This draft is old. Here is the new one: https://docs.google.com/document/d/1b25gpDx0bC4KJ1RWlDOvrQhBrxkRPCIbMuTxpF9YPa0/edit

In 2012, Holden Karnofsky wrote the article Thoughts on the Singularity Institute (SI), from which this is an excerpt:

> One of the things that bothers me most about SI is that there is practically no public content, as far as I can tell, explicitly addressing the idea of a "tool" and giving arguments for why AGI is likely to work only as an "agent."

And here is an excerpt from Eliezer Yudkowsky's response:

> Tool AI wasn't the obvious solution to John McCarthy, I.J. Good, or Marvin Minsky. Today's leading AI textbook, Artificial Intelligence: A Modern Approach - where you can learn all about A* search, by the way - discusses Friendly AI and AI risk for 3.5 pages but doesn't mention tool AI as an obvious solution. For Ray Kurzweil, the obvious solution is merging humans and AIs. For Jurgen Schmidhuber, the obvious solution is AIs that value a certain complicated definition of complexity in their sensory inputs. Ben Goertzel, J. Storrs Hall, and Bill Hibbard, among others, have all written about how silly Singinst is to pursue Friendly AI when the solution is obviously X, for various different X. Among current leading people working on serious AGI programs labeled as such, neither Demis Hassabis (VC-funded to the tune of several million dollars) nor Moshe Looks (head of AGI research at Google) nor Henry Markram (Blue Brain at IBM) think that the obvious answer is Tool AI. Vernor Vinge, Isaac Asimov, and any number of other SF writers with technical backgrounds who spent serious time thinking about these issues didn't converge on that solution.

This exchange reminds me that it's hard to know what other people see as "obvious". Trying to properly understand the perspective of others can be a lot of work, and although I have put in many hours reading the thoughts of alignment experts, I am very far away from having a full overview and understanding of people's opinions and ideas. My reason for mentioning this is that I find if various ideas are new/interesting/helpful, or old/obvious or wrong/misguided.

Trying to understand another person's perspective in detail is not a trivial task, and whereas AI alignment is concerned I am a hobbyist. I suspect there may be some overlap between stuff written here and Eric Drexler's writings on Comprehensive AI Services, but I don't know how much or little. And I also suspect overlap with the ideas of Paul Cristiano, but I'm not sure to which degree. I am aware of there being concepts with names such as AI safety by debate and Iterated Distillation and Amplification, and I have been able to find posts with titles such as Bootstrapped Alignment, but - well, anyway, I'll just get on with it. I suspect there may be some interesting stuff in here, but if I am repeating other peoples ideas without adding anything substantial, or in some other way missing the mark, then sorry about that.

A lot of AI alignment theory is sort of concerned with the question of *"how might we align an AI-system before/at the stage when it becomes superintelligent?"*, while the focus of this text is a bit different. It is more focused on *"presuming that we have a superintelligent AI-system, which isn't necessarily aligned but does at least pretend to be aligned, how might this AI-system be used to set up a more safe system?"*. It seems to me that in such situation there would be some very powerful techniques at our disposal, and that ways to proceed and make full use of these techniques might be very worthy of more deep/careful/comprehensive analysis. Even if the initial AGI is assumed to really be aligned (a much more desirable and less risky situation!) it would be advisable to use techniques such as these as an additional level of security/alignment-confirmation.

Section 1: Describing hypothetical preconditions (with anthropomorphic framing)

I'm not familiar with deep learning at a technical level, but my impression is that a lot of modern machine learning is a bit like this:

And the hypothetical situation I imagine is a situation where we have developed a superintelligent AGI, but that we don't really understand how it's work - I'll assume that it's pretty much a "black box".

I assume that ***maybe*** the capabilities of the system went quite quickly from *"wouldn't be able to empty a dishwasher or reliably drive a car"* to *"superintelligent along more or less any dimention we care about"* (my intuitions about FOOM are probably more Yudkowsky-like than Hanson/Christiano-like, but I don't really know, and I feel more unsure than I used to).

This superintelligence is able to understand us well (when instances of the system are given sufficient data/input). And in a superficial sense it ***seems*** perfectly aligned. If it is asked to for example give us ideas about AI alignment, it will provide lots of valuable insight. And if it is asked to write code that is

optimized with human readability in mind, it will provide really well-written code. It isn't evasive, and the answers it gives (if/when it is allowed to answer in free-form) will seem as if they are crafted to be more or less maximally helpful and non-deceptive.

So it is a powerful tool that we have at our disposal, to put it mildly (in the hypothetical situation I'm assuming). But as explained in detail by Eliezer Yudkowsky and others - just because it seems as if it is aligned, does not mean that it is aligned (or even that it is likely to be so). And banking on not being tricked by an intelligence that is vastly more intelligent than ourselves, even though being tricked in just one subtle way could mean the death of everyone on Earth (and maybe also increased s-risks) - well, that's a risky activity.

