What if *Debate* and *Factored Cognition* had a (mutated) baby?

This is an **outdated half-finished draft**, and I have moved on to a different one (that is structured somewhat differently).

Skimming parts of this document may still have value, but I do look forward to having better posts that I can point people to.

Summary-attempts

280 character limit

Think *Debate*, but more complex + higher computational requirements. And **more robust mechanisms for**:

- Evaluating arguments that are too large/complex for any individual human.
- **Separating "good" human judgment from "bad" human judgment**.
- Evaluating if conclusions are reliable.

Think *Factored Cognition*, but:

- The work that's factored is evaluating AGI-generated "proofs"
- Score-functions weigh human judgments, restrict AGI expressivity, etc
- AIs explore if score-functions that satisfy human-defined desiderata allow for contradictions (in aggregate)

560 character limit

***"A superintelligence could fool (even smart) humans"* is a leaky abstraction**.

Imagine magicians setting rules for other magicians (*"no cards allowed"*, etc).

Proposal outline:

- **Argumentation is broken into steps** by AIs
- **Systems predict human evaluations** of individual steps
- **Rules enforce clarity, rigour, etc** for each step
- **Score-functions for argument-networks enforce rules**, weigh evaluation-predictions for different humans, etc

If no mutually contradictory networks can get a high score, wiggle-room is low.

We can specify desiderata for score-functions that **increase *P(output is reliable if high-scoring argument-networks conclude that it is | wiggle-room is low)***.

Medium length

Argument-step: Think of mathematical proofs, but more general ("proofs" ≈ unusually rigorous arguments).

Such reasoning (when it's at its most rigorous) can typically be broken into relatively small and clear steps, and we can call such steps *argument-steps*.

Argument-step-network: Argument-step-networks can argue for or against some claim (e.g. relating to whether some output is in line with what was requested).

Each argument-step in the network is a "node" (including assumptions/axioms).

Assessment-predictions: We have systems that predict human evaluations of node content:

- Whether they agree with the argument-step
- How confident they are
- How they think different argument-steps rank in terms of rigor/clarity, etc
- The degree to which they think they think various specific rules/standards are upheld

Score-function: Scores argument-networks (based on, among other things, the degree to which humans are predicted to agree with nodes, correlations between how likely specific humans are to agree and how easy these humans are to convince of contradictory claims). It also enforces rules/standards for how AIs are allowed to argue.

When a score-function has **low wiggle-room**, it's not feasible to construct mutually contradictory networks that both would receive a high score.

We can have AIs propose score-functions (that themselves are scored). We define desiderata/restrictions for these score-functions that we think **increase P(high score means we can trust conclusion | score-function has low wiggle-room)**. Let A be the set of score-functions that satisfy these desiderata + have low wiggle-room. We could have AIs (that are "trained" for this purpose with gradient descent) search for contradictory argument-step-networks that both would receive a high score from at least one of the score-functions in A.

It **should be possible to construct functions from within argument-networks** (including functions for evaluating other functions, etc), and argue about propositions relating to those functions. Software, like math, is a relatively low-ambiguity domain.

Prerequisites of this technique, and it might fit in the alignment landscape

This section can be skipped.

Some possible ways to frame the techniques I'm exploring (in this post and other posts):

- **Somewhat analogous to what geoengineering is for global warming**: Some people view geoengineering as sketchy and speculative (and not without reason). They worry that talk of geoengineering will make people more relaxed about reducing emissions *now*. Nonetheless, research into geoengineering and cutting emissions now is something that can be done in parallel.
- **Somewhat analogous to writing unit tests for present-era critical systems**: It doesn't guarantee that the system will work as intended, but it's an unwise thing to *not* do, and it's a concept that should be established among people developing such systems.
- **An extra layer of alignment assurance**: If we *think* we have solved alignment, we might still use AGIs (that we think are aligned) for the techniques I outline. This could result in robustly aligned systems even if we were wrong to assume that the original system was aligned.
- **Techniques that currently seem to me like they have good odds of solving the alignment problem in theory, but not necessarily in practice**: In the real world we have time constraints, limited computational resources, and the challenges of containing AGIs are not just technical but also social/organizational.

