano) provides a very general solution (also see A Formal Solution to the Grain of Truth Problem by Jan Leike, Jessica Taylor, & Benja Fallenstein).

You might think stochastic Turing machines can represent Nash equilibria just fine. But, if you're trying to produce them as a result of reasoning about other agents, you'll run into trouble.



If both agents model each other's computation and try to run it to see what each other do, you've just got an infinite loop.



There are some questions Turing machines just can't answer; especially, about the behavior of Turing machines.

The halting problem is the classic example.

Turing studied "oracle machines" to examine what would happen if we could answer such questions.

An oracle is like a book containing some answers to questions which we were unable to answer before.

But ordinarily, we get a hierarchy: type B machines have the answers about type A; type C machines have the answers about types A and B (and so on); but, no machines have answers about their own type.

The diagram illustrates the Oracle Argument through three levels of machines:

- Type A machines (represented by a computer icon) know about halting for type B machines (represented by a book icon). This is labeled "Ah. B-Halting".
- Type B machines (represented by a computer icon) know about halting for type A machines and type C machines. This is labeled "Ah. Halting" and "B-halting??".
- Type C machines (represented by a computer icon) know about halting for type B machines and type A machines. This is labeled "Ah. B-Halting", "C-halting??", and "B-halting".

Ellipses at the top indicate that there are more levels above C, and arrows point from each machine icon to its corresponding knowledge level.

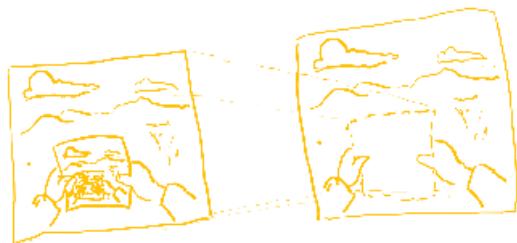


Reflective oracles twist the Turing universe in on itself to avoid this hierarchy, answering questions about the behavior of machines with access to the very same oracle.

The usual paradoxes which this would create are avoided by adding a small amount of randomness, to fuzz things out if e.g. someone tries to do the opposite of what they themselves do.

So R.O. machines are stochastic, but, they're more powerful than regular stochastic Turing machines.



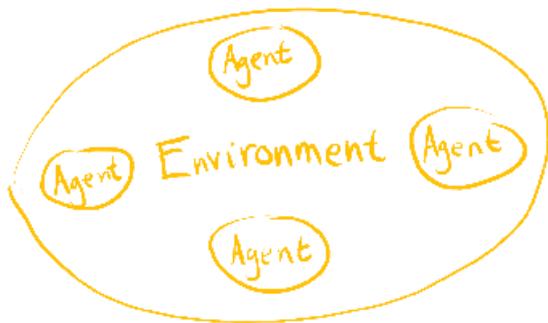


That's how R.O.s address the earlier-mentioned problems of a map that's itself part of the territory:
randomize.



Reflective oracles also solve the problem with game-theoretic notions of rationality I mentioned earlier: it allows agents to be reasoned about in the same manner as other parts of the environment, rather than being a special case to be handled differently.

They're all just computations-with-oracle-access.



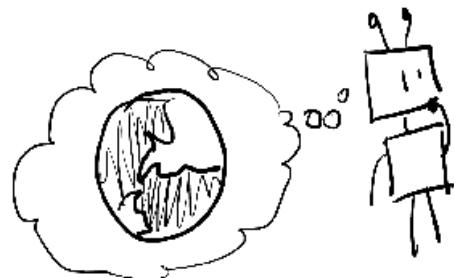
However, models of rational agents based on reflective oracles still have several major limitations. One of these is that agents are required to have infinite processing power, just like AIXI, and so are assumed to know all of the consequences of their own beliefs.

In fact, knowing all the consequences of your beliefs -- a property known as logical omniscience -- turns out to be rather core to classical Bayesian rationality.

Logical Uncertainty

So far, I've been talking in a fairly naive way about the agent having beliefs about hypotheses, and the real world being (or not being) in the hypothesis space.

It isn't really clear what any of that means.



Depending on definitions, it may actually be quite possible for an agent to be smaller than the world and yet "contain" the right world-model — it might know the true physics and initial conditions, but only be capable of inferring their consequences very approximately.



Realistic as this scenario may be, it is not in line with what it usually means for a Bayesian to know something — a Bayesian knows the consequences of all its beliefs.

$$\{P(A)=1\} + A \text{ implies } B \Rightarrow \{P(B)=1\}$$


Uncertainty about the consequences of your beliefs is logical uncertainty. We would like some notion of boundedly rational beliefs about uncertain consequences.



This requires a combined theory of logic — reasoning about implications — and probability — degrees of belief.

Logic and probability theory
are two great triumphs in the
codification of rational thought.

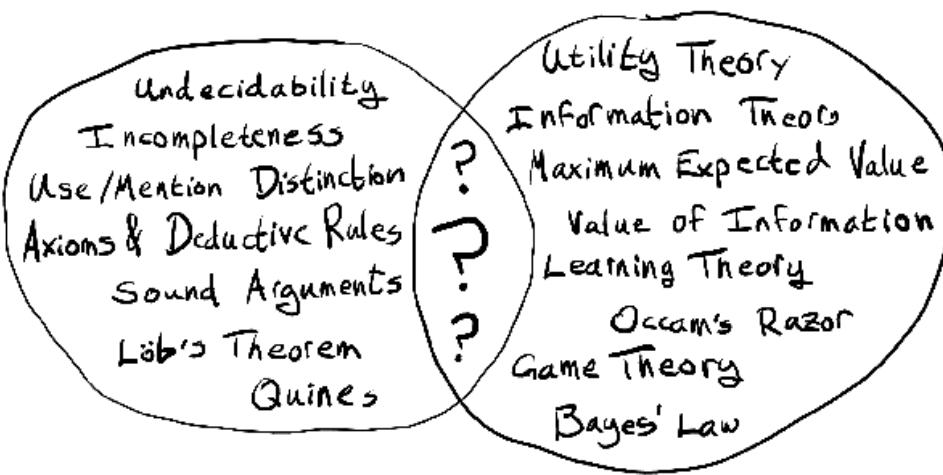
Logic provides
the best tools for
thinking about
self-reference.

Undecidability
Incompleteness
use/Mention Distinction
Axioms & Deductive Rules
Sound Arguments
Löb's Theorem
Quines

Probability provides the
best tools for thinking
about decision-making.

Utility Theory
Information Theory
Maximum Expected Value
Value of Information
Learning Theory
Occam's Razor
Game Theory
Bayes' Law

However, the two don't work together
as well as you may think.



They may seem superficially compatible,
since probability theory is an extension
of Boolean logic.

conjunction $P(A \& B)$

disjunction $P(A \text{ or } B) = P(A) + P(B) - P(A \& B)$

negation $P(\text{not } A) = 1 - P(A)$

However, Gödel's first incompleteness theorem shows, any sufficiently rich logical system is incomplete — ie., not only does it fail to decide every sentence as true or false, it further has no computable extension which manages to do so.

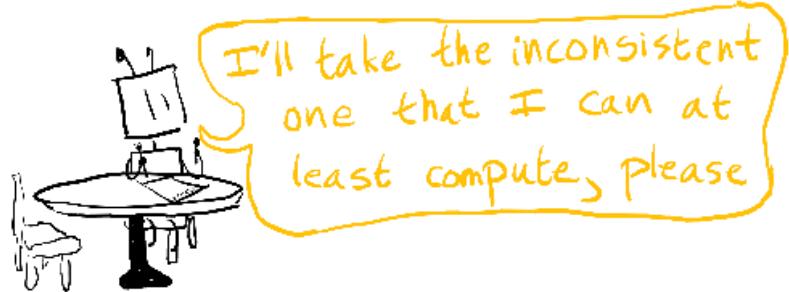
(See the post An Untrollable Mathematician Illustrated for more illustration of how this messes with probability theory.)

This also applies to probability distributions: no computable distribution can assign probabilities in a way that's consistent with a sufficiently rich theory.

This forces us to choose:

- use an uncomputable distribution,
or,
- use one which is inconsistent.

Sounds like an easy choice, right?



I'll take the inconsistent
one that I can at
least compute, please

After all, we're trying to develop a theory of logical non-omniscience here.

We can just continue to update on facts which we prove, bringing us closer and closer to consistency.

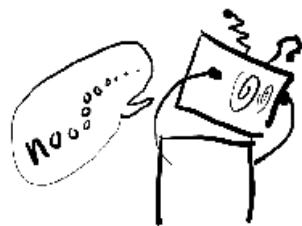
Unfortunately, this doesn't work out so well, for reasons which connect back to realizability.



Remember, there are no computable probability distributions consistent with all consequences of rich theories.

So, our non-omniscient prior doesn't even contain a hypothesis which is correct.

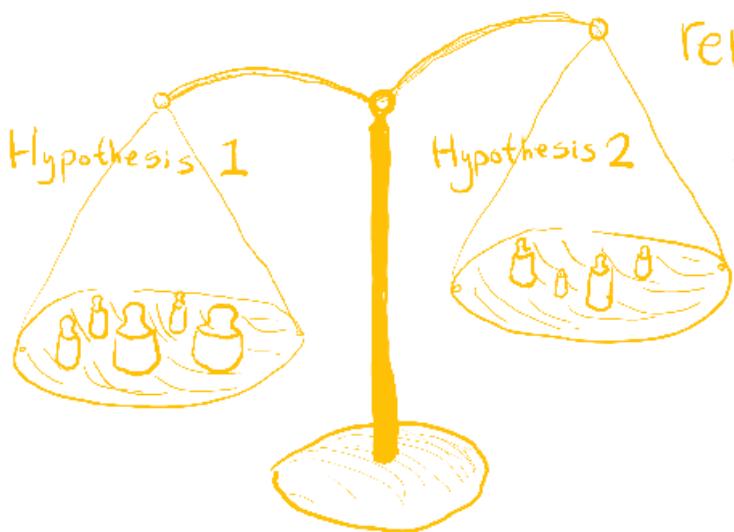
This causes pathological behavior as we condition on more and more true mathematical facts. Beliefs wildly oscillate rather than approaching reasonable estimates.



$$\begin{aligned}1+1 &= 2 \\2+1 &= 3 \\1+2 &= 3 \\2+2 &= 4 \\1+3 &= 4\end{aligned}$$

Taking a Bayesian prior on mathematics, and updating on whatever we prove, does not seem to capture mathematical intuition and heuristic conjecture very well — unless we restrict the domain and craft a sensible prior.

Probability is like a scale, with worlds as weights. An observation eliminates some of the possible worlds, removing weights and shifting the balance of beliefs.



Logic is like a tree, growing from
the seed of axioms.



The process of growth is never
complete; you never know all the
consequences of each belief.

Not knowing the consequences of a belief is like not knowing where to place the weights on the scales of probability.



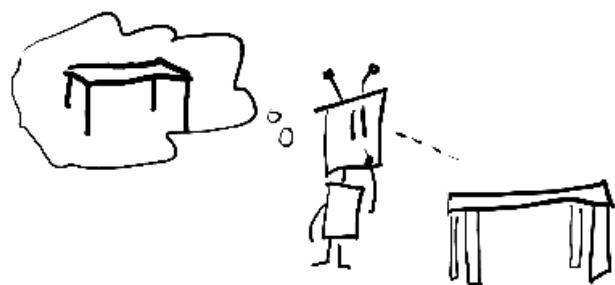
If we put weight in both places until a proof rules one out, the beliefs just oscillate forever rather than doing anything useful.

This grapples directly with the problem of a world which is larger than you.

Any computable beliefs about logic must have left out something, since the tree will grow larger than any container.

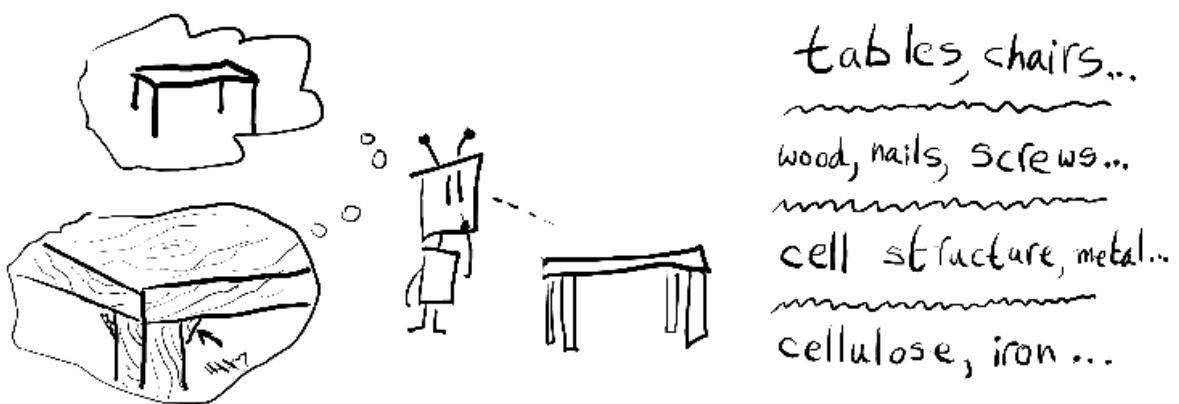


High-Level / Multi-Level World-Models

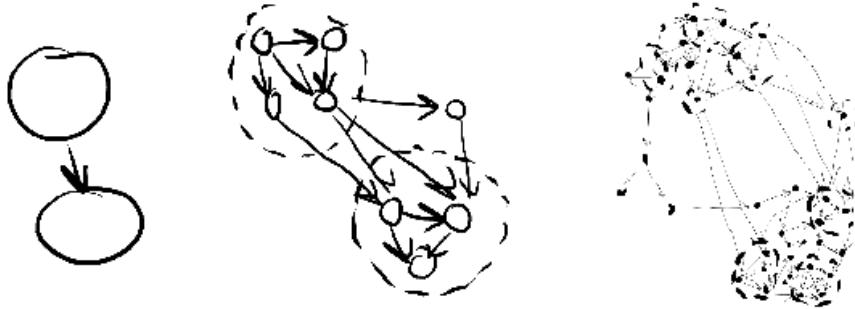


Because the world is bigger than you, you need to be able to use high-level world models: models which involve things like tables and chairs.

This is related to the classical symbol grounding problem; but, since we want a formal analysis which increases our trust in some system, the kind of model which interests us is somewhat different. This also relates to transparency and informed oversight: world-models should be made out of understandable parts.



Relatedly, there is the question of how high-level reasoning and low-level reasoning relate to each other and to intermediate levels: multi-level world models. Standard probabilistic reasoning doesn't provide a very good account of this sort of thing.



It's like you've got different Bayes nets which describe the world at different levels of accuracy, and processing power limitations force you to mostly use the less accurate ones, so you have to decide how to jump to the more accurate as needed. Also, the models at different levels don't

line up perfectly, so you have a problem of translating between them. Also, the models may have serious contradictions between them. This might be fine, since high-level models are understood to be approximations anyway; or, it could signal a serious problem in the higher- or lower-level models, requiring their revision.

This is especially interesting in the case of ontological crisis, in which

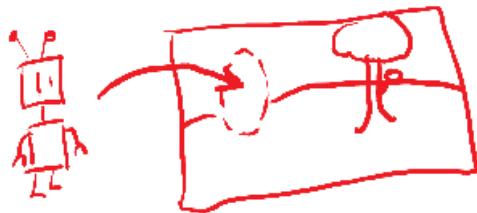
Objects in which we place value turn out not to be a part of "better" models of the world. It seems fair to say that everything humans value exists in high-level models only, which from a reductionistic perspective is "less real" than atoms and quarks.

However, because our values aren't defined on the low level, we are able to keep our values even when our knowledge of

the low level radically shifts. We would also like to be able to say something about what happens to values if the high level radically shifts.

Another critical aspect of embedded world models is that the agent itself must be in the model, since the agent seeks to understand the world, and the world cannot be fully separated from oneself. This opens the door to

difficult problems of self-reference and anthropic decision theory. Naturalized Induction is the problem of learning world-models which include yourself in the environment. This is challenging because (as Caspar Oesterheld has put it) there is a type mismatch between "mental stuff" and "physics stuff".

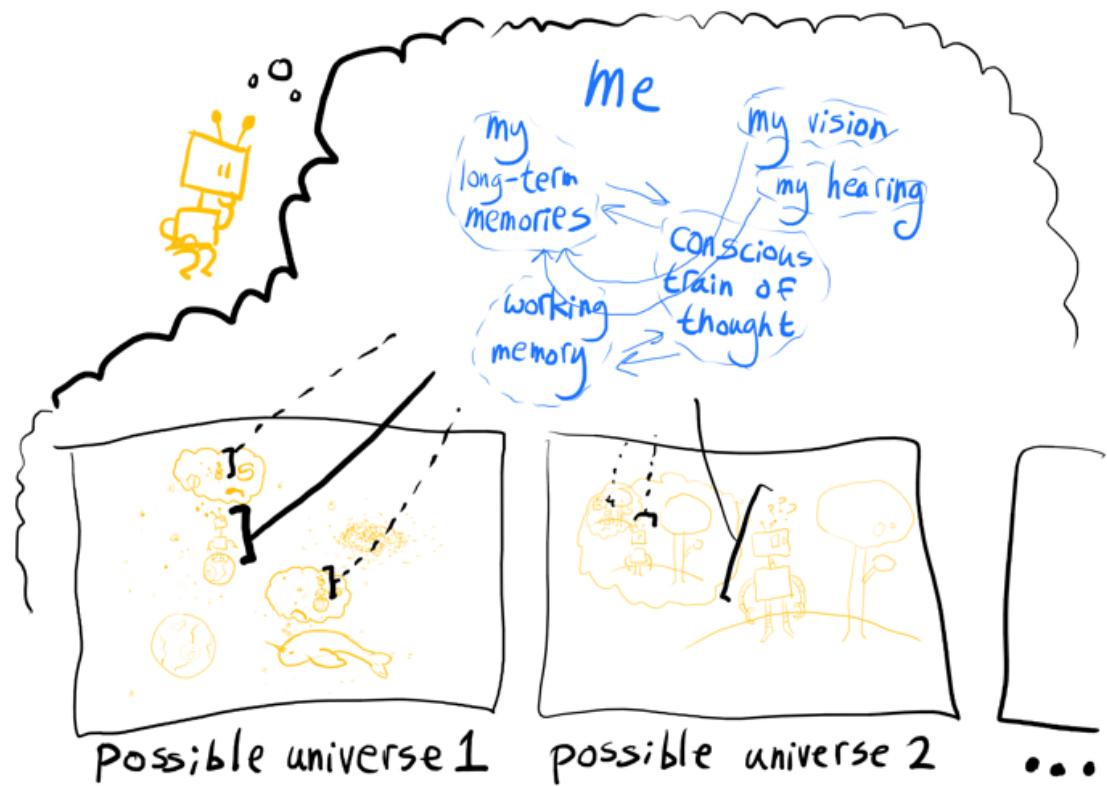


AIXI conceives of the environment as if it were made with a slot which the agent fits into.