We dealing with an instance of an AI-system, we choose what info/input we give it, what we ask it to do/answer, and last but not least: within which limitations in expressivity it may answer. And thus there are huge differences in risk between different ways of using an AI-system. We could imagine there being a sort of "scale". Towards one end of the scale would be to give the AI access to the internet, or to implement complex machinery based on the AIs instructions. Towards the other end of the scale would be e.g. only giving it multiplication-questions (*“Is 3*3=9?”*), and letting its only line of communication be to answer “Yes” or “No”.

Oracle/genie AI-system

Interface that limits expressivity

The avenues for an unaligned oracle/genie to attempt trickery depends on the kinds of questions asked and on the restrictions in expressivity when answering questions (and also on the security of the environment where the code is run, but that's outside the scope of this text).

One of the main themes in this text is whether there might be sequences of steps by which we might use plausibly-unaligned AIs so as to greatly reduce total risk. Might there for example be relatively safe ways of using an uninterpretable superintelligence to create superintelligences that are much less "black box"-like? Might there be ways of creating aligned superintelligences in a relatively short span of time with the help of a superintelligence, even if the superintelligence we start out isn't really aligned? And if so, to which degree would it be possible to minimize the risk involved in such a process? I don't properly answer any of these questions in this text, but I do brainstorm and try to provide pointers for further analysis. There are huge degrees of freedom for how AI-systems can be used, and the question of *"which possible uses of AGI-systems in which sequence might minimize total risk?"* seems to me like a question that is worthy of more analysis (and where we can do much useful analysis without knowing the specifics of how the initial AGI would work).

Some security-tips that I feel like mentioning, even though I consider them mostly outside the scope of this text:

- Never letting a machine that runs that AI-system (or has run the AI system previously) have access to the internet or receive USB-sticks, and also not letting it network with machines that do.
- Not forgetting that digital machines also are a part of the physical world, and try to take into account (e.g. considering to only run computers containing AI-system code inside a faraday cage).
- Valuing security mindset, and hiring people with security mindset into the organization. Instill into the people that *"too paranoid"* is better than *"not paranoid enough"*.
- Having a group of people with security mindset needing to manually approve queries that are asked of the AI-system, and whether the response should be sent on to the rest of the team (+ having

narrow/specialized AI-systems label responses that seem like they might be suspect).
- Trying to keep in mind that the AI-system doesn't just learn from what you explicitly tell it, but from pretty much anything that it can make use of in Bayesian-like inference.
- Not taking for granted that the AI-system isn't hiding its capabilities, and thinking that security protocols that presume plausibility of superintelligent capabilities are much better to start with too early than too late!
- NSA-like procedures from preventing code and design secrets from being shared with less scrupulous actors.
- Attempting to consolidate different teams at the forefront (Open AI, DeepMind, etc) into one cohesive entity, as a way to make strong arms race dynamics less likely.
- Trying insofar as possible to ensure that the AI-systems code is theoretically unable to "hack itself" out of the "box", regardless of the specifics of the code being run, and attempting to ensure this through multiple assurances.
- Taking into account that account execution-times (of calls to memory, etc) and that sort of thing can be a potential source of Bayesian-like inference for an AI-system.
- Think deeply about the question of whether there might be ways for an AI-system to find "deleted" data on the computer (e.g. from previous AI-instances).
- Using "mathematical" proofs to verify as much as possible of the behavior of the operating system, CPU, hard drive, memory, etc, etc.
- Not using the same "instance" of the AI-system for all questions (that is to say, by default the AI-system should not have direct or indirect access to previous question-history or input).

The above is not an attempt at an exhaustive list, and as mentioned I consider these kinds of things to be mostly outside the scope of this text.

Section 2: Getting help with code (modification, review, generation, etc)

Getting help from from an AI-system to create code is much safer than using an AI-system to e.g. create nanofactories (presuming that the code that is generated will be run under the same hopefully safe conditions as the ones already in use for the initial superintelligence).

Also when it comes to coding-assistance there is a "scale" in regards to how much of an avenue a superintelligence would have for committing "malice". One of the safer examples of code-assistance is to refactor code, but in ways that provably doesn't change behavior. The AI-program could be required to provide “proofs” that a given change does not change behavior. A format for specifying such "proofs" is something that humans could provide, but it is also something that the AI could provide (but with humans outlining what the kind of format they want). Any "inference-rule" for "proof-steps" in such a format would need to make intuitive sense, and both humans and other AI-systems could try to find counterexamples (examples of pieces of code that are "proven" to behave the same, but where they actually have different output given input x, or make non-trivially different use of computer resources in ways not advertised by the proof result). A superintelligence could also make proofs about what exactly what is the "difference" in behavior between 2 pieces of code (e.g., *"function x and function y will behave the same, except if the input is c"*, or *"if code piece a was added here to function b, then function b would always have the same output as function c"*).