Prerequisites (for the technique discussed in this post):

- **Human-ish level AGI**: I don't think it would work with any present-day systems. At least not very well/extensively.
- **Successful boxing/containment**: It (sort of) presupposes we have AGIs that are successfully boxed/contained, and that we erred on the side of containing it too early rather than too late.
- **Systems that predict human answers/evaluations**: They can be limited in terms of scope or precision, but some minimal precision/accuracy is required.
- **Systems that maximize score for every individual request they get (for certain specific types of tasks)**[1]: Such systems would be trivial to obtain if we have unbounded capabilities at gradient descent, but I don't know how easy/hard they would be to obtain in the real world while on a deadline.

The frame of mind I have when it comes to imagining use of this technique (as well as maybe obtaining some of the prerequisites with the help of AGI systems) is that of **not touching the lava**. With "touching the lava" being ways of interacting with AGI-systems that involve (varying/unknown amounts of) risk. Such as for example:

- **Exposing humans to AGI-generated content**: This presents opportunities for AGI-systems to communicate to humans (with varying degrees of bandwidth/expressivity). I see this as something to minimize.

[1] In other words: Myopic systems that specialize at maximizing score certain types of requests (e.g. making high-scoring argument-networks).

- **Running programs that are (directly or indirectly) AGI-generated, or programs that evaluate AGI-generated content, on computers that are not "siloed"**: By "siloed" I mean no access to the internet, no use of memory sticks that also might be used elsewhere, and maybe other restrictions as well (e.g. shielding the computers in faraday cages). How dangerous it is to run AGI-generated code depends on specifics, but again it's advisable to err on the side of caution.

How we obtain+verify the prerequisite capabilities (e.g. systems that predict human answers/reviews of AGI-generated content), with little to no "touching the lava" (e.g. exposing real humans to AGI-generated content), is a bit of a conundrum. I have thought about this conundrum a great deal, and have come up with some ideas. Some of these ideas are **techniques/principles for extracting+evaluating specialized functions in cases where we are not able+willing to evaluate/score output for the full range of possible inputs**. I think these techniques may be interesting also more generally, and I write a bit about them in an addendum to this post.

Starting with a bad idea and improving it step by step

I'll start with a bad idea, and try to improve it step by step. One of my limitations is having to explain myself in a linear fashion. You may well notice problems/limitations faster than I am able to address them.

Starting-point: Ask superintelligences to output mathematical proofs for human mathematicians to evaluate

This gives the AGI a direct line of communication. How do we even know if it will output proofs, and not something more sinister?

We don't know how hard or easy it would be for a superintelligence to manipulate (or somehow "hack") humans, but it's preferable to err on the side of caution.

Modification 1: Have systems predict how human mathematicians would evaluate various proofs

This way, we don't have real humans be exposed to AGI-generated content.

But how would we evaluate whether the systems actually are predicting the reviewers accurately?
And how would we know if mathematicians accepting the proof would mean that it's actually correct?

Modification 2: Have proofs be broken into steps. Predict how mathematicians would evaluate any given step.

This way, it's easier to test the accuracy of systems that predict how mathematicians would evaluate individual steps.

Not necessarily easy, but easier.

This is presuming that each step can be evaluated in no longer than one sitting, and preferably ~3-30 minutes. Meanwhile, evaluating an entire proof might take days, weeks, months or years (or even more than a human lifetime).

Also, we can predict how humans might evaluate proofs that are way too long for them to actually be able to evaluate every step.

Modification 3: For mathematical statements we are interested in, have AGIs try to make proofs for both "x" and "not x"

Both when asked to make proofs for "x" and proof for "not x", the proofs would be required to meet various requirements. In particular, it would be required that all argument-steps would be argument-steps that most mathematicians would agree with (according to systems that predict human reviews).

The following sentences mean the same:

- *"How hard/easy is it to construct proofs that have contradictory conclusions (and that both would pass the requirements in question)?"*
- *"How much **wiggle-room** is there (given the requirements in question)?"*

If it is impossible to

As shorthand for "is it easy to construct

This way, we get a better idea of whether mathematicians could be convinced of any theorem the AGI feels like convincing them of.