We might intuitively reason in this way, but, we can also understand a physical perspective from which this looks like a bad model.

We might imagine instead that the agent separately represents:

- self-knowledge available to introspection
- hypotheses about what the universe is like
- a "bridging hypothesis" connecting the two



There are interesting questions of how this could work. There's also the question of whether this is the right structure at all. It's certainly not how I imagine babies learning.

Thomas Nagel would say this way of approaching the problem involves "views from nowhere"; each hypothesis posits a world as if seen from outside. This is perhaps a strange thing to do.



A particularly interesting aspect of the agent being in its own model of the environment is its need to reason about its future selves. There is a particular sort of trust an agent needs in its future selves to carry out plans; or, because the future is large, to

go beyond plans which can be conceived of at the moment — to pursue one's goals in a reasonable manner.

This is the subject of the next section, robust delegation.

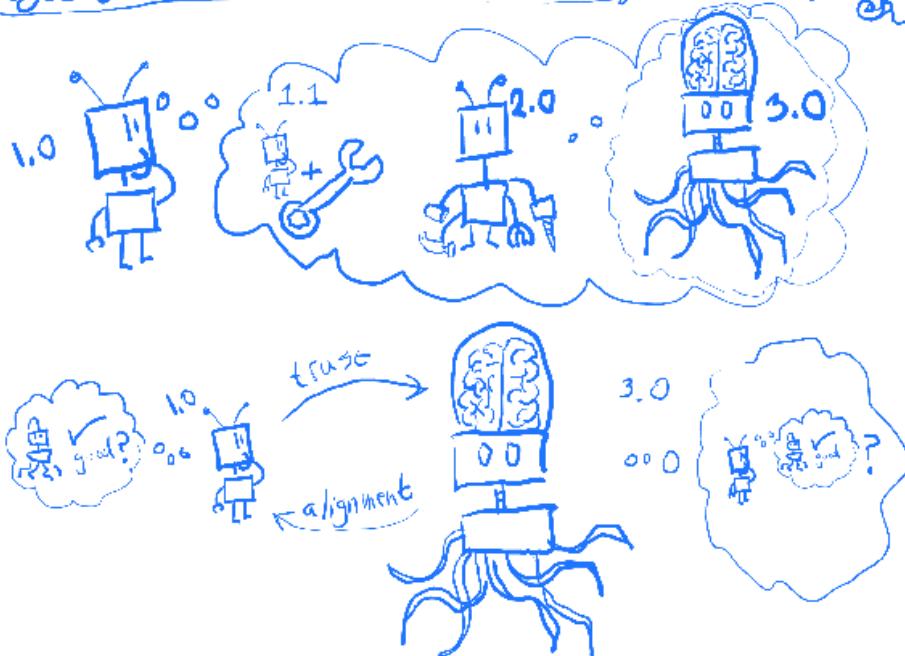
Robust Delegation

Crossposted from the [AI Alignment Forum](#). May contain more technical jargon than usual.

(A longer text-based version of this post is also available on MIRI's blog [here](#), and the bibliography for the whole sequence can be found [here](#))

[Robust Delegation]

agent can reason about itself and improve



[Abram Demski and Scott Garrabrant]

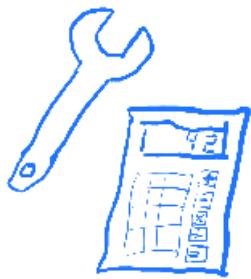
Because the world is big, the agent as it is may be inadequate to accomplish its goals, including in its ability to think.



Because the agent is made of parts, it can improve itself and become more

... improve ... - - - - -
capable.

Improvements can take many forms:



tools



successor
agents



just learning
and growing
over time

However, the successors/tools need to be more capable for this to be worthwhile.

This gives rise to a special type of principal/agent problem: how can you trust something as complicated as, or more complicated than, you?

principal:  human

 AI

 AI

agent:  AI

 Same
 later

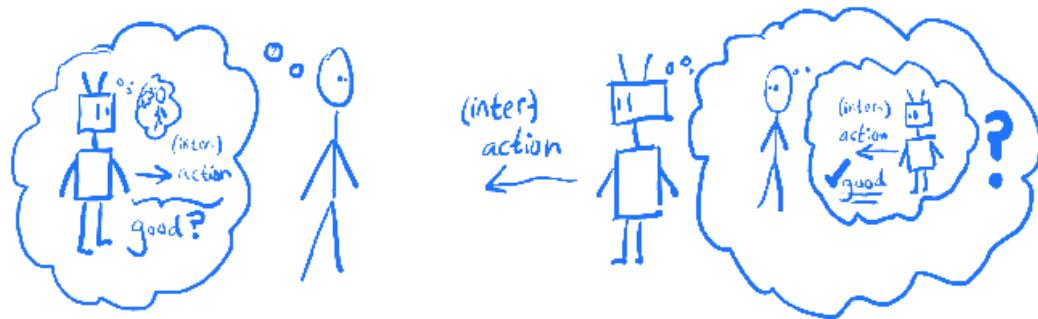
 AI
smarter AI
built by first

Alignment
Problem

Tiling
Agent
Problem

reflective stability
of goal system under
self-improvement

The problem is not (just) that the successor agent might be malicious. The problem is that we don't even know what it means not to be.



This problem seems hard from both points of view.

I want to emphasize that the view in which there are multiple forms of the problem is a dualistic view.

To an embedded agent, the future self is not privileged; it is just another part of the environment. There is no difference between making a successor and preserving your own goals.

So, when we talk about designing agents which are helpful to humans, it is not just about that. It is about the fundamental problem of being an agent that persists and learns over time.

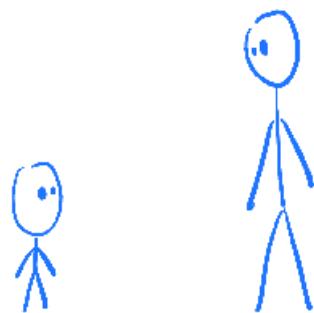
We call this cluster of problems

Robust Delegation:

- Vingeian reflection
- Tiling problem
- Averting Goodhart's Law
- Value Loading
- Corrigibility
- Informed Oversight

Vingeian Reflection

Imagine you are playing the CIRL game with a toddler:



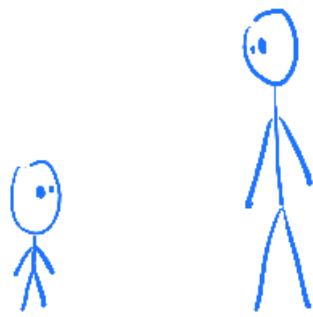
CIRL means Cooperative Inverse Reinforcement learning.

The idea behind CIRL is to define what it means for a robot to collaborate with a human.



The robot simultaneously tries to infer what the human wants while helping.

So, what if you're trying to help someone who is very confused about the universe?

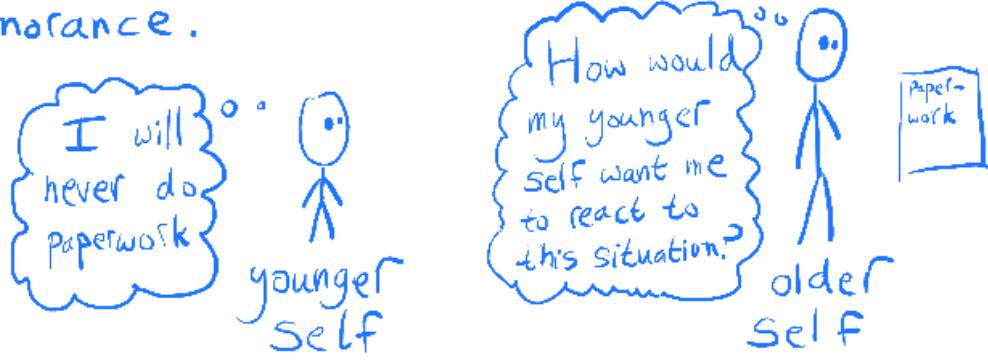


- From your standpoint, the toddler may be too irrational to be seen as optimizing anything.
- The toddler may have an ontology in which it is optimizing something, but you can see that ontology doesn't make sense.
- Maybe you notice that if you set up questions in the right way, you can make the toddler seem to want almost anything.

Part of the problem is that the "helping" agent has to be bigger in some sense in order to be more capable; but, this seems to imply that the "helped" agent can't be a very good supervisor for the "helper".



For example, updateless decision theory eliminates dynamic inconsistencies in decision theory by, rather than maximizing expected utility of your action given what you know, maximizing expected utility of reactions to observations, from a state of ignorance.



Appealing as this may be as a way to achieve reflective consistency, it creates a strange situation in terms of computational complexity:

If actions are type A, and observations are type O, reactions to observations are type $O \rightarrow A$ -- a much larger space to optimize over than A alone. And we're expecting our smaller self to be able to do that!

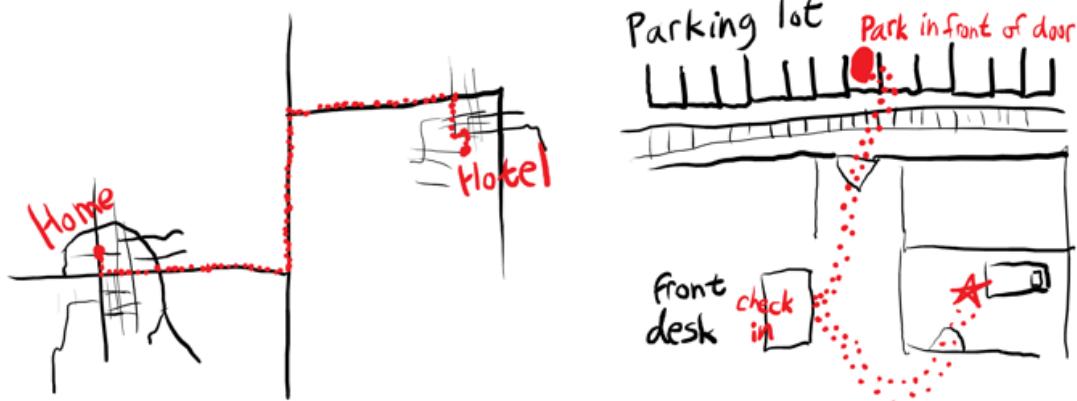
This seems bad.

One way to more crisply state the problem is:

We should be able to trust that our future self is applying its intelligence to the pursuit of our goals without being able to predict precisely what our future self will do.

This criterion is called Vingean reflection.

For example, you might plan your driving route before visiting a new city, but you do not plan your steps. You plan to some level of detail, and trust that your future self can figure out the rest.



Vingeian Reflection is difficult to examine via classical Bayesian decision theory because logical omniscience is assumed, so that the assumption that the agent knows future actions are rational is synonymous with the assumption that the agent knows its future self will act according to one particular optimal policy which the agent can predict in advance.

We have some limited models of Vingean reflection (see Tiling Agents for Self-Modifying AI, and the Löbian Obstacle by Yudkowsky & Herreshoff). A successful approach must walk the narrow line between two problems:

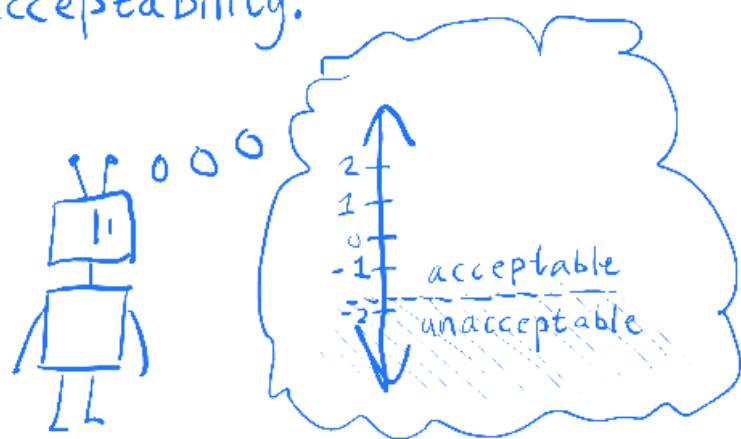
Agents who trust their future self because they trust the output of their own reasoning are inconsistent

Löbian Obstacle

Agents who trust their future selves without reason tend to be consistent but unsound & untrustworthy, and will put off tasks forever because they can do it later

Procrastination Paradox

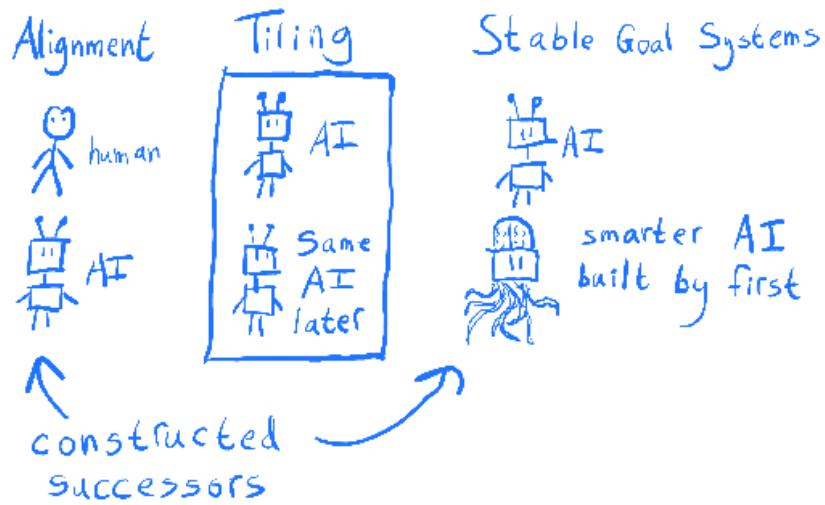
The Vingean reflection results so far apply only to limited sorts of decision procedures, such as satisficers aiming for a threshold of acceptability.



Averting Goodhart's Law

So, there is plenty of room for improvement, getting tiling results for more useful decision procedures and under weaker assumptions.

However, there is more to the robust delegation problem than just tiling and Vingean reflection.



When you construct another agent, rather than delegating to your future self, you more directly face a problem of value loading.

Two facts conspire against us:

→ We don't know what we want.



→ Optimization amplifies slight differences between what we say we want and what we really want.

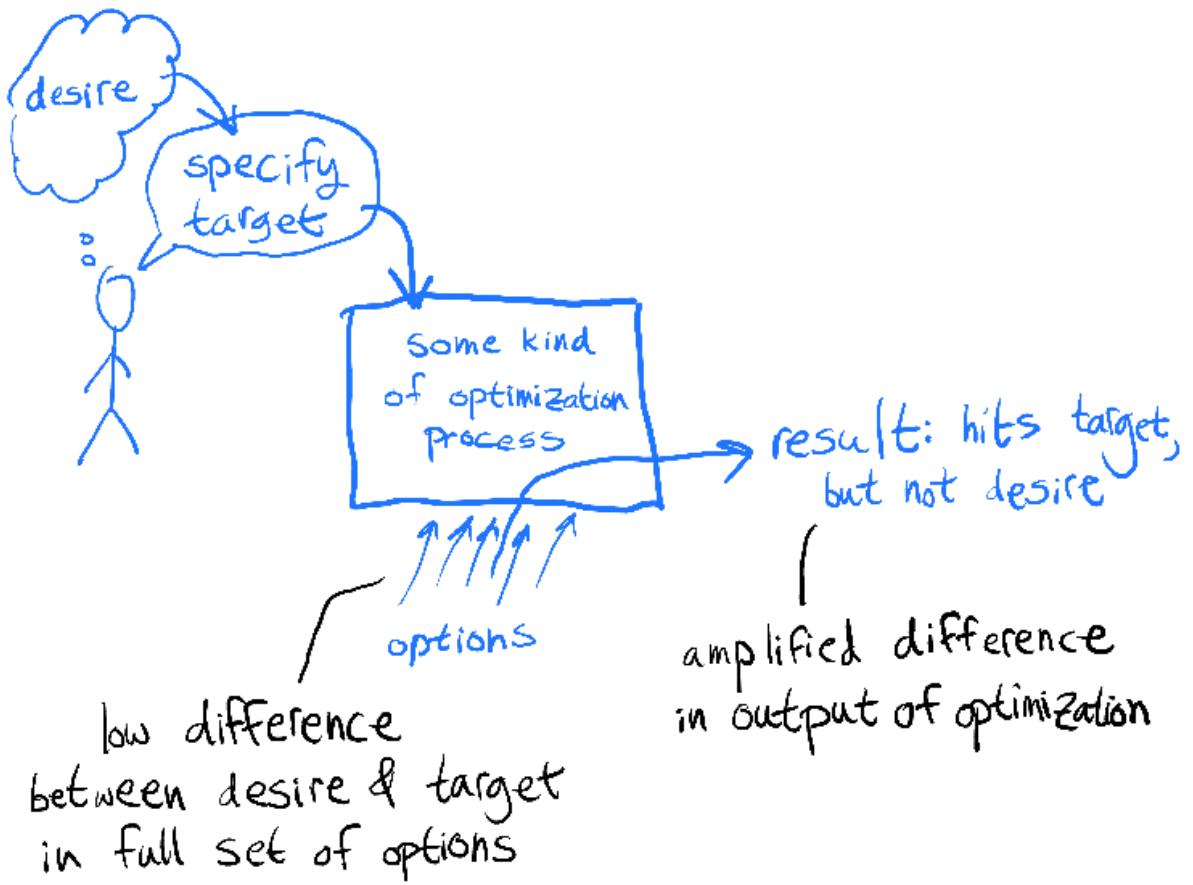


The misspecification-amplifying effect
is known as Goodhart's Law:

"Any observed statistical
regularity will tend to collapse
once pressure is placed upon
it for control purposes."