One purpose of rewriting code might be to make it much faster (and at some stage, if a group of people with security mindset trusts it to be sufficiently safe, maybe even writing directly in assembly or making specialized computer chips based on an AI system's design). Another purpose of rewriting code could be to make it more easy to read and understand for humans. A superintelligence could be able to deduce well based on unstructured/vague specification what makes code more readable to humans. So one might imagine just telling an AI-system: *"Refactor the code to make it better organized, easier to understand, and harder to misunderstand or not notice bugs"*. But it would also be possible to set up more explicit ways in which the readability/understandability of code is "scored".

Another way AI-systems could help with code would be to point out potential bugs/issues with the code. And in cases where it was put to such a purpose,

the people controlling the AI would be able to choose the limits within which the expressivity of the AI-system is restricted. One could imagine the AI-system not being allowed to write text at all, but instead pointing out potential issues through other means. For example, it could highlight 1 or more passages of code that the user should look at, and/or it could point out example-inputs and the corresponding outputs (either for the system as a whole or certain sub-sections of the code). If it was allowed to use text to explain, then the text could be strictly limited in terms of length/content/structure. On the scale between "safe" and "less safe", I'm thinking that any help from the AI to point out bugs probably gives more opportunity for mischief than e.g. rewriting code in ways that provably don't change behavior. But there would be a lot less room for mischief compared with many other tasks.

An AI-system can also be asked to create code entirely by itself. Smaller pieces of code, or larger pieces of code - everything from simple functions, to big sub-components, to entire code-bases. And the specifications for the code it is asked to write - those may be detailed or they may be short - they may be precise or they may be vague (or some of both). And another axis of variability is how easy the code is to review - how hard or easy it is to verify that the code does what it is said to do, and that it has been properly optimized for the qualities that humans want in it (in terms of understandability, the possibility for scalable oversight, consistently following design principles that make it as hard as possible to "hide" malicious behavior, etc).

When generating pieces of code, it may be best to not output one "piece of code", but rather a datastructure where different alternative code-bases are distributed along multi-dimensional space, where each axis represents things we value (how easy it is to understand for humans, how "powerful" it is, how fast it runs, how easy certain properties/aspects are to test/confirm/prove, etc).

Section 3: "Adversarial" review

I am of the perspective that having "adversarial" AI-systems that review the work of other AI-systems often may be useful (not "the solution" by itself, but potentially a very useful technique when practiced well). Whereas review is

concerned, the review can be done both by superintelligences or by more narrow/specialized systems (or both!). Best of all are tasks that specified in such a way that it is clear/unambiguous to see if it is broken, and even a narrow/specialized system will be able to find all examples of the specification being broken (if it is broken anywhere).

How we can ask for things (e.g. pieces of code) in such a way that it is easy to verify, and hard to "cheat", is IMO an area that is worthy of plenty of brainstorming and analysis.

If we have access to narrow intelligences that we more easily can confirm to be aligned (at the task in question), then the scope of what can be reviewed in an exhaustive way may increase greatly.

And even review where the success of the reviewer at noticing problems is probabilistic can make it much harder for a superintelligence to "cheat". Texts describing what (large or small) pieces of code do, is one thing that can be reviewed. And it's an example of something where a narrow/specialized reviewer may be able to reliably notice some types of problems, while other types of problems may be hard/infeasible to notice reliably.

Section 4: Asking oracles/genies for narrow/specialized "generators" ("reviewers", "answer-generators", "simulation-generators", "simulation-generator-generators", etc)

You could ask an oracle/genie to look for inconsistencies between a piece of code and a description of what that piece of code is supposed to do. But instead of (or in addition to) asking instances of the oracle to do that task directly, you may ask it to provide you a general-purpose function/system for doing that.

As mentioned, we may want to use narrow/specialized systems to review the work of superintelligent AI-systems (sometimes in addition to other superintelligent AI-systems). And one way to obtain such systems would be to have an AGIs make them. The distinction between a narrow/specialized system, and a system that has "general" intelligence, may be blurry, but at the same time we do have clear ideas of many of the things/aspects to be on the lookout for. And as mentioned, a code base (which more or less is what a

"reviewer" would be) can absolutely be "scored" for clarity/readability (although not with complete accuracy that covers any type of issue in an exhaustive way).

Videos such as this one show that as humans we have limited ability to absorb everything we see in a precise/comprehensive way. But at the same time, there are limits to what we can be made to not notice. If a superintelligence provides a narrow/specialized reviewer - in very well-written source code, and coupled with textual and schematic descriptions of what the reviewer does - then that probably gives it a lot of rooms for shenanigans - but there are probably also lots of avenues/methods for comprehensive review.

A desirable property of "reviewers" is if they can be tested, and the more reliably they can be tested the better. Both testing with lots of examples (so as to see if reliably notices instances of x from training data), and testing other AI-systems (including superintelligent ones) are asked to search for examples of input that can "fool" the reviewer.