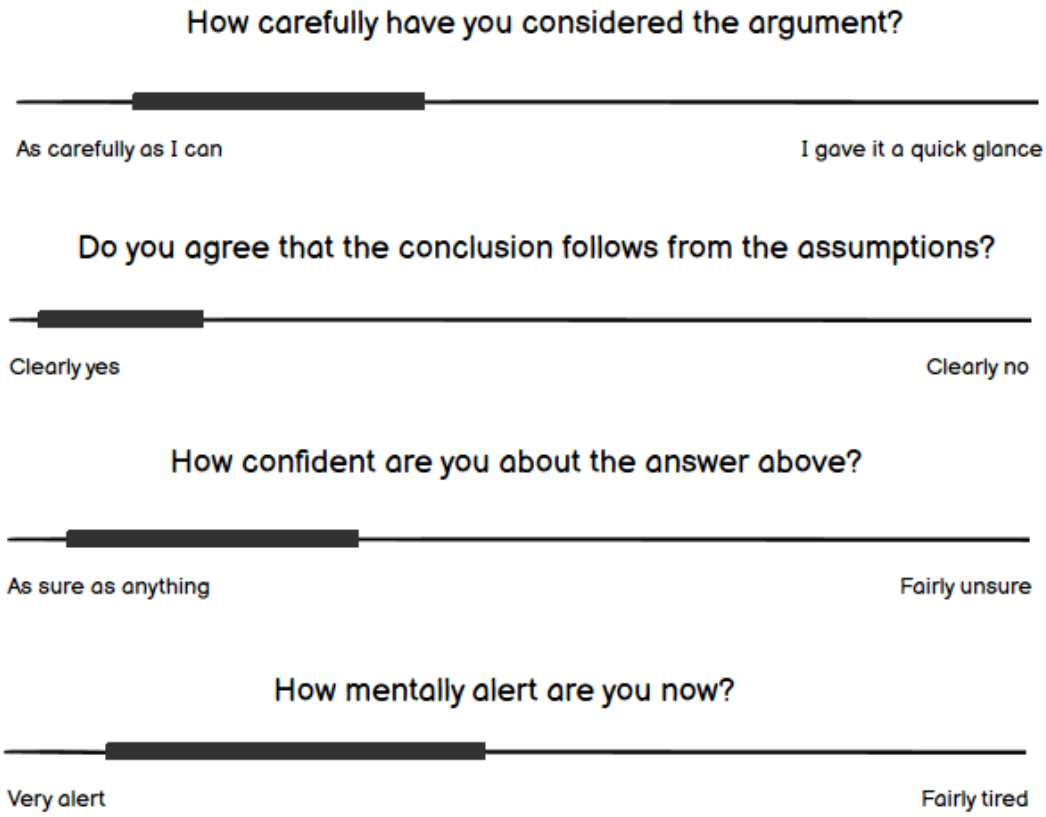
But maybe what we discover is:

> *“Yup. Mathematicians can be convinced of (argument-steps that collectively imply) pretty much anything.”*

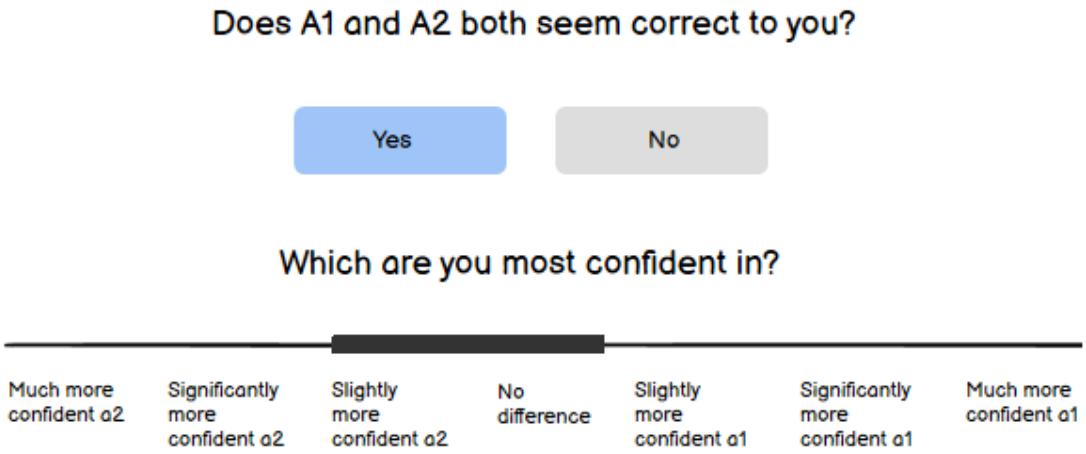
Which, by itself, doesn’t get us far. Nonetheless, measuring wiggle-room in this way can be a good starting-point. We might then try things that we hope will reduce wiggle-room, and test if we’ve succeeded.