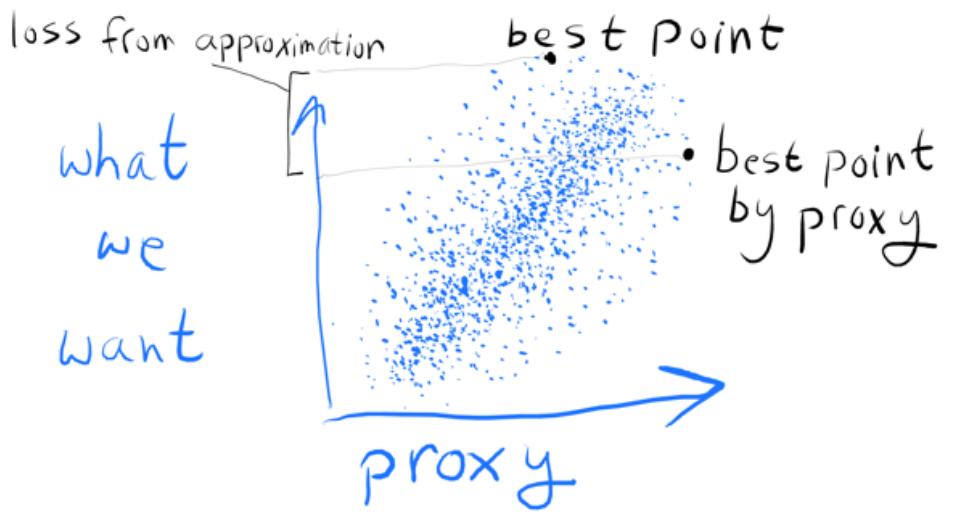
— Charles Goodhart

IE: when we specify a target for optimization, it is reasonable to expect it to be correlated with what we want (perhaps highly correlated, even); but, unfortunately, this does not mean that optimizing it will get us closer to what we want -- especially at high levels of optimization.



There ^(at least) are four types of Goodhart:

- regressional
- extremal
- causal
- adversarial



regressional

Regressional Goodhart can occur when there is a less than perfect correlation between the proxy and the goal.

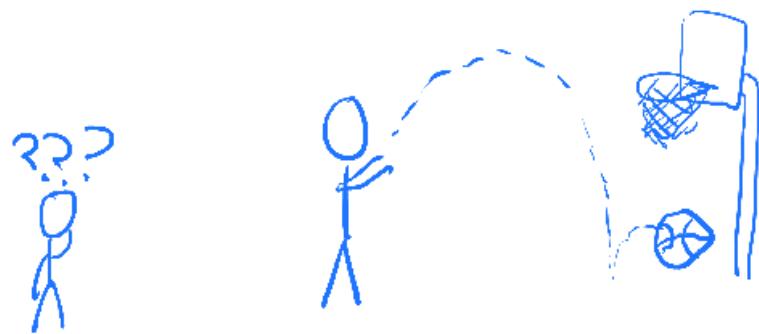
It is more commonly known as the optimizer's curse, and it is related to regression to the mean.

For example, you might draft players for a basketball team based on height alone.



This isn't a perfect heuristic, but there is a correlation between height and basketball ability, which you can utilize to make your selection.

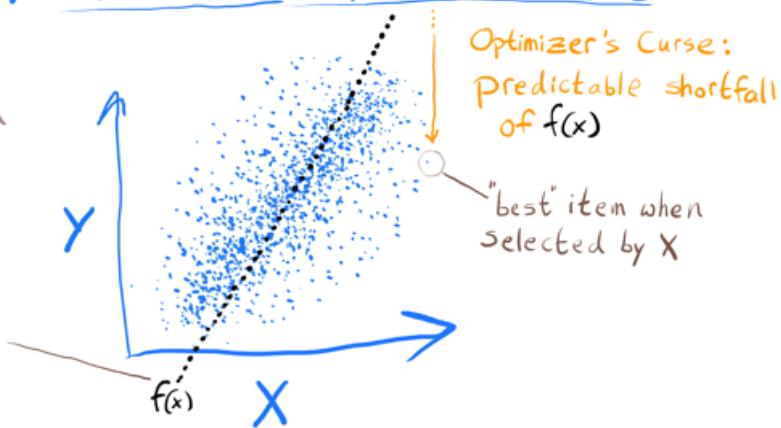
It turns out that, in a certain sense,
you will be predictably disappointed
if you expect the general trend to hold
up as strongly for your selected team.



Stated in statistical terms: an unbiased estimate of Y given X is not an unbiased estimate of Y when we select for the best X .

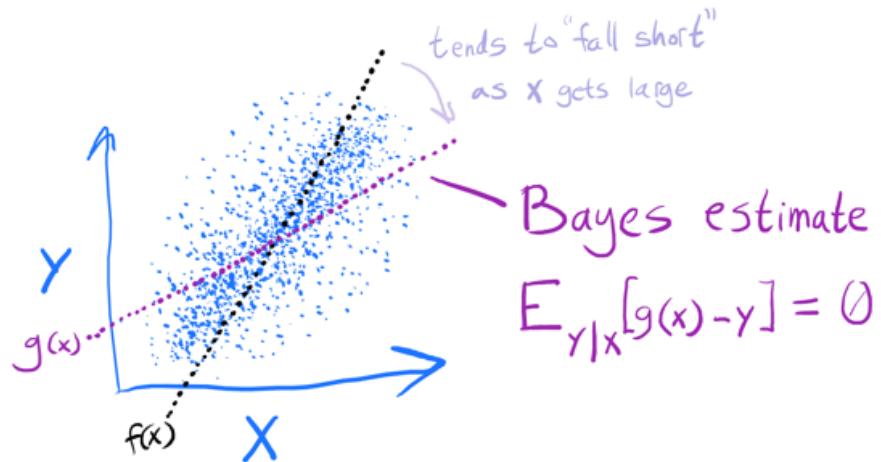
$f(x)$ is an unbiased estimate of y as a function of x , ie, $f(x)$ satisfies

$$E_{x|y} [f(x) - y] = 0$$

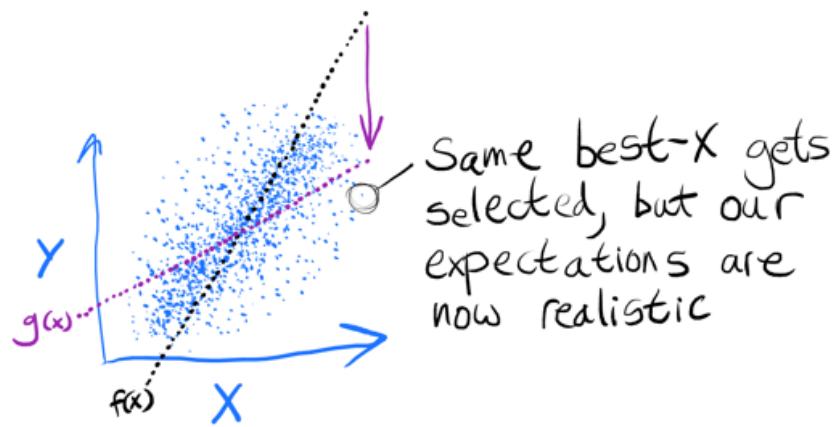


In that sense, we can expect to be disappointed when we use X as a proxy of Y for optimization purposes.

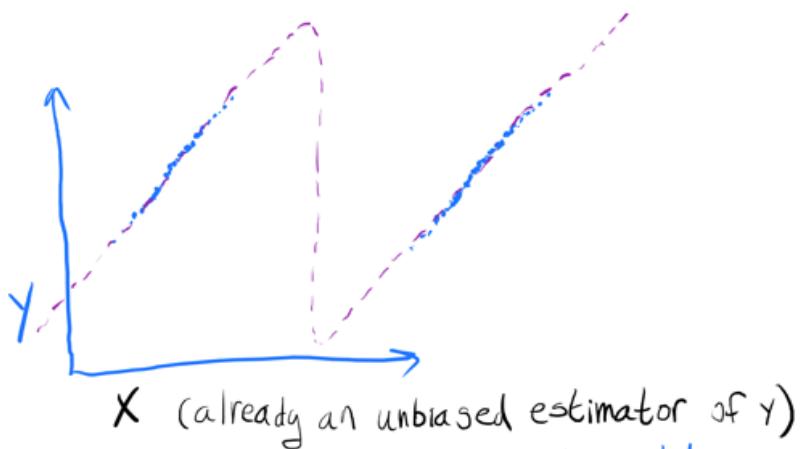
Using a Bayes estimate instead of an unbiased estimate, we can eliminate this sort of predictable disappointment.



The Bayes estimate accounts for the noise in X , bending toward typical Y -values.



This doesn't necessarily allow us to get a better Y value, since we still only have the information content of X to work with. However, it sometimes may.

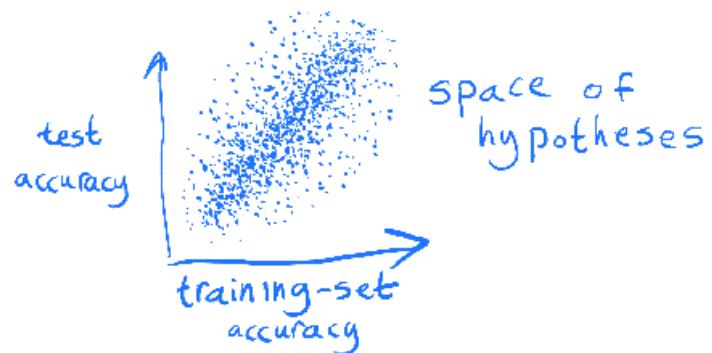


If Y is normally distributed with variance 1, and X is $Y \pm 10$ with even odds of + or -, a Bayes estimate will give better optimization results by almost entirely removing the noise.

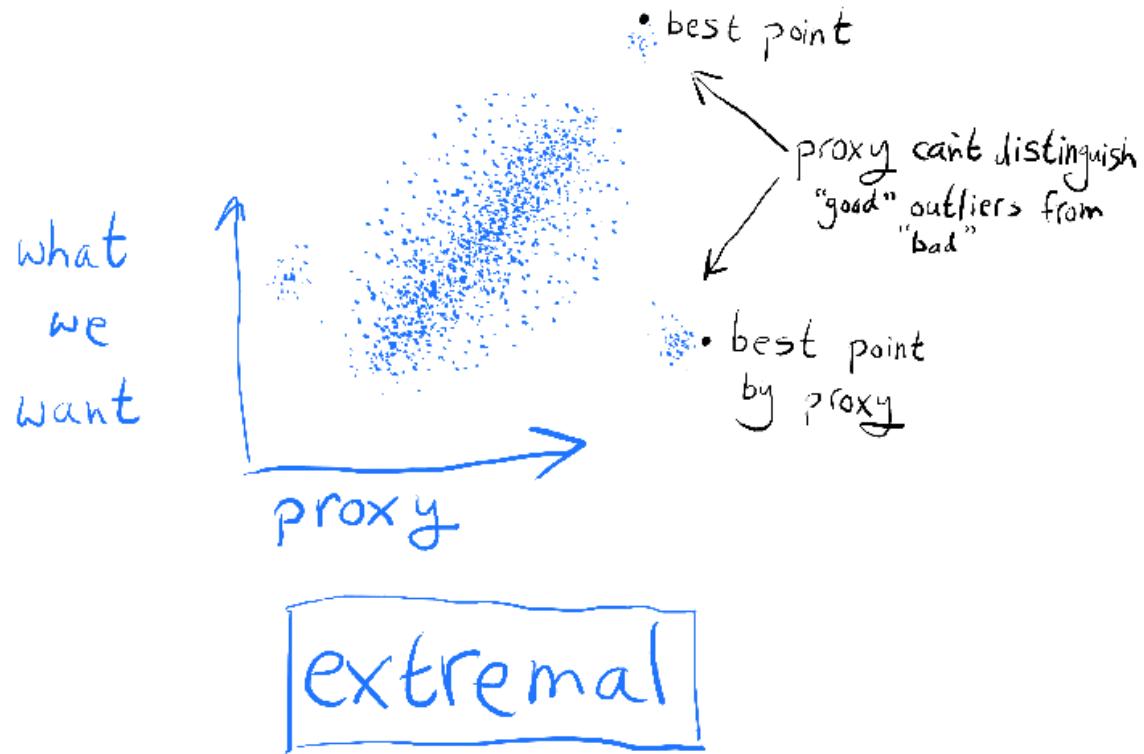
Regressional Goodhart seems like the easiest form of Goodhart to beat: just use Bayes! However, there are two big problems with this solution.

- Bayesian estimators are very often intractable in cases of interest.
- It only makes sense to trust the Bayes estimate under a **realizability** assumption.

A case where both of those problems become critical is computational learning theory.



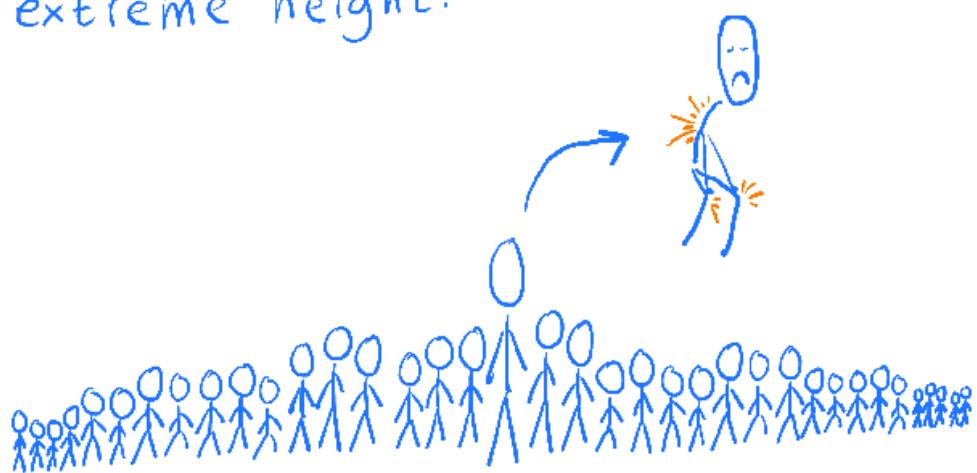
It often isn't computationally feasible to calculate the Bayesian expected generalization error of a hypothesis. But, even if you could, you still would wonder whether your chosen prior reflected the world well enough.



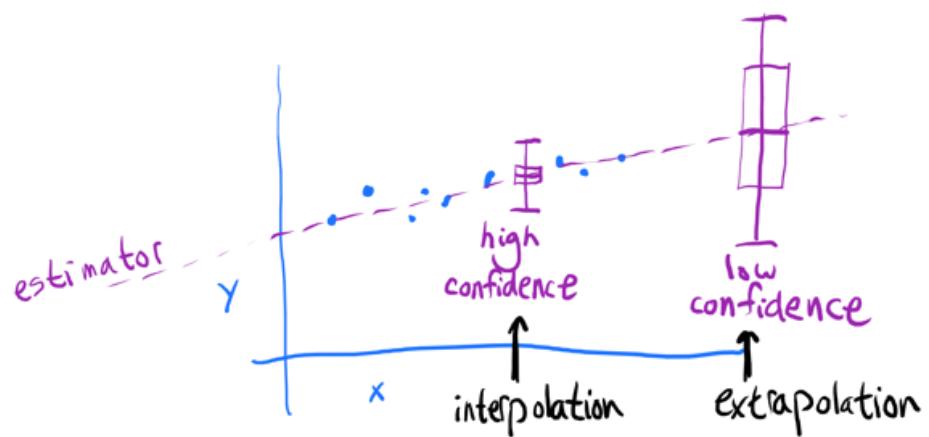
In extremal Goodhart, optimization pushes you outside the range where the correlation exists, into portions of the distribution which behave very differently.



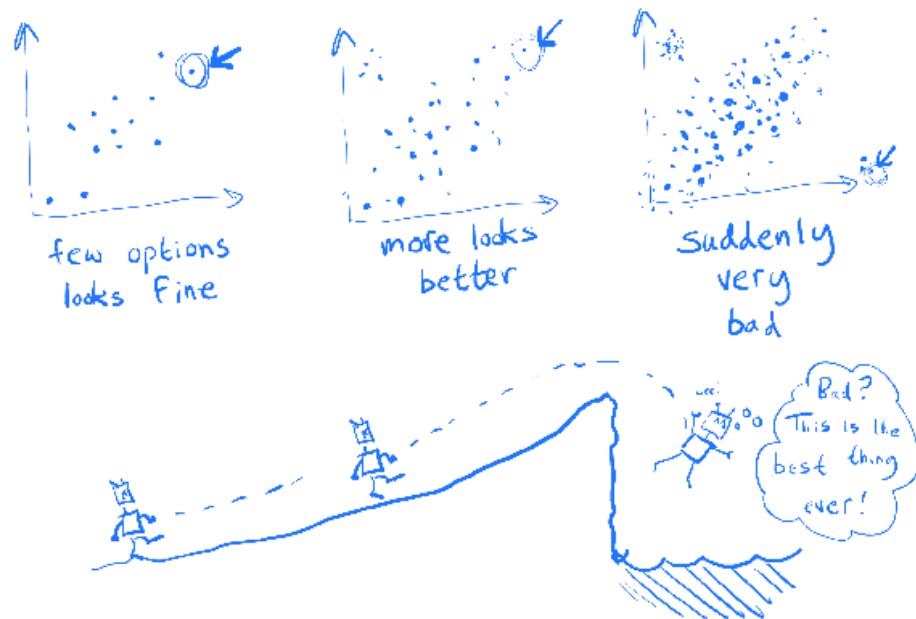
For example, if you select the tallest people in the entire world to play basketball, they may do quite poorly due to health problems which come with extreme height.