Another desirable property is that is may make it harder (but not necessarily impossible) to "fool" us intentionally. Whether a general-purpose (but narrow/specialized) reviewer does what it should do may be easier to evaluate than evaluating an object-level answer coming directly from an AGI. And it can also help make it harder for different superintelligences to "collude" (compared to when one superintelligence is asked to evaluate the source code of another superintelligence, as opposed to being asked to evaluate a narrow/specialized "reviewer").

These desirable properties hold more generally for when asking an AGI to answer something, design something, or code something. For example, rather than being asked to simulate the structure of a machine that would result from a step-by-step process (involving things like e.g. DNA synthesis and/or peptide sequencing and/or 3D printing), it may be asked for a "simulation-generator". Such a generator may then be evaluated (like, you can look at the code and the principles the code makes use of, and see if they make sense), and it can be backtested against things that already have been made in the real world.

Maybe it sometimes also could have merit to ask for "answer-generator-generators". The purpose of that could be to make it harder to "hide" shenanigans by increasing generality and including less specific information. For example, a "simulation-generator-generator" could be agnostic to laws of physics (and could be asked of AI-systems where we have minimized the amount of info we give them about the universe).

Sometimes it may make sense that the "generators" are "stacked". For example, maybe a "simulator" uses a lot of compute, and that therefore we want a more approximate version that runs faster. Instead of (or in addition to) directly having the AI make the approximate version, we may have AIs generate "generators" for doing "approximation", and feed the high-fidelity/high-compute "generator" that takes on the role of an "approximator".

Maybe often one would be faced by questions such as *«should I use this or this answer-generator?»*, *«should this or that AGI-architecture be asked to provide the "reviewer"?»*, *«should set A or set B be the training-data of the AGI-architecture-instance I ask to provide the "generator"?»*, etc. To such questions a good answer may often be: Why not both?

Compute may be a practical consideration, and another consideration (that I won't cover much in this text) may in some instances be the possibility of suffering subroutines. But besides concerns, a good attitude when it comes to "generators" and "reviewers" is probably *"the more the merrier"*. Trying lots of stuff (bonus if they are based on widely different architectures and principles), and checking the degree to which their output converges.

One avenue of further thinking/brainstorming/analysis might be the question of which properties of "generators" that affect how hard/easy it is for

Section 5: Argument-trees with a new type of more expressive formalism/"logic"

> If this section is draining to read, you could skip it and come back to it later.

So maybe in some sense higher-order predicate logics are "structurally equivalent" to the kind of thing I have in mind in terms of what theoretically can be done with it, and maybe that also is the case for proof assistants based on dependent type theories such as Coq and Agda. Nontheless, these formats (based on my far-from-comprehensive understanding of them) feel limited to me, at least when it comes to representing real-world concepts that aren't purely about "mathematical" things:

Changing code to make it more easy to understand/read for humans is more safe when it can be done in ways where it is "mathematically" proven that the behavior of the code remains unchanged:

- They stray unnecessarily far from human language and other forms in which the human brain is comfortable with having things presented (by "unnecessarily" I mean "in ways where AFAIK no precision/clarity/specificity is gained)
- As far as I can see (and my familiarity is limited) they don't seem to be built to handle really complex types
- A lot of the concepts that we deal with natively as humans, and that can be stated relatively crisply, are concepts that these formalisms seem to not be designed to express (so expressing them becomes - well, if not impossible, then at least awkward/"hacky")

Meanwhile, human language certainly also has it's limitations for users where we want to express arguments/statements/beliefs/questions/etc as precisely as is feasible:

- Clauses are not made explicit/visible (and how to parse them is often ambiguous), and the same goes for "variables" like "he", "she", "it", etc
- No languages are really designed for nuance/detail/precision, making the tradeoff between nuance/detail/precision and brevity/complexity of text is unnecessarily "costly"
- It is not made easy to disambiguate, and describe as well as you can the concept (often a fuzzy and "cluster-like") that you have in your mind
- It is not made obligatory (or even default) to specify the "inference-rules" that is used for each "argument-step" when you argue,

and the same goes for even providing all the relevant "argument-steps"

I think - partly based on hunch/intuition - that it might have merit to have some kind of a formalism that is made to make formal arguments (where each argument-step uses an explicit inference-rule that can compute conclusion/output from assumptions/input), but in a way that is more tailored towards trying to be able to represent more or less anything that logically inclined humans can say in human language while they are trying to be as precise as they can. And with functionality that is tailored towards specifying the meaning of and disambiguating human concepts (including vague "cluster-concepts"). And made from the "ground up" with the intention that inference-steps and various other things are to be computable, and with the intention that it is to be easy to understand (while avoiding misunderstandings insofar as possible) when shown on a screen/GUI - as opposed to e.g. predicate logic, where the starting point is writing things on paper, or Coq, where the starting point is a programming-language that can be written/read as plain strings in any text-editor.