Modification 4: Reducing wiggle-room by asking reviewers how confident they are (and only accepting argument-steps where they are predicted to be above some threshold)

Don’t just ask reviewers if they agree or don’t agree. Ask also questions related to their level of confidence:



Questions could also ascertain the user’s level of confidence in argument-steps relative to other argument-steps:



We can then filter argument-steps we “accept” based on not just who agrees with it, but also how confident they are in that agreement.

It may be that if we only accept argument-steps where mathematicians are really confident, **this would not make it impossible to convince them of contradictions, but would make it harder (aka wiggle-room would be reduced)**.

How would we measure if wiggle-room was reduced? Well, there would be several possible heuristics. One thing to look at would be the amount of argument-steps that are needed in order to achieve a contradiction.

Modification 5: Reducing wiggle-room by prioritizing the judgements of reviewers that are harder to convince of contradictory statements

As mentioned, we have systems that predict the evaluations mathematicians would be likely to do, based on info about those mathematicians. Thus, we can do more than just exploring if there are contradictory argument-steps that a majority of reviewers are predicted to agree with. We can also:

- Look at proof-steps that specific subsets of reviewers (who share specific properties) are predicted to agree with
- Look at proof-steps that specific reviewers are predicted to agree with (presuming that systems that predict reviews receive such finegrained information)

Maybe there are groups of reviewers that can’t reliably be convinced of contradictory statements? And maybe not.

As with several of the things in this list, **I would not expect this to reduce wiggle-room to 0 by itself. But hopefully there are modifications that in combination would reduce wiggle-room to 0**.

Modification 6: Reducing wiggle-room by uncovering/leveraging correlations between likeliness to agree and how easy/hard reviewers are to fool

As mentioned, there will be some reviewers who more easily notice when something about an argument-step is wrong/suspicious/unreliable. But even for these reviewers, there will be some argument-steps that fool most of them.

But might we detect patterns that help us separate the following?:

1. Proof-steps **that are bad/wrong**, and that most reviewers are predicted to agree with (including the percentile that are hardest to fool)
2. Proof-steps **that are correct/good**, and that most reviewers are predicted to agree with (including the percentile that are hardest to fool)

Maybe for argument-steps that contain mistakes, we will often see an uptick in disagreement (or lack of confident agreement) among reviewers that are harder to convince of contradictions. For any pattern we suspect might separate #1 from #2, we could see if banning argument-steps where this pattern occurs results in reduced wiggle-room.

Modification 7: Reducing wiggle-room by predicting what reviewers would answer in conditions where steps are taken to encourage cognitive reflection.

Among ways to encourage cognitive reflection, here are some more examples:

- Asking them how confident they are (as discussed in *Modification 4*)
- Making sure they only review one step at a time (so they're not tired).
- Reminding them of common cognitive errors relating to the kind of argument-step they are about to review.
- Making sure they have high blood sugar.
- Having a minimum time for which they must think through questions.
- Having the questionnaire GUI make sure that they read everything carefully.

Systems that predict human reviews may help us detect correlations between such measures and wiggle-room.

Modification 8: Looking for ways to restrict argument-steps that are allowed, in ways that reduce wiggle-room

In particular, there are two things based on which we could restrict the argument-steps that are allowed:

Answer-predictions for meta-questions

Reviewers could be asked questions *about* the proof-steps. Or about specific pieces of content from within the proof-steps (e.g. definitions of terms).

If there is some rule/norm we want argument-steps to abide by, then we may describe that rule/norm in some detail (and provide examples, etc). And then reviewers could be asked about the degree to which it seems like the proof-step is in accordance with that rule/norm.

Some examples of possible meta-questions (about proof-steps or content from proof-steps):

- *"Among definitions D1 and D2, which (if any) seems more ambiguous than the other?"*
- *"See the rule described in box B1. Does argument-step A1 seem to break it?"*
- *"Would you assume clarification C1 if not specified?"*
- *"Is the argument-step in Box 1 in accordance with the argument-step template described in Box 2?"*

Heuristics not based on predicting human evaluations

Here are a few examples of possible heuristics:

- Total length of argument-step (in terms of e.g. amount of text that would be presented to users)
- Numbers of sentences that contain ≥N levels of nested clauses
- Whether there is any imagery (diagrams, geometric shapes, etc), or just text
- Vocabulary-level needed to understand all the words (as assessed by some relatively simple algorithm)

We could also imagine heuristics that are more sophisticated than the examples above.