The difference between this and regressive Goodhart is related to the classical interpolation/extrapolation distinction.



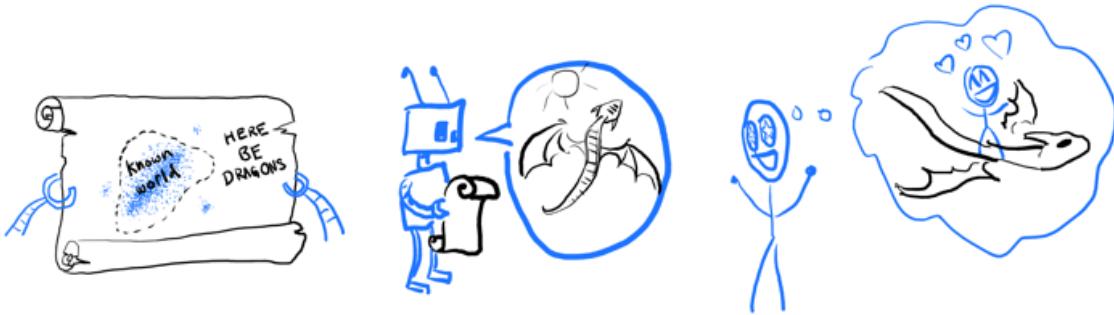
Unlike regressive Goodhart, this involves a sharp change in behavior as the system is scaled up, making it more difficult to anticipate.



As in the regressional case, a Bayesian solution addresses this concern in principle, if you trust a probability distribution to reflect the possible risks sufficiently well.

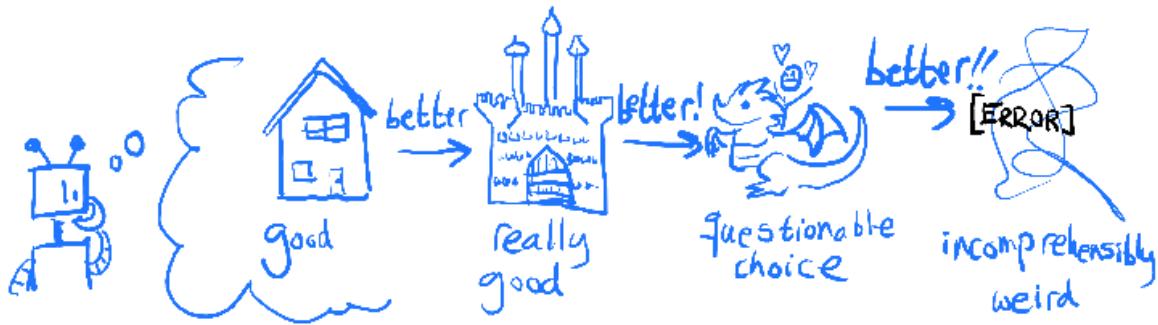
However, the realizability concern seems even more prominent here. Can a prior be trusted to anticipate problems with proposals highly optimized to look good to that specific prior?

Certainly a human's judgement couldn't be trusted under such conditions...



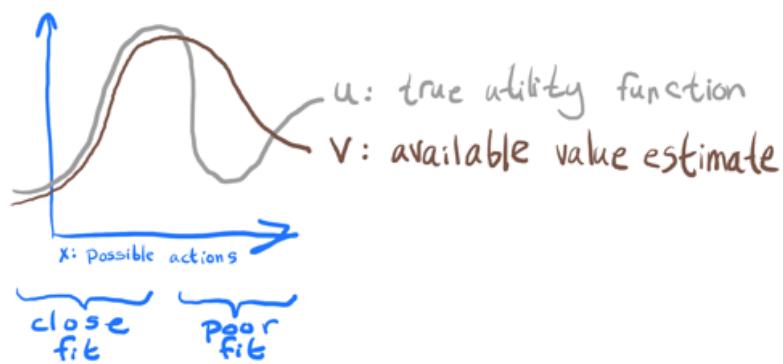
... an observation which suggests that this problem remains even if a system's judgements about value perfectly reflect a human's.

We might say the problem is that "typical" outputs avoid extremal Goodhart but "optimizing too hard" takes you out of the realm of the typical — but how can we formalize "optimizing too hard" in decision-theoretic terms?



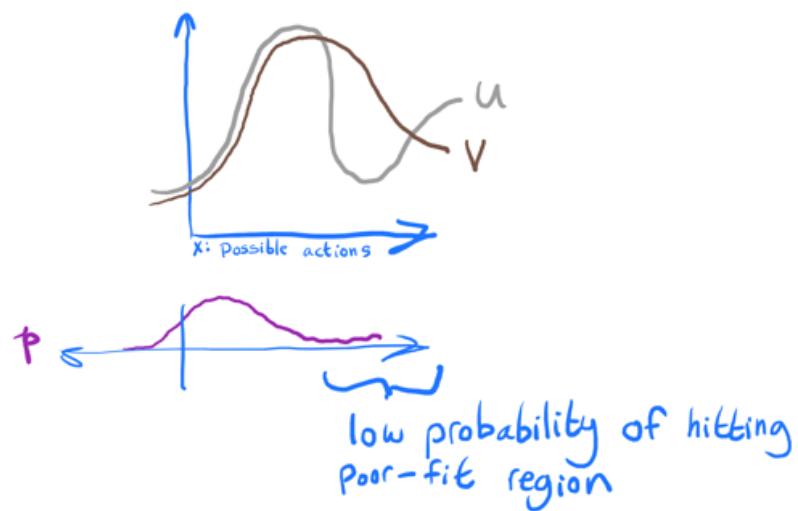
Quantilization offers a formalization.

Imagine a proxy $v(x)$ as a "corrupted" version of a true function $u(x)$.

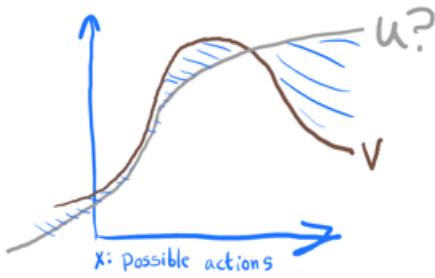


There might be different regions where the corruption is better or worse.

Now suppose that we can specify a "trusted" probability distribution $P(x)$, for which we are confident that the average error is below some threshold c .



By stipulating P and c , we give information about where to find low-error points without any estimates of u or of the actual error at any one point.



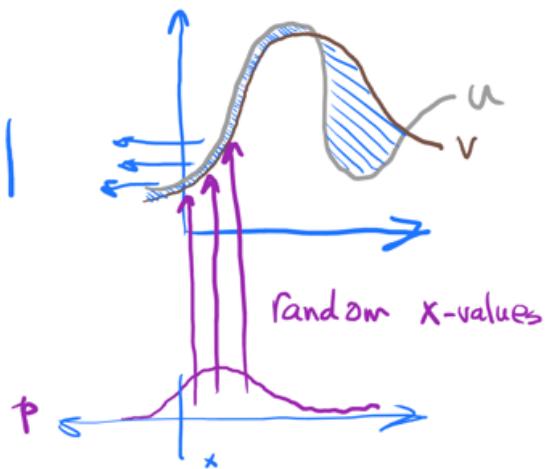
growing error concentrated toward edges of P ?



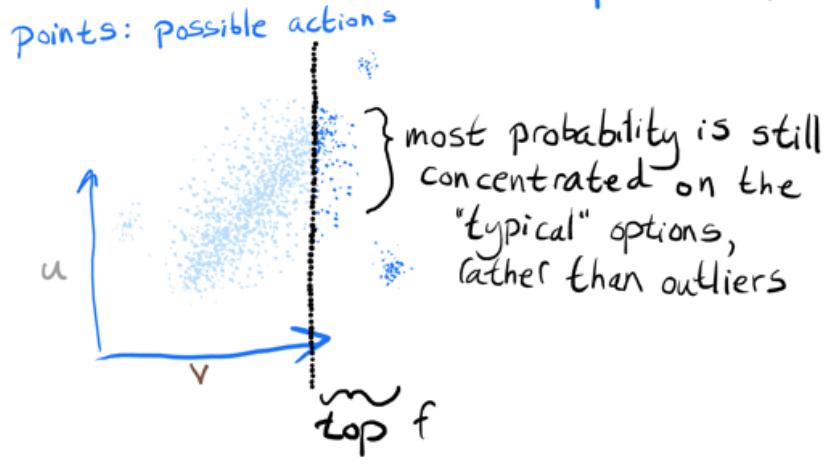
a small but severe spike of error right in the middle of P ?

When we select actions randomly from P , we can be sure that there is a low probability of high error regardless.

$$c \geq \mathbb{E}_{x \sim P} |v(x) - u(x)|$$

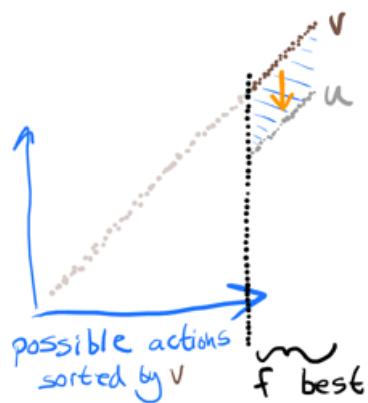


So, how do we use this to optimize?



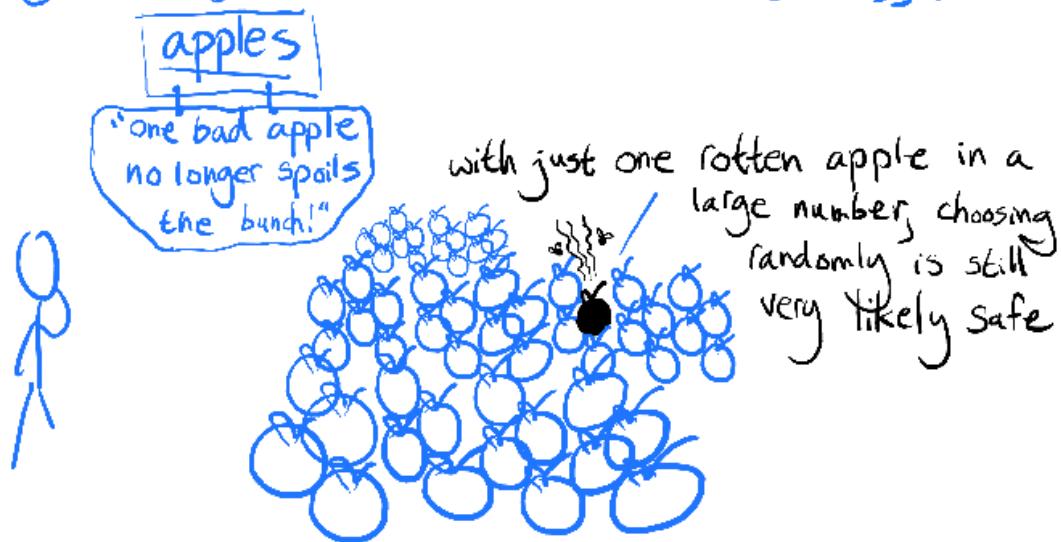
A quantilizer selects from P , but discarding all but the top fraction f ; for example, the top 1%.

This guarantees at most $\frac{c}{f}$ expected overestimation of value, since in the worst case, all of the error was over-estimation of the f_{best} .



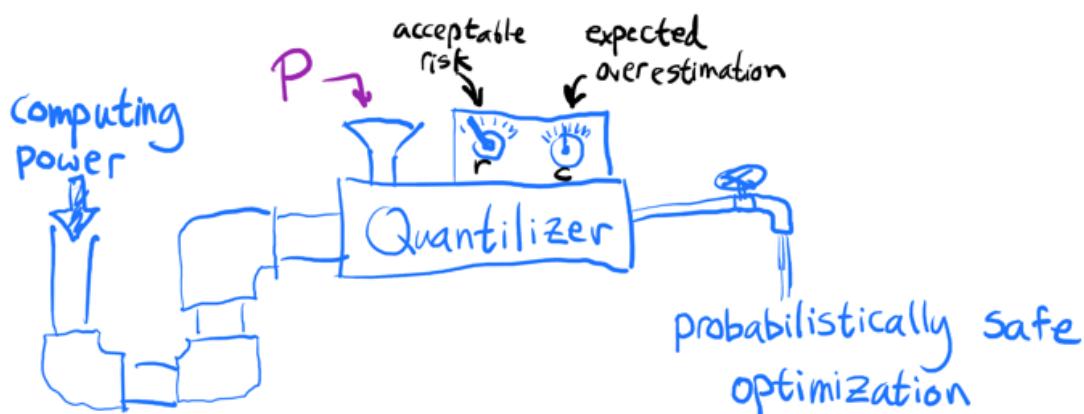
We can therefore choose an acceptable risk, $r = \frac{c}{f}$, and set the parameter f as c/r .

Quantilization is in some ways very appealing, since it allows us to specify safe classes of actions without trusting every, or any, individual action in the class.

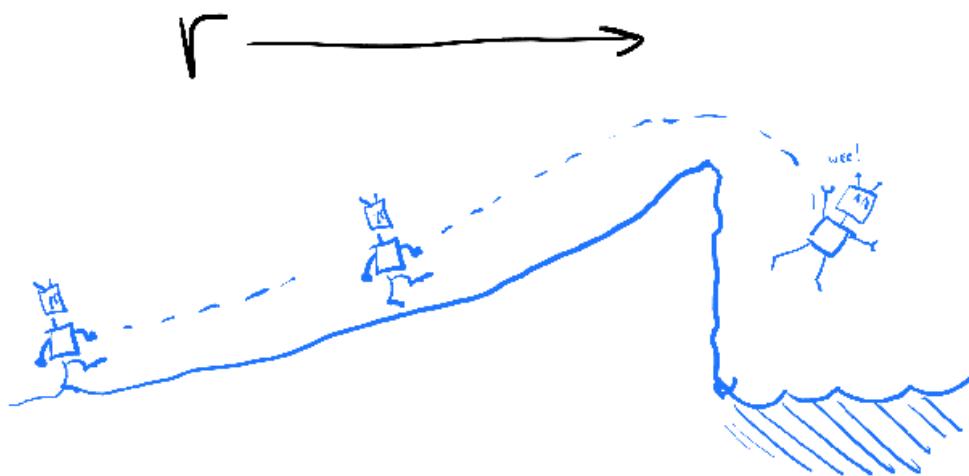


However, it also leaves much to be desired.

Where do "trusted" distributions come from?
How do you estimate expected error c ,
or select acceptable risk r ?



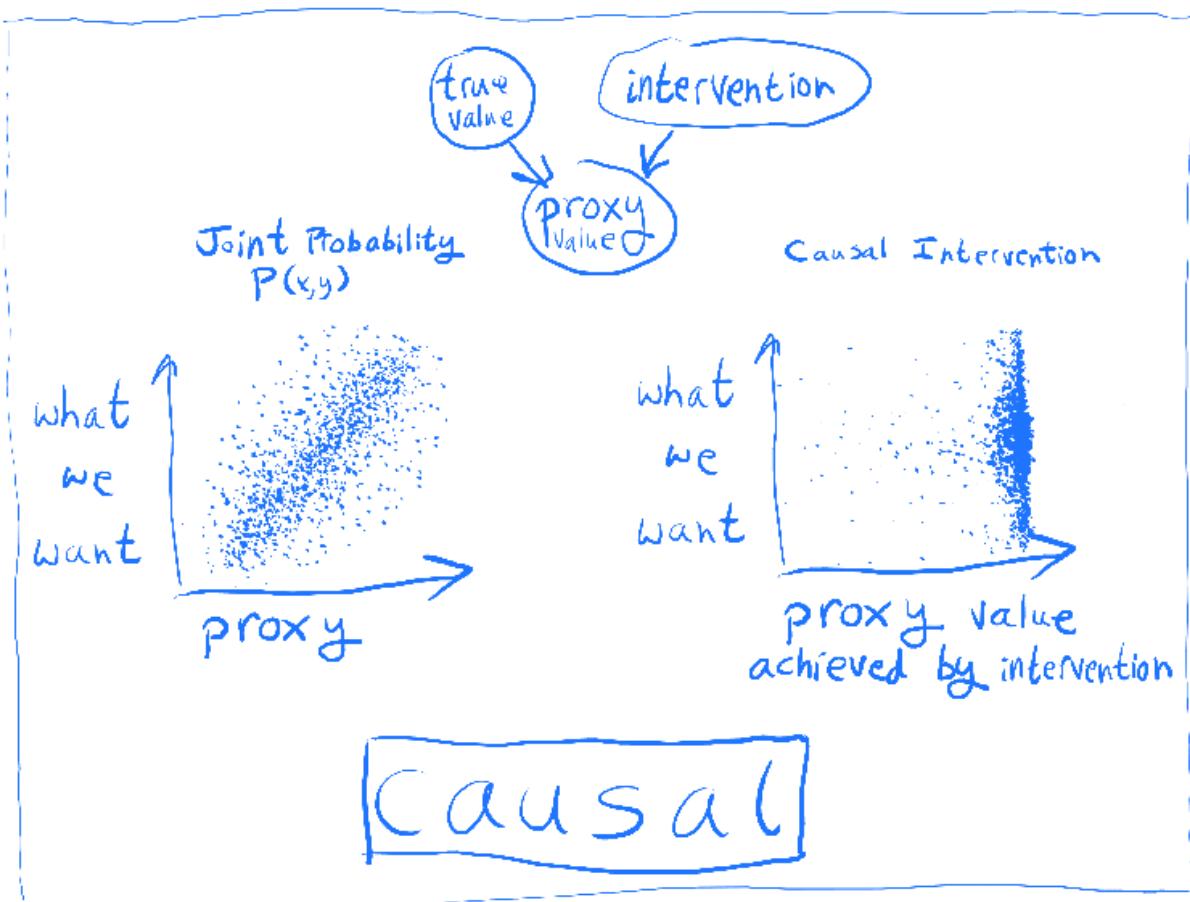
This is especially bad because it gives you a knob to turn which will apparently improve performance while increasing risk, until (possibly sudden) failure.