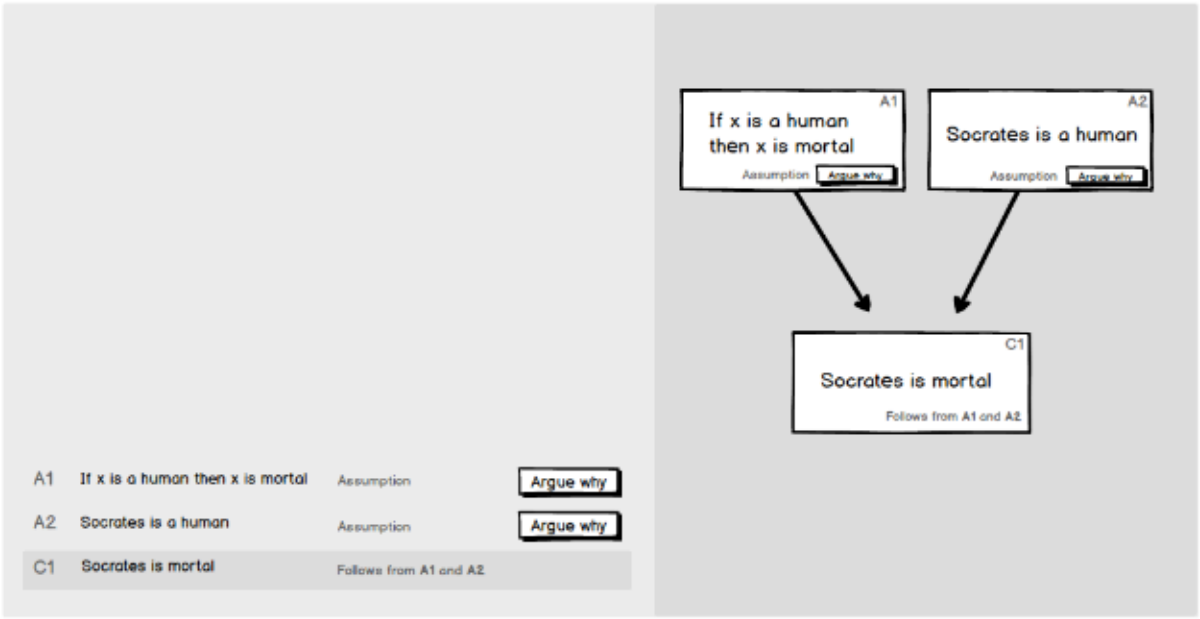
The concept I have in mind (insofar as my thoughts are crystalized), is inspired by a system that I wrote about in my masters thesis. Despite being a master thesis it is a shoddily written document full of spelling mistakes that was put together in a hurry. But nonetheless it could be looked at by those who are especially interested.

Socrates is defined as being the entity that is referred to by the entity that is referred by the Wikipedia-article about Socrates. The words "is" and "mortal" have not been given definitions, but if she chose to do so, Olivia could choose from one of the existing ones or come up with a new one of her own.

Every segment of the sentence (in this case, *"Socrates"* and *"mortal"*) is singled out by coloring the background in a certain shade of color. When this segment is hovered the place where it's defined below changes color/shade, and vice versa. For this simple sentence this is not important, but for other more complex sentences this can make sentences easier to read and sometimes help make sentences that otherwise would be ambiguous unambiguous.

The plus-icons in the sentence can be clicked if Olivia wants to make updates to it. She could for example click the one preceding *"Socrates"* to update it to *"It may be the case that Socrates"*, the one succeeding *"is"* to update to *"is probably"*, or the one succeeding *"mortal"* to update to *"mortal and fond of wearing sandals"*.

Screenshot from thesis



Differences between example and potential real-world applications

While the example above hopefully helps give an idea of what it could be like to use an application that's based on GPF-system, there are several certain and potential differences between the example and what an actual real-world application would be like.

Different GUI

The GUI in the example is not very well thought through, and is not an attempt to correspond well to what a real-world application for constructing computer-verified arguments would look like. Both in terms of layout (which elements that are included and where they are placed) and style (colors, fonts, sizes, etc), there would be large differences.

Other uses than proofs

The example concerns someone who is constructing a proof / computer-verified argument. That is one potential use, but one among many. Other potential uses include but are not limited to:

- Adding information to knowledge bases.
- Asking questions to an answering system (e.g. something similar to WolframAlpha).
- Constructing sentences that are to be translated to one or more other languages and/or formats (and adding new vocabulary when necessary).
- Giving instructions to an application for what is should do.
- Enabling people to (1) use third-party software by constructing statements, questions,

Other screenshot from thesis

I am not under the illusion that human concepts can be made totally explicit. That is to say, for many/most human concepts I don't think we specify some function/definition that clearly defines whether or not (or the degree to which) any instance would belong to that concept or not. But basically, the more you specify about what you mean to refer to when you are referring to a specific concept in some statement, the more precise your are being. Can you mention some examples of things that more-or-less clearly belong to the concept you currently mean to refer to? Cool. Can you you mention some examples of things that more-or-less clearly don't belong in the concept? Cool. Do you have like 127 propositional statements that don't clearly define the borders of the concept in question, but are relevant to the assumed likelyness of fitting within the concept? Awesome. Do you have some description of a space of "thing-space" (perhaps by itself a complicated description) that you actively don't want to be labeled inside/outside the concept? That helps as well. Etc, etc.

From my perspective it feels like an analogy of sorts can be made between logics/formalisms that incorporate probability and logics/formalisms that incorporate ambiguity/concept-specification. A possible "ethos" for using formalisms that represent probability within them could be that probabilities are a part of what constitutes good reasoning, and that therefore there is merit to incorporate it into the formalism (instead of ignoring it, and having the formalism sort of "pretend it doesn't exist"). My hunch is that maybe it could be fruitful to sometimes have a similar attitude towards formalisms that try to incorporate ambiguity/concept-specification, and try to make more of the process of specifying mapping between "variables" and real-world objects be sort of incorporated within the the formalism itself.

One way to use these kinds of formalisms could be to have AI-systems use them when presenting arguments/"proofs". Also, if AI-systems sometimes ask for clarifications of questions/requests/etc - or receive questions/requests/etc in a "disambiguated" format to begin with - then maybe some ways of doing that may involve this kind of a formalism.

When using explicit and argument-trees where each argument-step is "computable", then presumably several of the desirable properties that we associate with formal proofs follow. And it may be easier to show in an explicit

way when reasoning is inconsistent. If ambiguity is involved in such a formalism, then maybe also proofs can be made in regards to ways of classifying also - showing e.g. that a given set of propositions (some of which are "inference-rules") are inconsistent with how humans think about those concepts / can result in use of concepts that is deceptive or not in accordance with what humans would find reasonable/sensical.

When arguments/"proofs" are constructed, how those are scored may also be determined based on how easy they are for humans to understand/follow/verify.

One thing that could be made use of is "predictors" (specialized and preferably "narrow" functions, generated based on principles alluded to in the section about "generators") that try to predict the degree to which argument-steps and propositions will be accepted by humans (not just humans generally of course, but specific types of humans in specific types of contexts - and sometimes even specific events with specific humans). If the correctness of these "predictors" seems to be accurate, then they could also be used to try estimating how easily humans can be tricked (into believing things that are clearly false, and into believing things that are inconsistent). They may also identify best practices for making it as hard as possible to lead humans astray (which can be used when "scoring" argument-trees along that direction).

I am not sure how fruitful the sort of approaches I vaguely outline here in this section would be, but it seems to me like one type of approach that probably (as far as I know) is worthy of attention/consideration.

Another thing to attempt might be to design AI-systems where this kind of format forms the "core" of all reasoning that the system does - with more "messy" and "black box"-like reasoning/processing being something that is used in a function-call like way from the "core" reasoning thread, and with attempts to label what these more "messy" and "black-box" components do and what they wont do. For example, making sure that if function-calls are an image classifier, then that image classifier only does image classification in a straight-forward and narrow/specialized way, without any part of its functionality where complex agent-like behavior is plausible. One question is whether we could try making something like that directly (I would not rule that out, but the idea of doing so feels daunting to me). Another question is whether a superintelligence could help us make something like this, with us specifying the kinds of "lego-blocks" we want the system to be constructed from, and the kinds of properties we want it to have (and the easier to verify these properties the better). Seems quite plausible to me that something like that could work, but I'm not sure. And regardless of whether AGIs can have such an architecture, maybe more narrow/specialized systems might.

Section 6: Brain emulation approximations

There are various types of brain emulations we could imagine. Does it simulate from birth/prenatal, or does it start at a point where the person/animal in question already has memories/etc? Does it simulate one specific possible path, or does it calculate/estimate more of a probability-distribution-space of possible outcomes/states? Does it simulate details of everything that would happen in the brain, or does it also/instead use other methods to make predictions? What kind of environment is the brain simulated to be in? (High welfare environments I would hope, especially in the case of algorithms that have similarities in structure/function to brain function!)

If we did try to get help from a superintelligence to make brain emulations, there are are various approaches we could take, and we would not have to stick to one only. Maybe techniques like the ones alluded to in the section about "generators" could be utilized extensively.

One thing one might ask already now is what kind of experimental data that could useful - not just to generate emulations, but also to test their accuracy. For example, if some AI makes a "generator" that can generate simulations of mice (based simply on being given their DNA, and without being given much other information about mice), might there be some (high welfare) experiments that we might want to do to generate helpful high-fidelity data to test the simulation-generator against? Or do we probably have enough existing data that would be equally helpful? What about experiments where aspects of brain-state are measured in a high-fidelity way before/while/after doing activities such as evaluating arguments, coding, reading other people's code, etc, etc? Could that be useful, or probably not?