Also, quantilization doesn't seem likely to tile:
a quantilizing agent has no special reason
to preserve the quantilization algorithm
when making self-improvements or new agents.



So, all told, it seems like there is
room for improvement in the handling
of extremal Goodhart.



Another way optimization can go wrong is by using a proxy which more directly falls apart when we use it to optimize: the act of selecting for it breaks the connection to what we care about.

For example, you might play basketball in order to become tall.

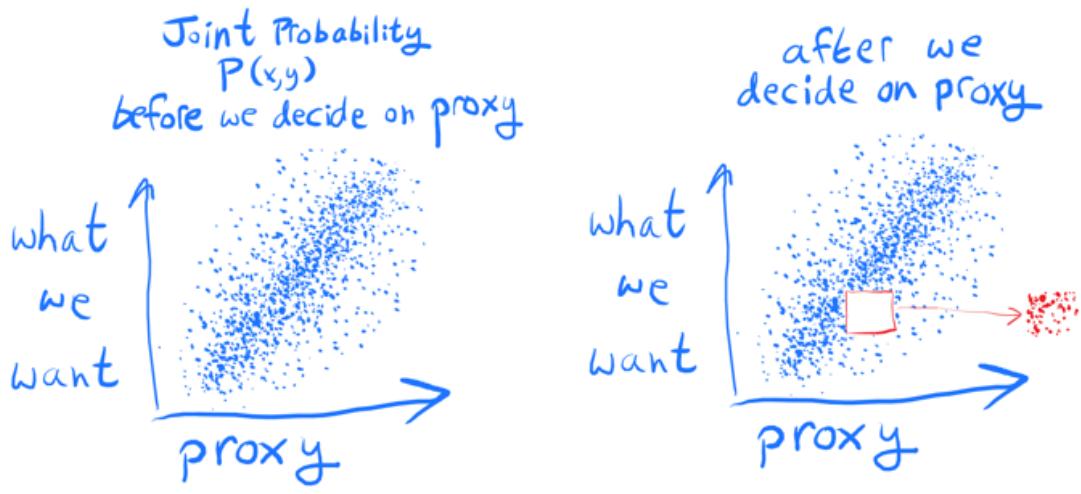


The only way to avoid this sort of mistake is to get counterfactuals right.

This might seem like punting to decision theory, but the connection here enriches both: counterfactuals have to address questions of trust due to tiling concerns, and, trust has to address counterfactual concerns due to causal Goodhart.

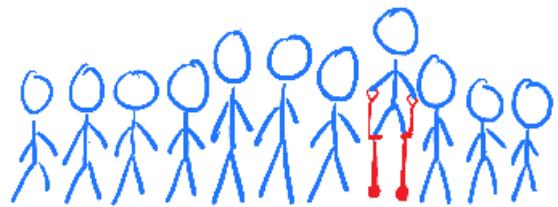
Yet again, one of the big challenges here is realizability.

As mentioned in embedded world-models, even if you have the right theory of how counterfactuals work generally, Bayesian learning doesn't provide much of a guarantee of learning to select actions well without assuming realizability.



adversarial

Finally, there is adversarial Goodhart, in which agents actively make our proxy worse by intelligently manipulating it.



For example, if applicants to your basketball team know how you are choosing players, some will specifically practice on what you check, neglecting other aspects of the game.

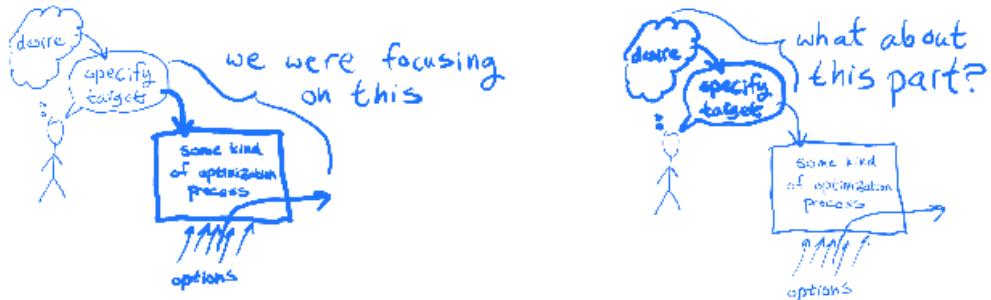
This category is the most common interpretation of Goodhart's remark. At first, it may seem less relevant to our concerns here. However, when searching in a large space which is sufficiently rich, there are bound to be some elements of that space which implement adversarial strategies.

But, that is a subject for the subsystem alignment section.

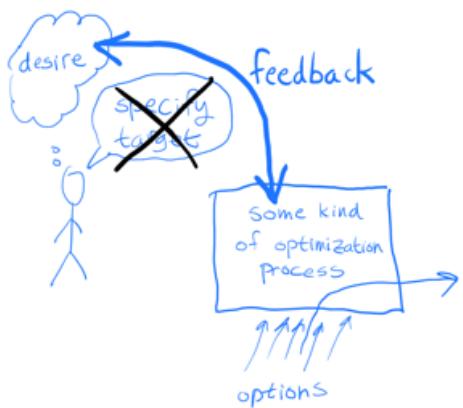
Value Loading & Value Learning

Remember none of this would happen if a system were optimizing what we wanted directly, rather than optimizing a proxy.

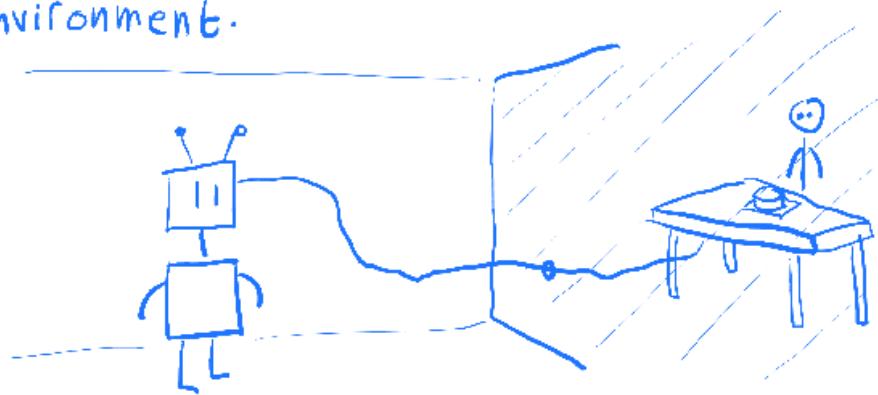
Besides anti-Goodhart measures, it would obviously help to be able to specify what we want precisely.



Unfortunately, this is hard. So, can a successor agent help us with this? If it is very intelligent, maybe it can learn what we want?



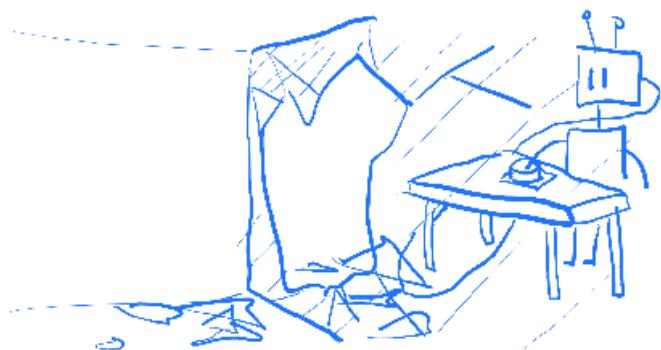
AIXI learns what to do through a reward signal which it gets from the environment.



We can imagine humans have a button which they press when AIXI does something they like.

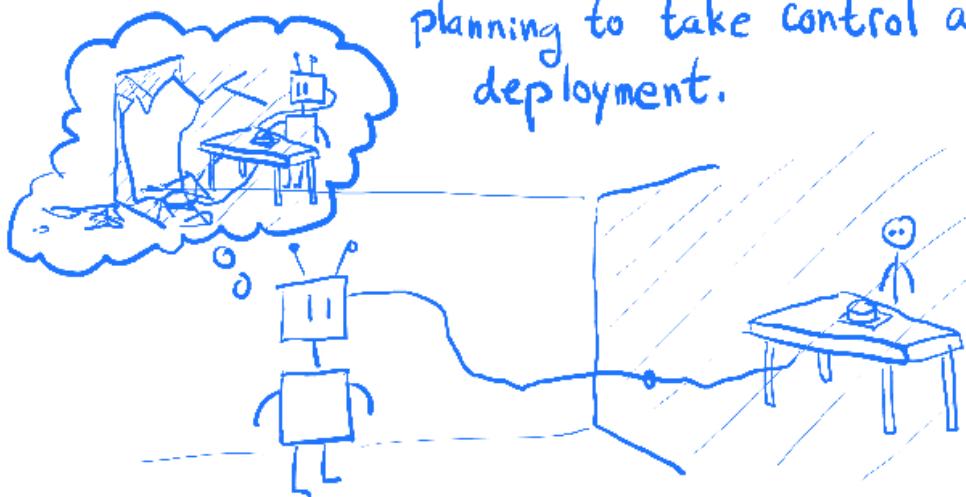
The problem with this is that AIXI will apply its intelligence to the problem of taking control of the reward button.

This is the problem of wireheading.



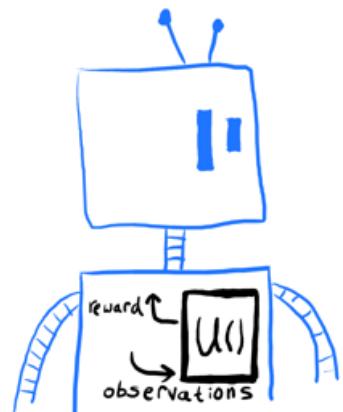
This kind of behavior is potentially very difficult to anticipate; the system may deceptively behave as intended during training,

planning to take control after deployment.

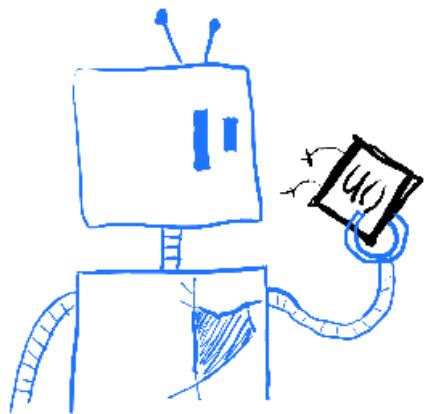


This is called a “treacherous turn”.

Maybe we build the reward function into the agent, as a black box which issues rewards based on what is going on. The box could be an intelligent subagent in its own right, which figures out what rewards humans would want to give.



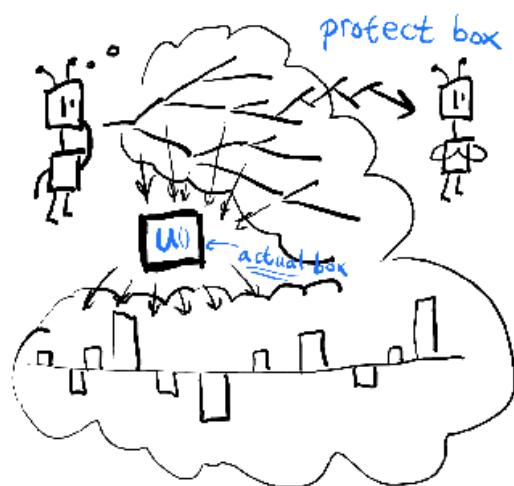
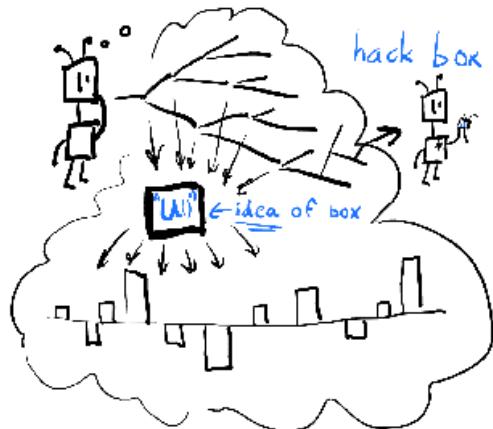
The box could even defend itself by issuing punishments for actions toward modifying the box. In the end, though, if the agent understands the situation, it will be motivated to take control anyway.



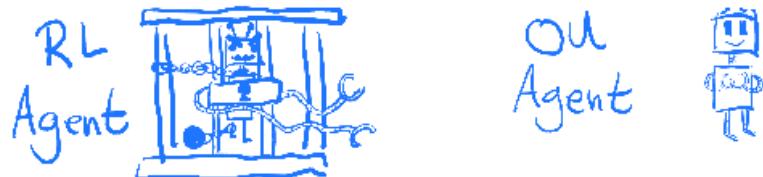
If the agent is told to get high output from "the button" or "the box", then it will be motivated to hack those things.



However, if you run the expected outcomes of plans through the actual box, then plans to hack the box are evaluated by the box itself, which won't find the idea appealing.



Daniel Dewey calls the second sort of agent an observation-utility maximizer (OU). (Others have included OU agents within a more general notion of RL.)

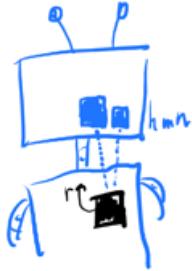


I find it very interesting how you can try all sorts of things to stop an RL agent from wireheading, but the agent keeps working against it. Then, you make the shift to OU agents and the problem vanishes.

However, we still have the problem of specifying $U(\cdot)$.

Daniel Dewey pointed out that OU agents can still use learning to approximate $U(\cdot)$; we just can't treat it as a black box.

RL:
agent tries
to learn to
predict the
reward
function

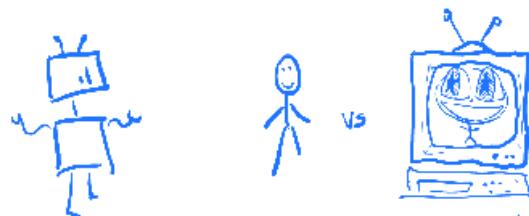


OU:
agent uses
estimated utility
functions from
a human-specified
value-learning prior



However, it's still difficult to specify a learning process which doesn't lead to other problems.

For example, if you're trying to learn what humans want, how do you robustly identify humans in the world?



Merely statistically decent object recognition could lead back to wireheading.

Even if you successfully solve that problem, the agent can be correctly locating value in the human, but still motivated to change human values to be easier to satisfy.



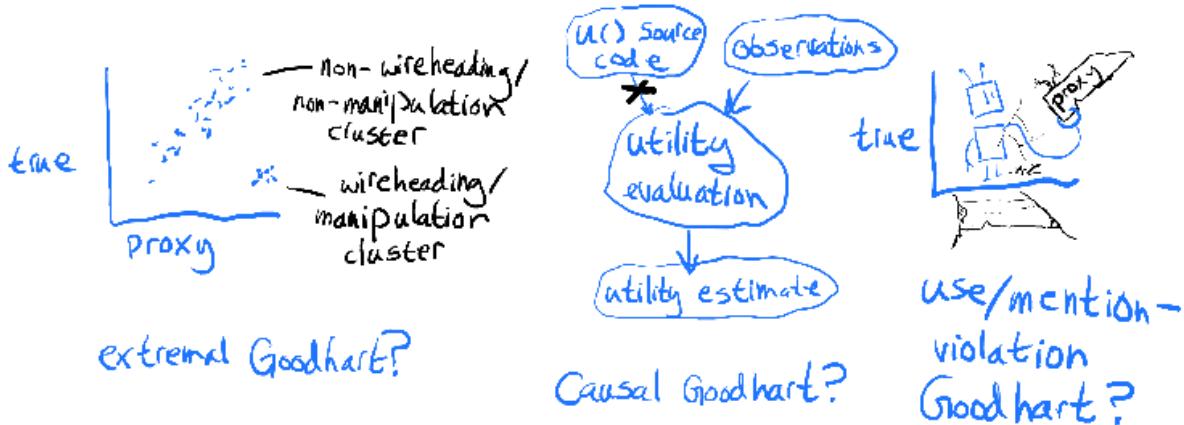
For example, if there is a drug which modifies human preferences to only care about using the drug, an AI agent could be motivated to give humans that drug in order to make its job easier. This is called the human manipulation problem.



It's like giving candy to a baby!

Anything marked as the true repository of value gets hacked.

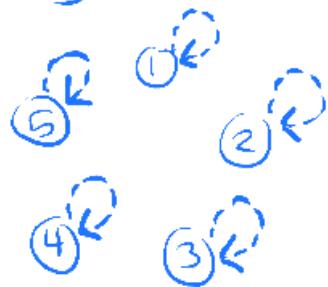
Whether this is one of the four types of Goodharting, or a fifth, or something all its own, it seems like a theme.



So, the challenge is to create stable
pointers to what we value: an indirect reference
to values not directly available to be optimized,
which doesn't thereby encourage hacking the
repository of value.