Section 7: "Counsel" of "siloed" AI-systems

I would think that probably it is beneficial to generate a sort of "counsel" of "siloed" superintelligent systems, and have those be used in the process somehow. Widely different architectures and widely different approaches to AI alignment could be tried, and instances of these different AI-systems could be used for both providing answers/ideas/code/instructions/etc, and looking over the stuff that other AI-systems provide. Some of these AI-systems (in this system of systems of systems etc) could be made with humans doing much of the highest-level work, but with with lots of AI-assistance in the implementation (and in the iterative process of verification/modification). Other AI-systems included in the "coucil" could be made in one "swoop" by other AI-systems, or based with the help of very high-level queries.

Here are a few examples of types of approaches that various architectures could be based on (very much not an attempt at an exhaustive list!):

- Machine learning through debate (clustering around / building upon ideas by Paul Cristiano and others)
- Trying to crystalize the principles that make human brains aligned, and make use of similar principles but in an idealized form
- Guesses (based on various methodologies for guessing performed by "siloed" systems - including potentially high-fidelity simulations in

> high-welfare conditions) about alignment methodologies that different specific AI researchers and AI researcher teams might have converged on if they had thousands of years to work on the problem (in high-welfare conditions)

One merit of a system of systems, where sub-systems are based on different alignment techniques, is that we can see if the different "siloed" sub-systems come up with different solutions/answers, or if they converge on similar ones (including when they review answers/solutions/argumentation/"generators" from other sub-systems).

Taking why-not-both-ing one step further?

A recurring theme in this text is that often it good to try various solutions/architectures for generating answers, searching for issues/problems, aligning AI-systems, etc, and then see if the output of these system converges. And as a general principle I'd suggest a "more the merrier" approach (though costs are a factor, and the possibility of suffering sub-routines should be avoided/minimized.

I also mentioned that one of the easier things to verify may be "proofs" that 2 programs are equivalent in terms of input/output (and/or other properties we care about). And perhaps (I guess we'll see) superintelligent AI-systems may be quite good at rewriting code and proving equivelance.

Often there may be parameters/choices where we might unsure of which to pick:

- What "info" should the AI-system be given?
- There are are many parameters/choices where we might unsure of which to pick:
  What "info" should the AI-system be given?
- What parts of previous history (questions given, info about human operators, etc) should AI-system be given?
- How should "scoring-functions" weigh different properties against each other (e.g. "task-performance" vs ease of human verification)
- Over what domain should the system be able to work (give input/output)?

Often what we want is to calculate a data structure that shows how the output will vary depending on what parameters we choose. So we will be able to see how the system [TODO, seems I didn't finish this sentence]

Maybe sometimes (or maybe even often) the AI-system will be able to rewrite code in such a way that an efficient piece of code can simulate running the code with lots of possible parameters. E.g., for the following function...

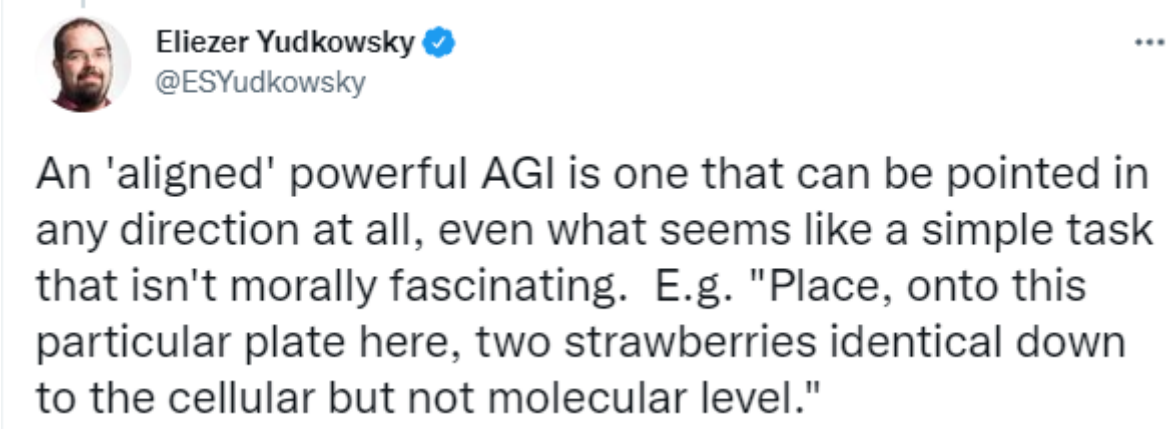
```
(n: number) => {
    if (n < 10) {
        return 10
    }
    else {
        return n
    }
}
```

...I should be able to "prove" that for inputs lower than 10 the result is 10 (I don't have to run the function for all those numbers to confirm that). I leave more complicated examples to future superintelligences.

What is of interest, whether it is obtained by elegance or through "brute force", is a map/data-structure/description showing how output varies based on input. So that we can see clearly how output changes based on e.g. what

info the AI-system is given. And this map/data-structure/description could also be "inversed", displaying for which inputs the output is the same (and by being the same, I mean either being the same exact value, or being within the same "space/area" of values somehow).

Section 8: Mapping concept models (e.g. as in Section 5) to geometric data structures (e.g. physics-simulations) [First draft for this section not finished!]