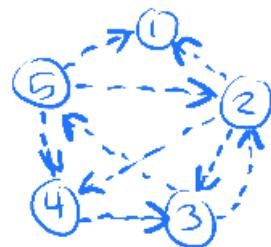
One important point is from Reinforcement
Learning with a Corrupted Reward Channel (Tom Everitt et al):
the way you set up the feedback loop makes
a huge difference.

They draw the following picture:



Standard RL:

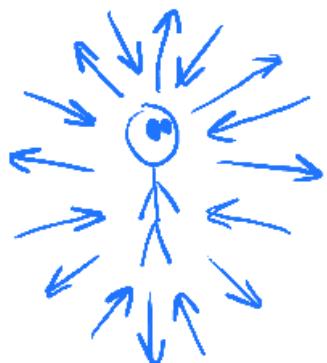
the feedback about the value of a state comes from the state itself, so corrupt states can be "self-aggrandizing"



Decoupled RL:

the feedback about the quality of a state comes from some other state, making it possible to learn correct values even when some feedback is corrupt.

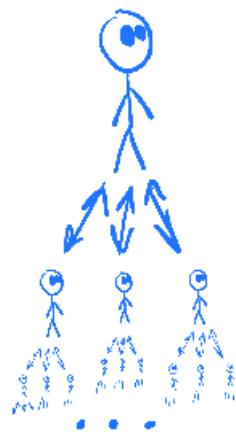
In some sense, the challenge is to put the original, small agent in the feedback loop in the right way.



However, the problems with updateless reasoning mentioned earlier make this hard; the original agent doesn't know enough.

One way to try to address this
is through intelligence amplification:
try to turn the original agent into a
more capable one with the same values,
rather than creating a successor agent
from scratch and trying to get value
loading right.

For example, Paul Christiano proposes an approach in which the small agent is simulated many times in a large tree, which can perform complex computations by splitting problems into parts. However, this is still fairly demanding for the small agent: it doesn't just need to know how to break problems



down into more tractable pieces; it also needs to know how to do so without giving rise to malign subcomputations.

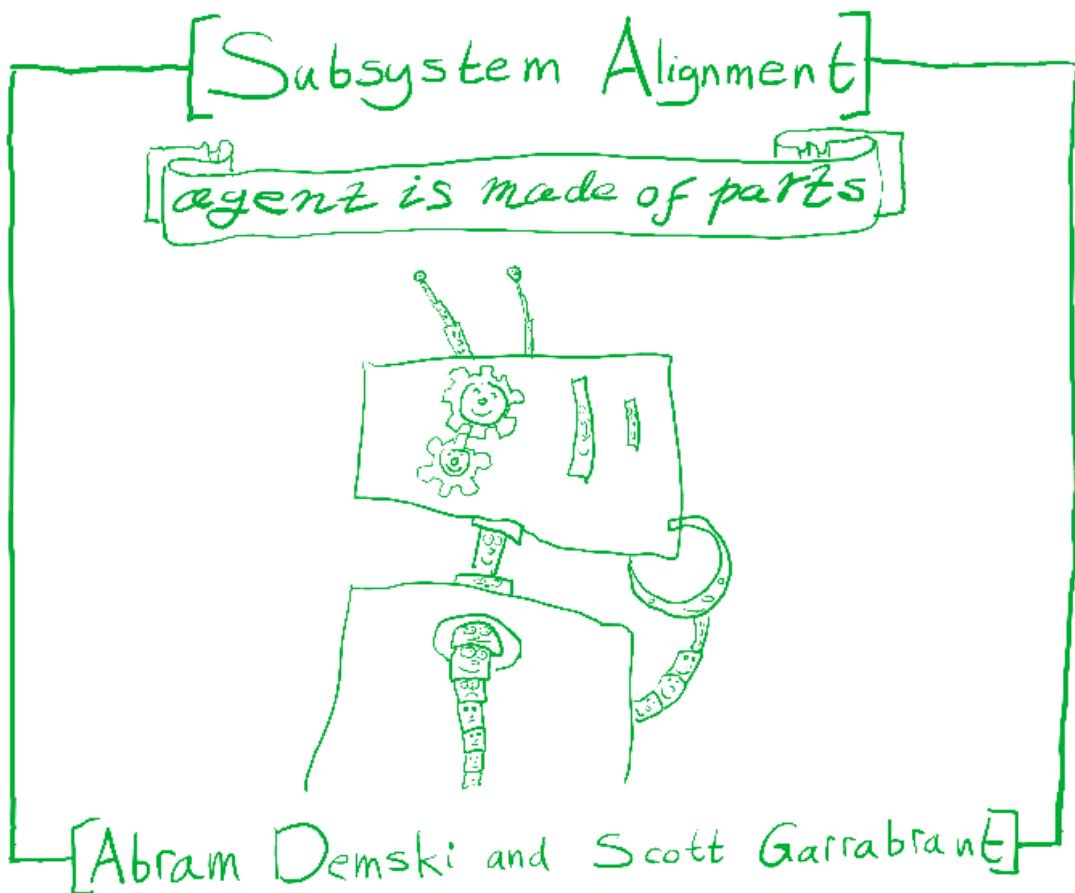
For example, since it can use the copies of itself to get a lot of computational power, it could easily brute-force search for solutions and Goodhart itself on a poor proxy for what it values.

The issue of aligned top-level computations giving rise to malign subcomputations is the subject of the next section, subsystem alignment.

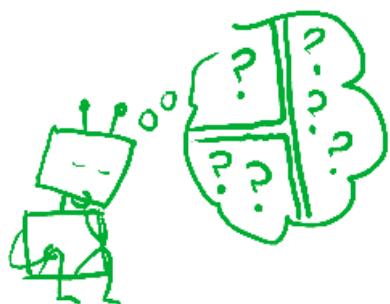
Subsystem Alignment

Crossposted from the [AI Alignment Forum](#). May contain more technical jargon than usual.

(*The bibliography for the whole sequence can be found [here](#)*)



You want to figure something out, but you don't know how to do that yet.



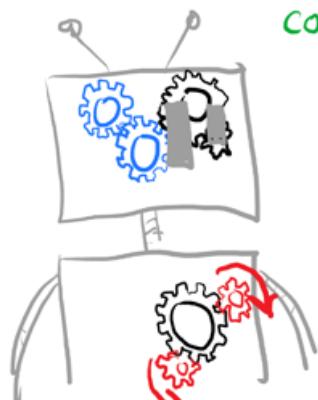
You have to somehow break up the task into sub-computations.

There is no atomic act of "thinking";

intelligence must be built up at non-intelligent parts.

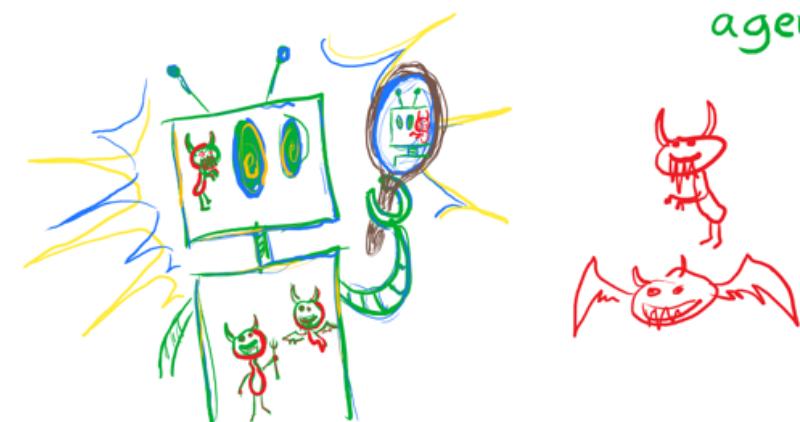
The agent being made of parts is part of what made **counterfactuals** hard, since the agent may have to reason about impossible

configurations of those parts.



Being made of parts is what makes self-reasoning & self-modification even possible.

What we're going to discuss in this section, though, is another problem: when the agent is made of parts, there could be **adversaries** not just in the external environment, but inside the agent as well.

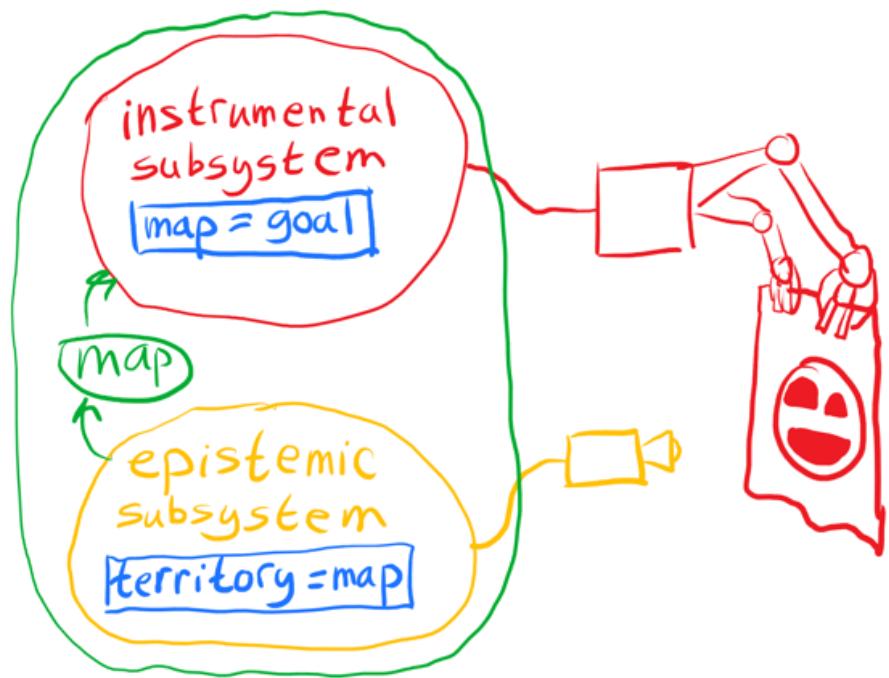


This cluster of sub-problems is

Subsystem Alignment: Ensuring
that subsystems are not working at
cross purposes; avoiding subprocesses
optimizing for unintended goals.

- Benign Induction
- Benign Optimization
- Transparency
- Mesa-Optimizers

Here's a straw agent design:



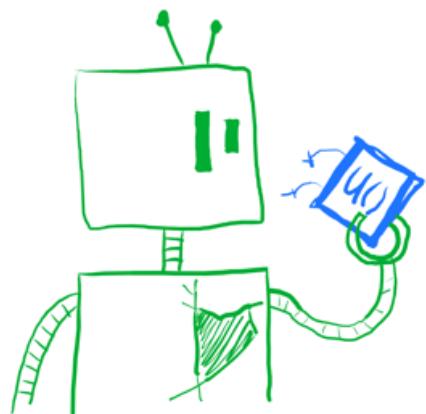
The epistemic subsystem just wants accurate beliefs.

The instrumental subsystem uses those beliefs to track how well it is doing.

If the instrumental subsystem gets too capable relative to the other, it may start fooling it (as depicted).

If the epistemic subsystem gets too strong, things could possibly go badly, too.

This agent design is not particularly realistic, because the subsystems need not be agents with goals of their own. However, we did see in the section on wireheading that the problem is hard to avoid.



One reason to avoid booting up sub-agents who want different things is:

Robustness to Relative Scale

An approach is robust to scale if it still works, or fails gracefully, as you scale capabilities. There are three types:

- robustness to scaling up
- robustness to scaling down
- robustness to relative scale

Robustness to scaling up means that your system does not depend on not getting too powerful. One way to check this is to think about what would happen if the function the agent optimizes were actually maximized.

Think Goodhart's Law.

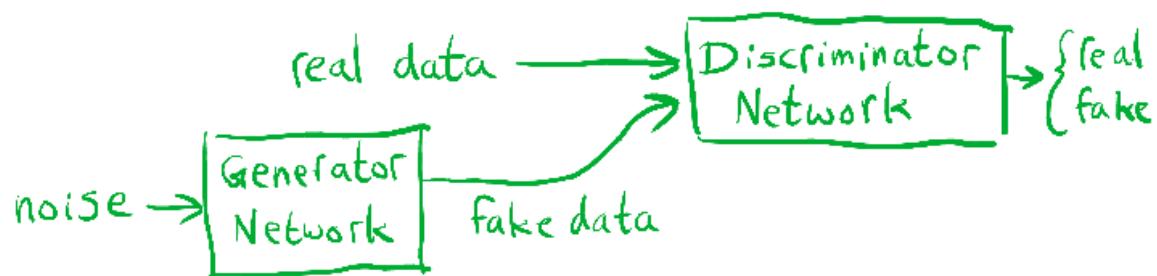
Robustness to scaling down means your system still works if made less powerful.

Of course it may stop being useful, but it should fail gracefully.

Your system might work if it can exactly maximize some function, but is it safe if you approximate?

For example, maybe a system is safe if it can learn human values very precisely, but approximation makes it increasingly misaligned.

Robustness to relative scale means that your design does not rely on subsystems being similarly powerful. For example, GAN (Generative Adversarial Network) training can fail if one sub-network gets too strong, because there's no longer any training signal.



Lack of robustness to scale isn't necessarily something which kills a proposal, but it is something to be aware of; lacking robustness to scale, you need strong reason to think you're at the right scale.

Robustness to relative scale is particularly important for subsystem alignment. An agent with intelligent sub-parts should not rely on being able to outsmart them, unless we have a strong account of why this is always possible.

Moral: have a unified agent not working at cross purposes with itself.

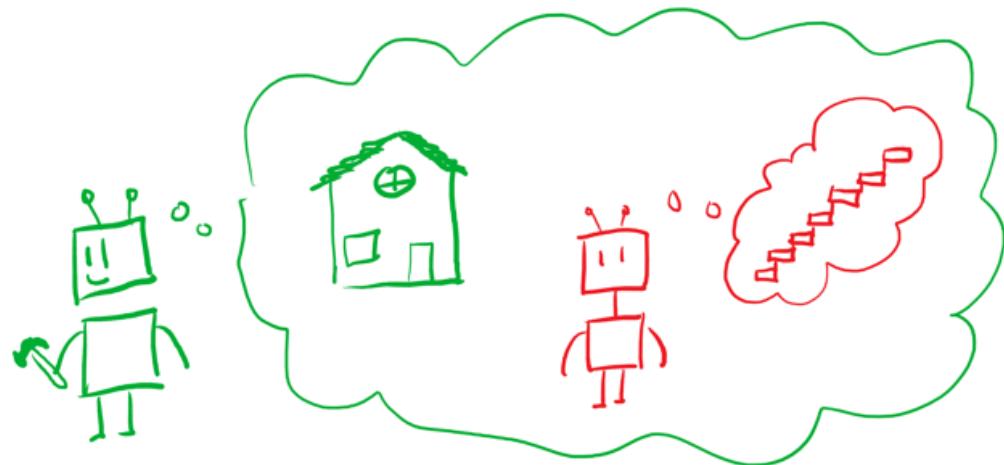
Why would anyone make an agent with parts fighting against one another?

- subgoals
- pointers
- Search

Subgoals: Splitting up a task into subgoals may be the only way to efficiently find a solution.

However, a subgoal computation should not forget the big picture!

An agent designed to build houses
should not boot up a sub-agent who
cares only about building stairs.

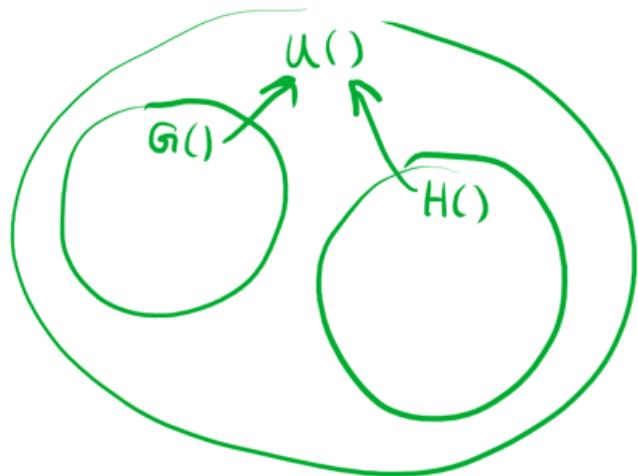


The obvious way to avoid agents that pursue subgoals at the cost of the overall goal is to have subsystems not be agentic.



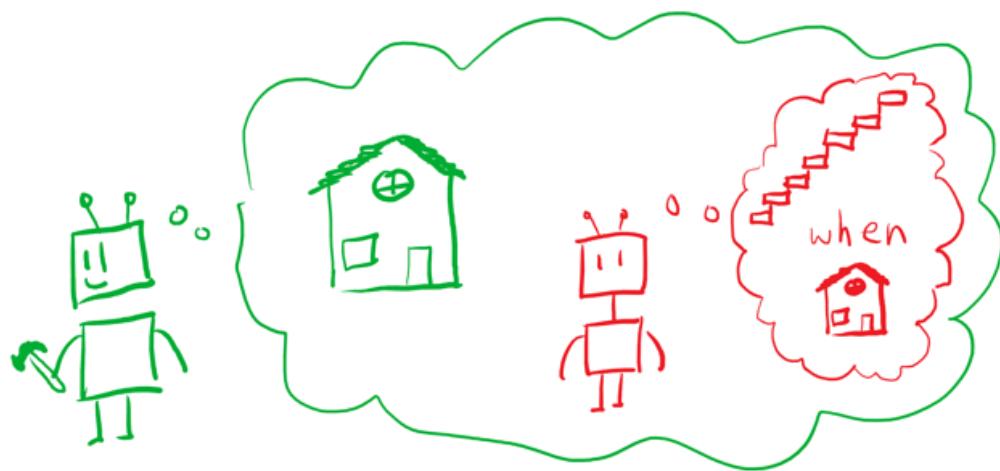
The problem is that there are convergent instrumental incentives to be agentic — we don't know how to systematically build highly capable nonagents.