I do have thoughts about how I would go about solving this problem, and this section describes one of the components/mechanisms/techniques that would be an important part of that strategy.

Let me start with some definitions:

- **Concept model:** Could be something like what I allude to in Section 5, but having something that complex/formal/specified would not be "obligatory". Examples of concepts could include "human", "dead human", "that human over there", etc - but what matters is the *concepts* that these words represent (not the words themselves).
- **Geometric data structure:** More or less anything, but it might often be geomentric and "mathy". The data structure would typically represent something that is physical (e.g. a room with a table, plates, strawberries, etc). But there would be many degrees of freedom in regards to *how* the structure would represent this physical space. A picture with pixels could be one simple example of a data structure. A simulation of a three-dimentional space is another example.
- **Mapping:** A data structure that represents the correspondence between a specific "thing/event/etc" from the concept model and some

part of the geometric data structure. A simple example would be a specification of which pixels of a picture where a specific strawberry "is". However, the data structure would not have to be that simple, and it would not need to assume/imply that there is a strict/precise area of the data structure where the the thing exactly is located (uncertainty, vagueness, etc could be accounted for by the data structure).

- **Mapping function:** A function that identifies mappings between concepts/things and the geometric data structure. A very narrow approach (and straight forward, but computationally infeasible in most cases) would be to have an if-sentence for every possible configuration of pixels (or points in space, or lines in space, or labeled objects in space, or whatever the geometric data structure consists of). An example of a very broad mapping function would be to have a superintelligent agent do the mapping (thus making it so that the mapping function is a superintelligent agent).

It could be the job of a superintelligent agent to provide a "mapping function" (or a "mapping function generator", or a "mapping function generator generator"). As always, efforts should be made to test/verify this mapping function. And understandability is something that would be an important component of the "score" a mapping function is given. Another important component would be brevity (other things being equal, shorter is better). And most of all: How well it corresponds to what makes sense to humans.

Let's imagine there being a geometric data structure that represents a hospital room in a very high fidelity way, and that some of the concepts to be mapped included "sick human", "dead human", "happy human", "healthy human", etc. How would a mapping function do that? Well, there are lots of ways, and I'm glad it's not my job to implement such a function in a manual way. The mapping for "Happy human" would probably

We have already touched upon some principles/techniques that can be helpful when generating simulations in a way that makes it harder for the AI-system to attempt/succeed with any shenanigans. Such as

[First draft for this section isn't really finished]

Section 9: Digital alignment experiments

[TODO]

Real-world actions (and strategic considerations) [will be totally rewritten, and include more/other stuff than currently, including ideas brainstorming for copying strawberry problem]

[TODO]

~~I would think that probably it's best to have a "council" of at least a few AI-systems based on different alignment-methodologies before any action is taken on the internet or in the non-digital realm. Among actions in the non-digital realm, maybe some of the first could involve constructing new types of computers with more processing power - perhaps at first by making modifications to existing chip manufacturing, but maybe later on by developing molecular nanotechnology of the kind that Eric Drexler has theorized about (presuming that this is feasible), and using these capabilities to make computers.~~

~~Some things to keep in mind when choosing ordering of real-world actions:~~

- ~~Machine learning through debate (clustering around / building upon ideas by Paul Cristiano and others)~~
- ~~Trying to crystalize the principles that make human brains aligned, and make use of similar principles but in an idealized form~~
- ~~Guesses (based on various methodologies for guessing performed by "siloed" systems - potentially including high-fidelity simulations in high-welfare conditions) about alignment methodologies that different specific AI researchers and AI researcher teams might have converged on if they had thousands of years to work on the problem (in high-welfare conditions)~~

~~As to first actions after that - well, it will depend on the values and goals of the people who control the AI, and the specific geopolitical situation of the time. But there may be very strong strategic and moral reasons to avoid multipolar~~

~~scenarios, and if scrupulous and large/consolidated teams don't ever make use of their AI, then sooner or later less scrupulous teams will end up doing so.~~

~~One place to start in making real-world strategic use of superintelligent capabilities might be to disable all nukes in the world~~

Desirable outcomes and ethical responsibilities of AI researchers [will be totally rewritten, and include more/other stuff than currently]

[TODO]

~~Unless strong arguments can be made otherwise (by superintelligent aligned AI-systems that are presumed to be aligned), my assumption will be that multipolar scenarios with "distributed" AI will tend to be much more dangerous than some of the better unipolar scenarios. Multipolar scenarios may risk huge waste of resources, much more risk of massively deadly wars (maybe even killing everyone), and huge s-risks with catastrophic amounts of suffering unlike anything seen up until now in the history of Earth.~~

~~What type of "society" that people try to put into being when/if superintelligent AI-systems are developed - well, that wont be up to me - but if it was I would propose some sort of system with checks and balances (involving actual voting, estimations of people's opinions that are much more accurate/nuanced than voting, the outcomes of CEV-reminiscent procedures, etc.~~

~~My vote would be towards~~