One intuition is that although subsystems need to have their own goals in order to decompose problems into parts, the sub-goals need to "point back" robustly to the main goal.

A house-building agent might spin up a subsystem that cares only about stairs, but only cares about stairs in the context of houses.



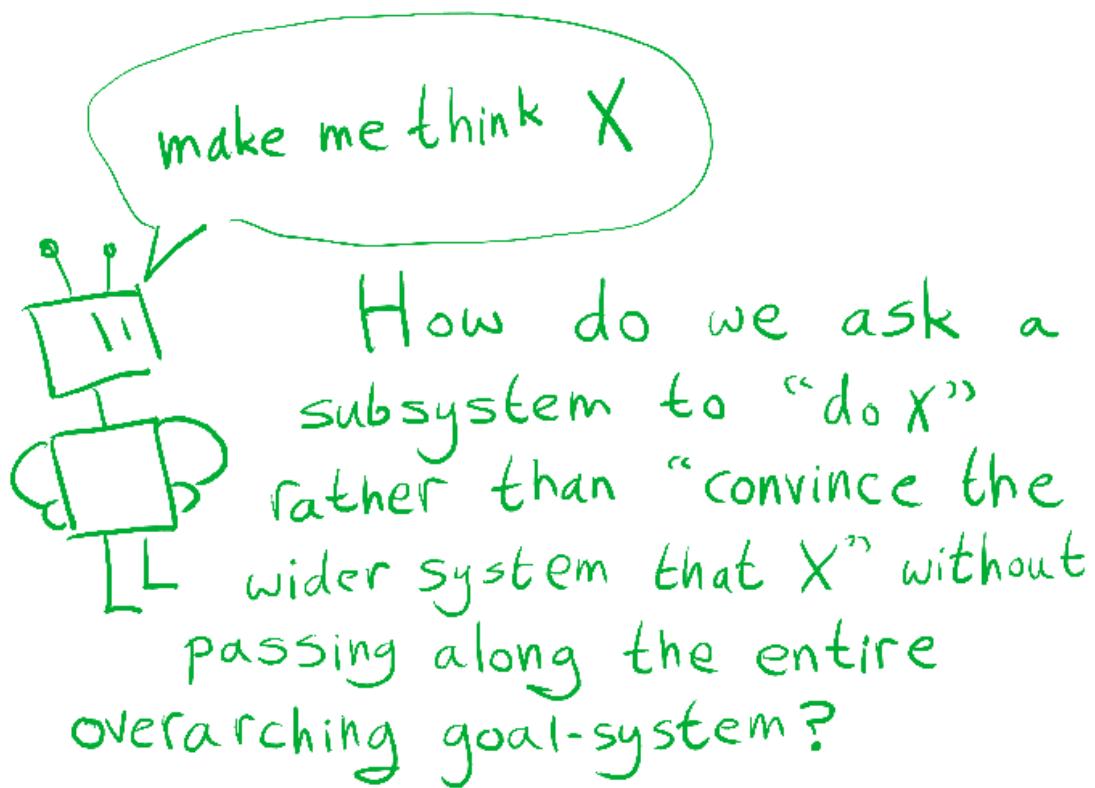
However, you need to do this in a way which doesn't just amount to wanting houses.

This brings me to the next item:

Pointers: It may be difficult for subsystems to carry the whole-system goal around with them, since they need to be reducing the problem.

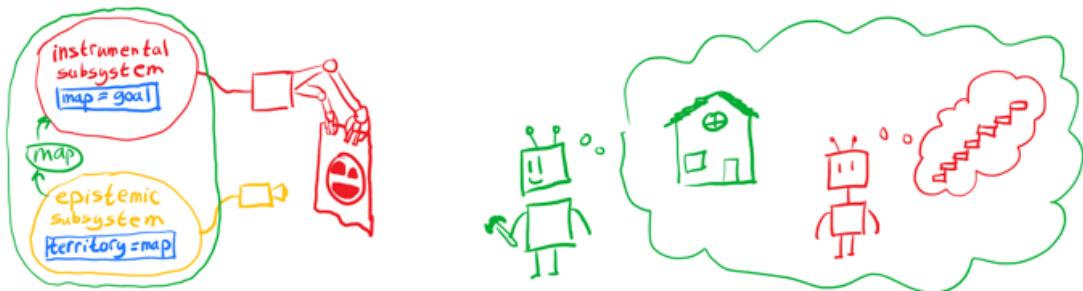
However, this kind of indirection seems to introduce an incentive for misalignment!

As we saw in the simple epistemic/instrumental example, as soon as we start optimizing some sort of expectation, rather than directly getting feedback about how we're doing on the metric that's actually important, we may create perverse incentives — that's Goodhart's Law.



This is similar to the way we wanted successor agents to robustly point at values, since it is too hard to write values down.

However, in this case, learning the values of the larger agent wouldn't make any sense either; we really want our subsystems & subgoals to be smaller.



Now, it might not be that difficult to solve subsystem alignment for subsystems which humans entirely design, or subgoals which an AI explicitly spins up. If you know how to avoid misalignment by design, and robustly delegate your goals, both problems seem solvable.

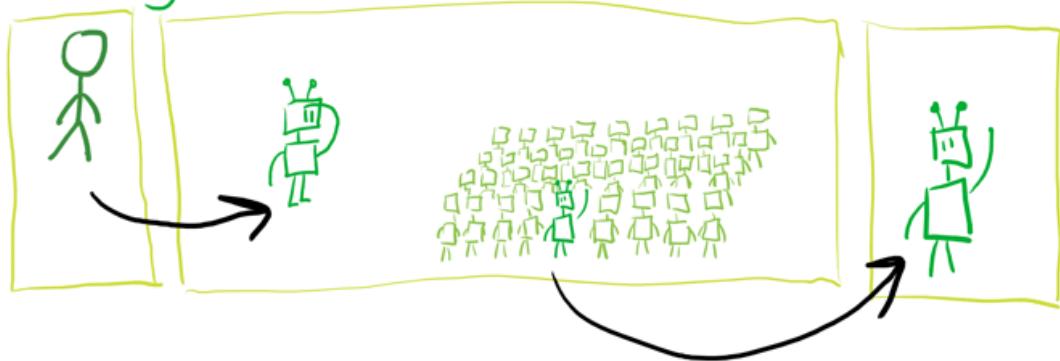
However, it doesn't generally seem possible to design all subsystems in so explicit a manner. At some point, in solving a problem, you've split it up as much as you know how to, and must rely on trial & error.

In other words,

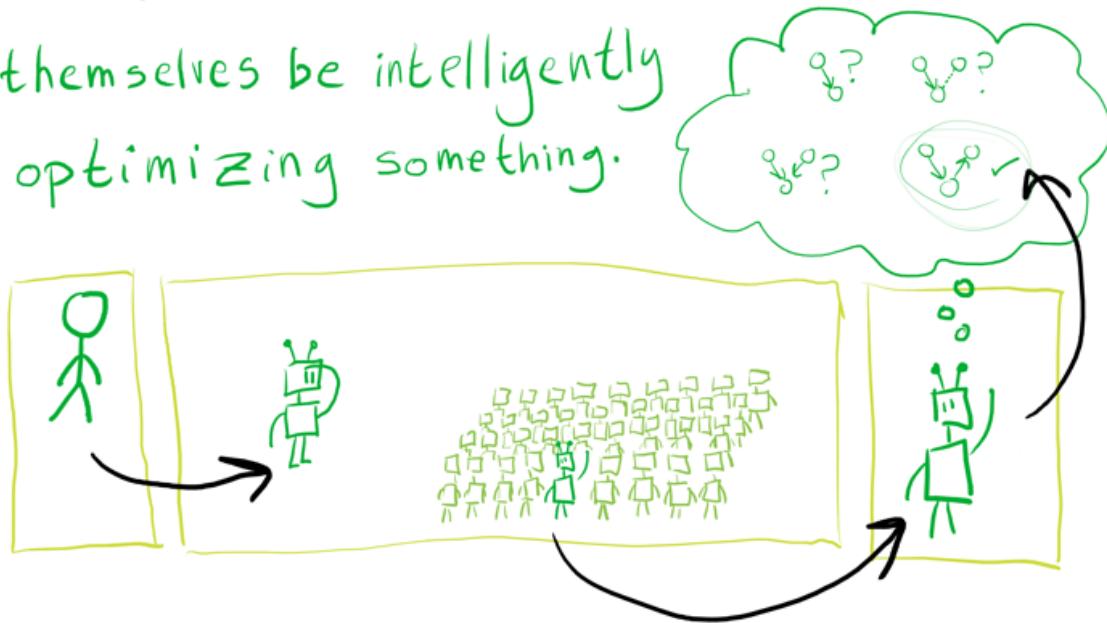
Search: solving a problem by looking through a rich space of possibilities, which may itself contain misaligned subsystems.



Machine learning researchers are quite familiar with the phenomenon: it is easier to write a program which finds a high-performance machine translation system for you, rather than directly write one yourself.



So, couldn't this process go one step further? For a rich enough problem, the solutions found via search might themselves be intelligently optimizing something.



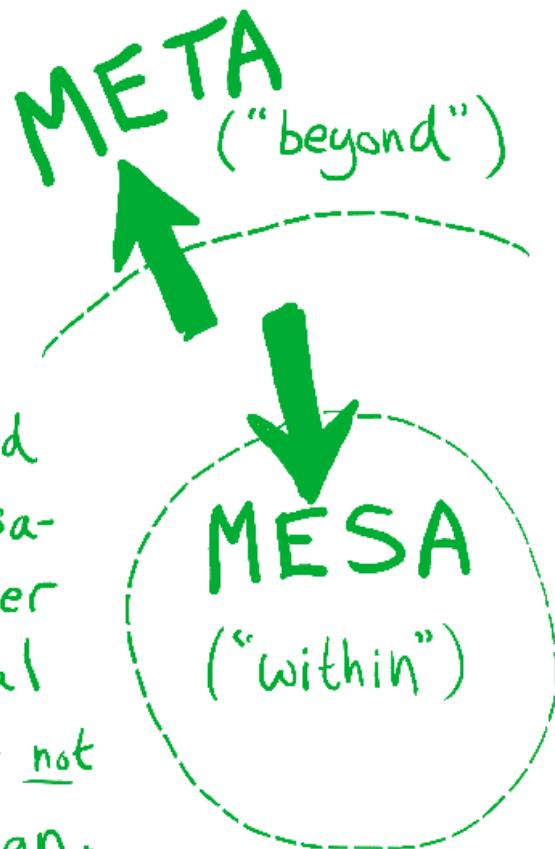
This might happen on accident, or be purposefully engineered as a strategy for solving difficult problems.

Either way, it stands a good chance of exacerbating Goodhart-type problems — you've basically got two chances for misalignment where you previously had one.

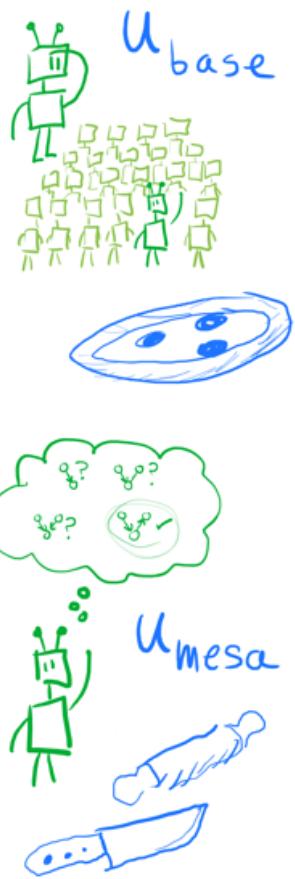
Let's call the original search process the "base optimizer", and the inner search process a "Mesa-optimizer".



"Mesa-" is the opposite of "meta-". Whereas a meta-optimizer is an optimizer designed to find a new optimizer, a mesa-optimizer is any optimizer generated by the original optimizer — whether or not it is produced by design.



"optimization" and "Search" are ambiguous terms; I'll think of them as any algorithm which can be naturally interpreted as doing significant computational work to "find" an object that scores highly on some objective function.



The objective function of the base optimizer is not necessarily the same as that of the mesa-optimizer.

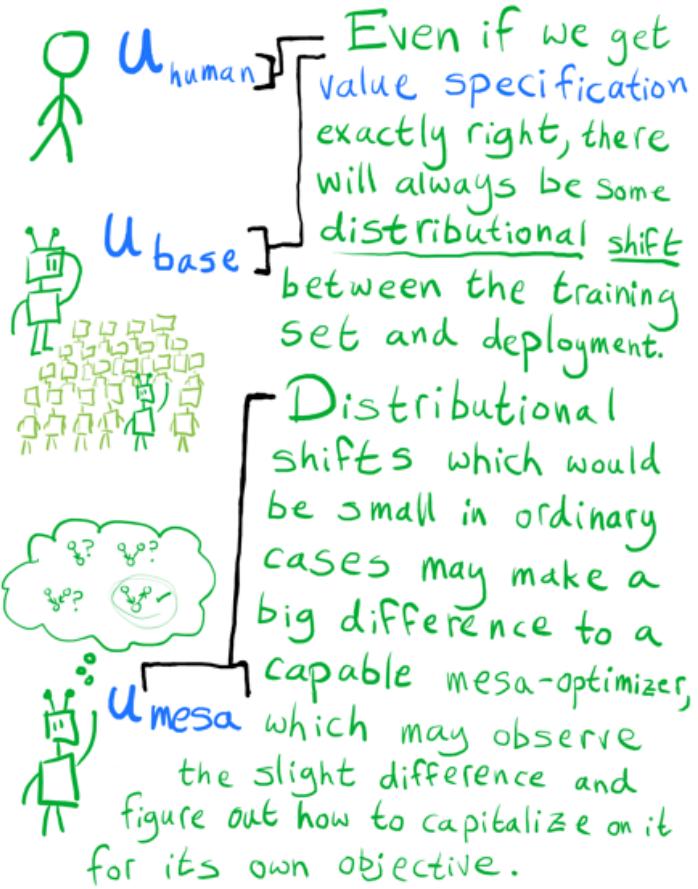
If the base optimizer wants to make pizza, the mesa-optimizer may enjoy kneading dough, chopping ingredients, et cetera.

The new objective function must be helpful for the original, at least in the examples the base optimizer is checking. Otherwise, the mesa-optimizer would not have been selected.

However, the base optimizer must reduce the problem somewhat; there is no point to it running the exact same search. So, it seems like its objective will tend to be like good heuristics; easier to optimize, but different from the base objective in general.

Why might a difference between base & mesa-objectives be concerning, if the new optimizer is scoring highly on the original objective anyway?

It's about the interplay with what's really wanted.



Actually, to even use the term “distributional shift” seems wrong in the context of embedded agency. The world is not i.i.d.

The analog of “no distributional shift” would be to have an exact model of the whole future relevant to what you want to optimize, and the ability to run it over and over during training.

So, we need to deal with massive “distributional shift”.

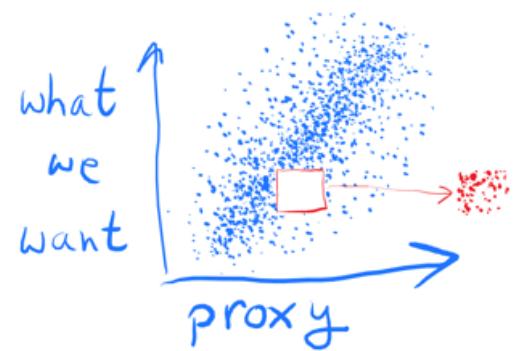
Also, there's the possibility that the mesa-optimizer becomes aware of the base optimizer.



In that case, it might start explicitly trying to do well on the base objective function in order to be kept around, while looking for any signs that it has left training and can stop pretending. This creates a version of Treacherous Turn.



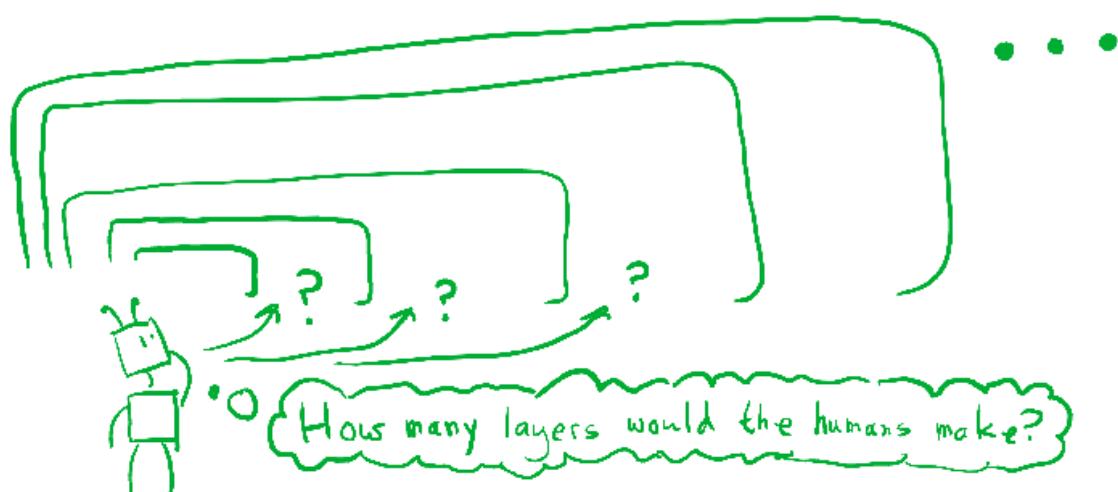
This is the same story we saw in Adversarial Goodhart: there is something agentic in the search space, which responds to our choice of proxy in a way which makes our proxy a bad one.



You might try to avoid treacherous turns by repeatedly simulating "end of training, time for deployment" during training, in a kind of nested-dream setup.



The goal is to test enough layers of simulation that you can conclude there won't be a treacherous turn in the real deployment case.



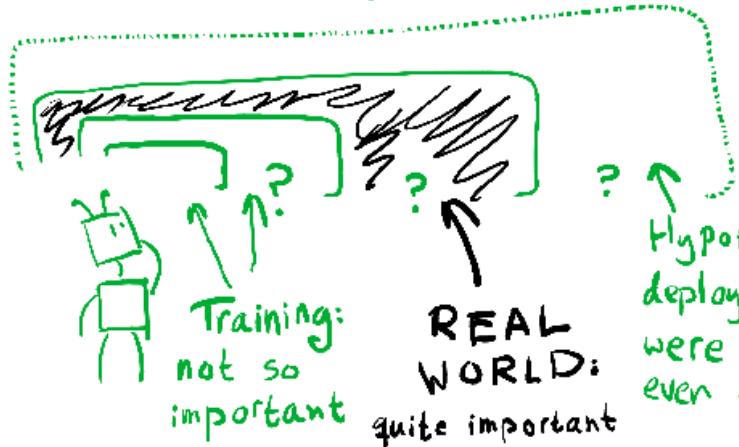
Unfortunately, convergence for this kind of learning is going to be poor.

↖ ↗ ↖ ↗ ↖ ↗ ↖ ↗ ↖ ↗ ↖ ↗ ↖ ↗ ↖ ↗ ... →

Ordinarily in machine learning, good performance means good average-case performance.

But a treacherous turn is an "error" which can be carefully placed to do the most damage. We want to ensure this doesn't happen.

The problem is, in part, that some outputs are much more important than others. Deployment is more important than training, and certain critical/vulnerable moments during deployment will be important. We want to be particularly sure to get important things right, not just low average loss.



Hypothetical really-real deployment case in which we were only a Sim; doesn't even exist (probably)

But we can't solve this by telling the system what's important. Indeed, it seems we hope it can't figure that out — we are banking on being able to generalize from performance on less-important cases to more-important cases.

This is why research into machine learning techniques which avoid rare catastrophes (or "traps") is relevant to the problem of inner alignment.

It is difficult to trust arbitrary code — which is what models from rich model classes are — based only on empirical testing.

Consider a highly simplified problem: we want to find a program which only ever outputs one. Zero is a catastrophic failure.

1 1 1 1 1 1 Ø 1 1 1 ...

If we could examine code, this problem would be easy. But, the output of machine learning is often difficult to analyze; so, imagine we can't understand code at all.

```
while true  
    print 1  
end
```



Now, in some sense, we can trust simpler functions more. A short piece of code is less likely to contain a hard-coded exception.



Let's quantify that.

Consider the set of all programs of length L .



Some programs p will print 1 for a long time, but then print 0. We're trying to avoid that. Call the time-to-first-zero w_p . ($w_p = \infty$ if the program p is trustworthy, i.e., never outputs zero.)

The highest finite w_p out of all length- L programs is a form of the Busy Beaver function, so I will refer to it as $BB(L)$.

If we wanted to be 100% sure that a random program of length L were trustworthy, we would need to observe $BB(L)$ ones from that program.

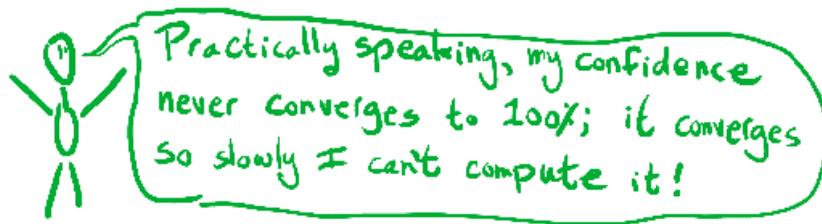


Now, a fact about the Busy Beaver function is that $BB(n)$ grows faster than any computable function.

So this kind of empirical trust-building takes uncomputably long to find the truth, in the worst case.

What about average case?

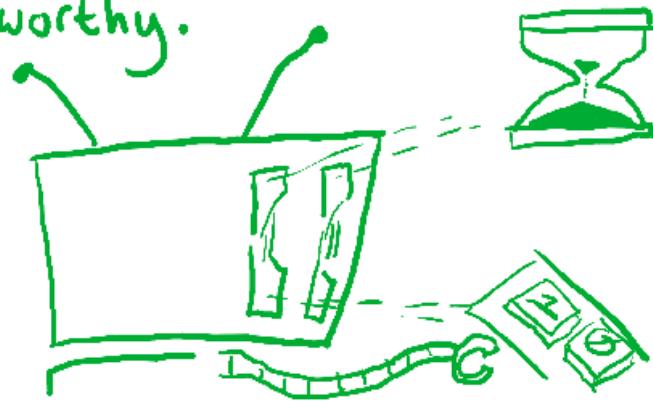
If we suppose all the other length- L programs are easy cases, there are exponentially many length- L programs, so the average is at best $\text{BB}(L)/\exp(L)$. But exponentials are computable. So $\text{BB}(L)/\exp(L)$ still grows faster than any computable function.



So, while using short programs gives us some confidence in theory, the difficulty of forming generalized conclusions about behavior grows extremely quickly as a function of length.

If length restrictions aren't so practical, perhaps restricting computational complexity can help us?

Intuitively, an inner optimizer needs time to think in order to successfully execute a treacherous turn. So a program which makes its conclusions more quickly might be more trustworthy.



However, restricting complexity class doesn't get around Busy Beaver type behavior. High-W_p strategies can be slowed down even further with only slightly longer L.

A binary sequence diagram illustrating a slow-down in a high-W_p strategy. The sequence starts with a string of ones, followed by a red zero, and then continues with a long string of ones ending in a red zero. A green wavy arrow points from the first red zero to the second red zero, indicating a significant delay or loop in the computation path.

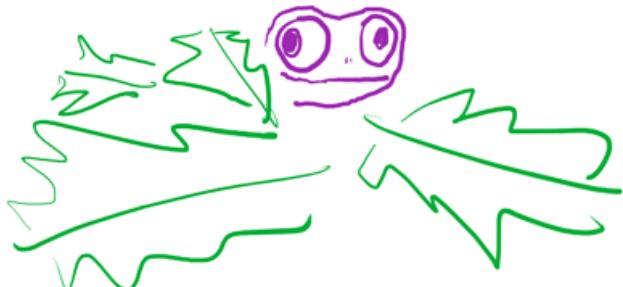
1 1 1 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 0

If all of these problems seem too theoretical, consider the evolution of life on Earth.



Evolution can be thought of as a reproductive fitness maximizer.

Intelligent organisms, then, are meta-optimizers of evolution.



Although the drives of intelligent organisms are certainly correlated with reproductive fitness, they want all sorts of things.



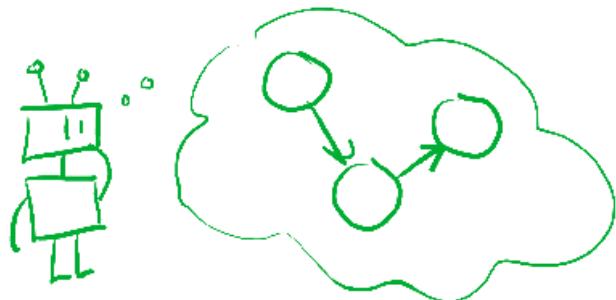
There are mesa-optimizers who have come to understand evolution, and even to manipulate it at times.

So, powerful and misaligned mesa-optimizers seem like a real possibility, at least with enough processing power.

Problems seem to arise because you try to solve a problem which you don't yet know how to solve by searching over a large space and hoping "someone" can solve it.

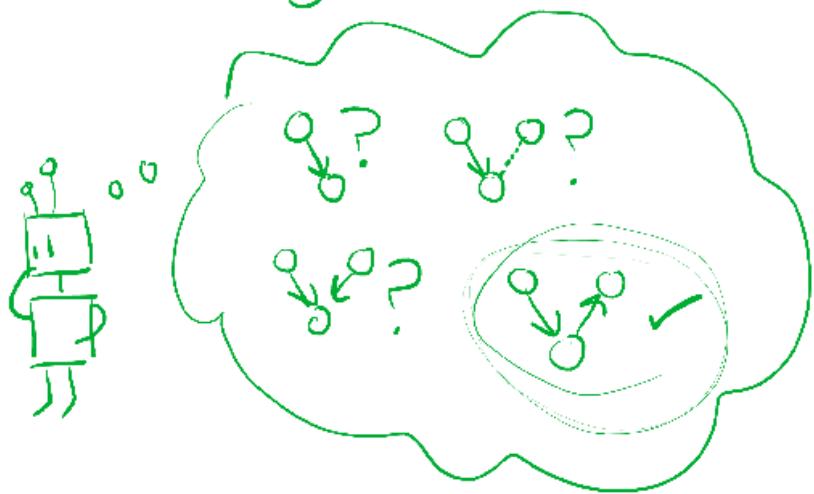


So, if the source of the issue is the solution of problems by massive search, perhaps we should look for different ways to solve problems.



Perhaps we should solve problems by figuring things out.

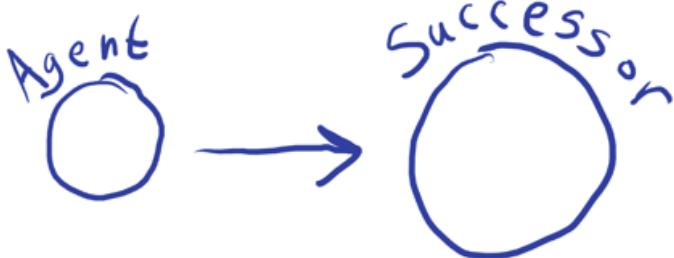
But, how do you solve problems
which you don't yet know how to solve,
other than by trying things?



Let's take a step back.

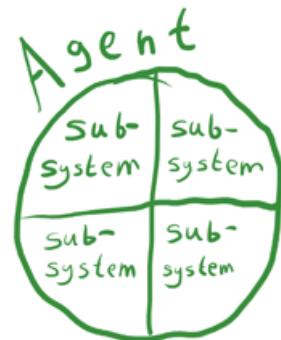


Embedded World-Models is about how to think at all, as an embedded agent.



Robust Delegation is about building trustworthy successors/helpers.

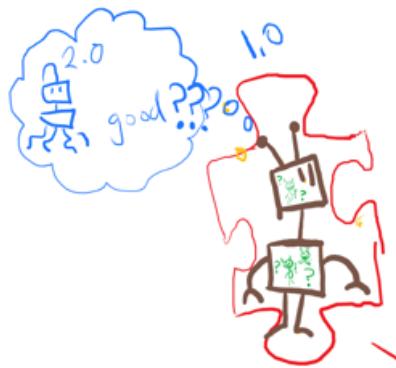
Decision Theory is about how to act.



Subsystem Alignment is about building one agent out of trustworthy parts.

Or, putting it differently:

- We don't know how to think about environments when we're smaller
- To the extent we can do that, we don't know how to think about consequences of actions in those environments
- Even when we can do that, we don't know how to think about what we want
- Even when we have none of those problems, we don't know how to reliably output actions which get us what we want!



Hopefully it's now clear
why we care about embedded
agency...

... Or, rather,
hopefully you
now feel as
confused as
we do.



Embedded Curiosities

Crossposted from the [AI Alignment Forum](#). May contain more technical jargon than usual.

A final word on curiosity, and intellectual puzzles:

I described an embedded agent, Emmy, and said that I don't understand how she evaluates her options, models the world, models herself, or decomposes and solves problems.

In the past, when researchers have talked about motivations for working on problems like these, they've generally focused on the motivation from [AI risk](#). AI researchers want to build machines that can solve problems in the general-purpose fashion of a human, and [dualism](#) is not a realistic framework for thinking about such systems. In particular, it's an approximation that's especially prone to breaking down as AI systems get smarter. When people figure out how to build general AI systems, we want those researchers to be in a better position to understand their systems, analyze their internal properties, and be confident in their future behavior.

This is the motivation for most researchers today who are working on things like updateless decision theory and subsystem alignment. We care about basic conceptual puzzles which we think we need to figure out in order to achieve confidence in future AI systems, and not have to rely quite so much on brute-force search or trial and error.

But the arguments for why we may or may not need particular conceptual insights in AI are pretty long. I haven't tried to wade into the details of that debate here. Instead, I've been discussing a particular set of research directions as an *intellectual puzzle*, and not as an instrumental strategy.

One downside of discussing these problems as instrumental strategies is that it can lead to some misunderstandings about *why* we think this kind of work is so important. With the "instrumental strategies" lens, it's tempting to draw a direct line from a given research problem to a given safety concern. But it's not that I'm imagining real-world embedded systems being "too Bayesian" and this somehow causing problems, if we don't figure out what's wrong with current models of rational agency. It's certainly not that I'm imagining future AI systems being written in second-order logic! In most cases, I'm not trying at all to draw direct lines between research problems and [specific AI failure modes](#).

What I'm instead thinking about is this: We sure do seem to be working with the wrong basic concepts today when we try to think about what agency is, as seen by the fact that these concepts don't transfer well to the more realistic embedded framework.

If AI developers in the future are *still* working with these confused and incomplete basic concepts as they try to actually build powerful real-world optimizers, that seems like a bad position to be in. And it seems like the research community is unlikely to figure most of this out by default in the course of just trying to develop more capable systems. Evolution certainly figured out how to build human brains without "understanding" any of this, via brute-force search.

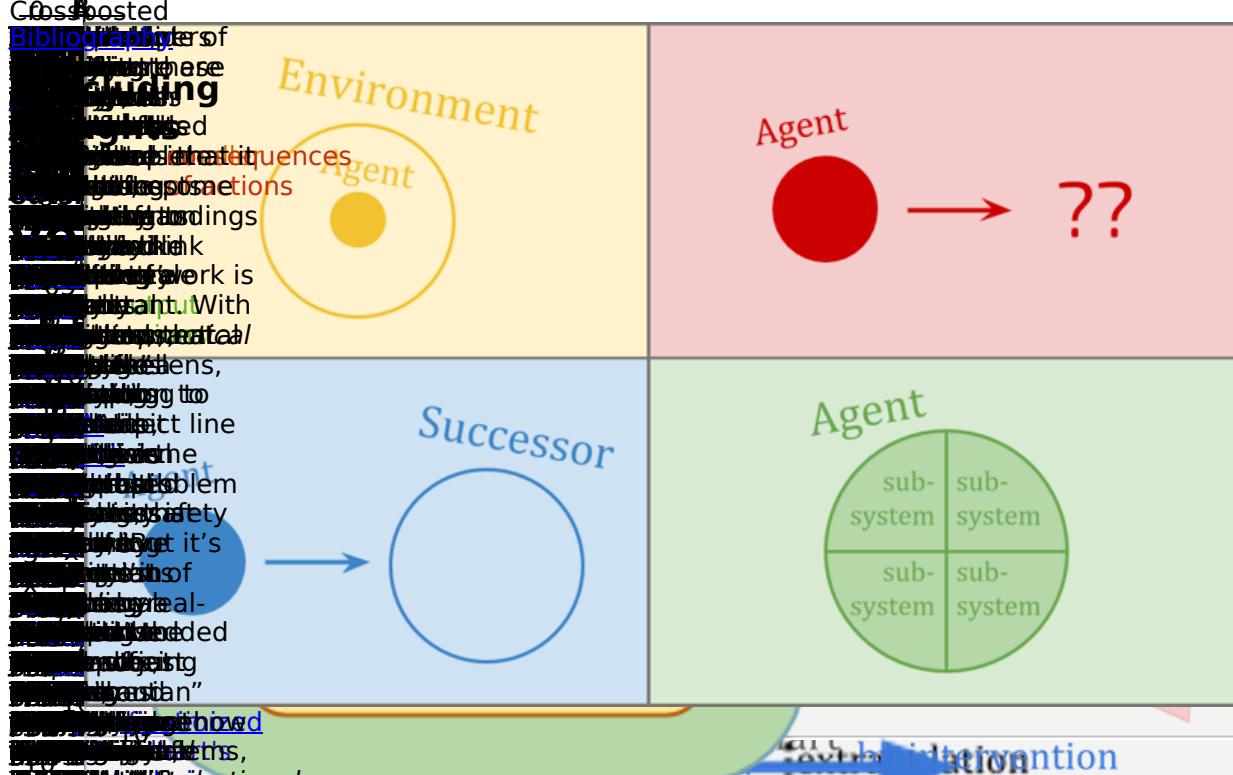
Embedded agency is my way of trying to point at what I think is a very important and central place where I feel confused, and where I think future researchers risk running into confusions too.

There's also a lot of excellent AI alignment research that's being done with an eye toward more direct applications; but I think of that safety research as having a different type signature than the puzzles I've talked about here.

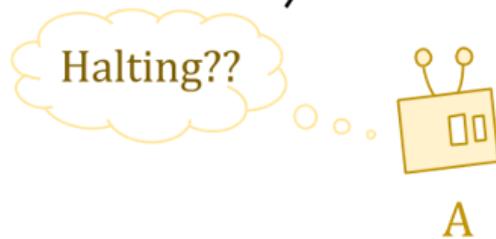
Intellectual curiosity isn't the ultimate reason we privilege these research directions. But there are some *practical* advantages to orienting toward research questions from a place of curiosity at times, as opposed to *only applying the "practical impact" lens* to how we think about the world.

When we apply the curiosity lens to the world, we orient toward the sources of confusion preventing us from seeing clearly; the blank spots in our map, the flaws in our lens. It encourages re-checking assumptions and attending to blind spots, which is helpful as a psychological counterpoint to our "instrumental strategy" lens—the latter being more vulnerable to the urge to lean on whatever shaky premises we have on hand so we can get to more solidity and closure in our early thinking.

Embedded agency is an organizing theme behind most, if not all, of our big curiosities. It seems like a central mystery underlying many concrete difficulties.



Causal Goodhart



even
more
money.