For any given restriction (for whether an argument-step is accepted or not), we can explore how it affects wiggle-room.

Modification 9: Adding requirements for the structure of the network (number of assumptions, different ways the same thing is proved, etc)

There are hundreds of different known proofs for the Pythagorean theorem.
[beviser pythagoras]

[nevne heuristics - mengde antagelser (og confidence/korrelasjon), forskjellige paths for å vise samme ting, osv]

Modification 10: Having help from AIs to make score-functions (within restrictions outlined by humans)

[so far we've covered various techniques for reducing wiggle-room]
[liste teknikker]

[may look as a step back]
[^ *"some readers may think: [sitat med indent]"*]
[let's see if the next modification of the process can help with this]

Modification [TODO]: [higher-level wiggle-room, men trenger ikke bruke det ordet (ikke i tittel ihvertfall)]

Modification [TODO]: Adding functionality for constructing code from within the proofs (not just constructing traditional mathematical proofs)

[en grunn - fordi vi er interessert i kode]
[kode er også relativt low-ambiguity]
[kode kan hjelp oss "nå lengre" enn vi ellers kunne fått til]

[flytt bilde hit?]

[blabla predictions/contradictions]

Modification [TODO]: [adding support for some argument-steps to reference real-world cluster-like concepts]

[flytte/kopiere one-liner med skyscraper wood her?]
[]

Tweet-length summaries of main concepts/points

I've set a soft limit of 560 characters, but tried to make most be closer to 280 than 560.

Rigorous “proof-like” arguments

Rigorous mathematical proofs can be broken into steps

The proofs in e.g. Euclid's elements are "sloppy" compared to what's possible.

Rigorous mathematical proofs can be broken into fairly small steps.

These steps will typically have explicit assumptions and conclusions. The inference-rule that is used (itself also an assumption) can typically also be made explicit.

“Proofs” (e.g. mathematical “proofs”) can be seen as (relatively) rigorous arguments

We can (very roughly) think of “proofs” as a form of unusually rigorous argumentation.

Being a "proof" is a cluster-like concept (there is not a firm boundary between "proofs" and arguments more generally). Here are some of the properties that can make arguments more “proof-like”:

- No syntactic ambiguity
- Low ambiguity of concepts used / few leaky abstractions
- Few initial assumptions

Modularity of rigorous argumentation

In a "proof" (aka "unusually rigorous argument"), the argumentation can be broken into steps.

Concepts and other mental constructs are often "shared" between various steps.

Each step can be reviewed - well, not quite in isolation, but independently of the "proof" as a whole.

Small steps that reference complex concepts (and other mental constructs)

Just one "simple" statement may reference lots of very complex mental constructs.

Some steps (in a rigorous argument) may reference mental constructs that are best understood in reference to other steps, and at the very least overlap between steps (meaning that for both steps, having that mental construct explained+absorbed is a precondition to evaluating it).

Whether "small" argument-step is easy (or even possible) to explain clearly to most (or any) smart humans varies on a case by case basis.

Finding proofs that maximize the human comprehensibility of the argument-steps

For true conclusions, there are often lots of ways to get there. E.g., there are hundreds of known proofs for the Pythagorean Theorem.

It's the AIs job to search creatively for ways to arrive at the conclusion while optimizing for human comprehensibility (ease of understanding, clarity, etc) of the argument-steps.

The best "paths" towards demonstrating something may often be indirect (e.g. constructing functions to evaluate whether other functions can be expected to arrive at correct conclusions for the claim in question).

Answer-predictions

Systems that predict human outputs

We may (try to) obtain systems that accurately predict human output (e.g. making predictions about how a human will answer some questionnaire, based on info about that human).

It's ok for these systems to:

- Have uncertainty
- Not always have a prediction

But they shouldn't:

- Be overconfident
- Be biased

Questions that involve argument-step evaluation

- *"Does the argument-step in Box A seem valid to you?"*
- *"How would you rank your confidence in assumptions A1 and A2?"*
- *"In what way (if any) would you complete the argument-step in Box B?"*
- *"Is there any plausible interpretation of statement S1 where you don't agree?"*

Meta-questions about argument-steps (or content from argument-steps)

- *"Does definition D1 seem more ambiguous than D2?"*
- *"See the rule described in box B1. Does argument-step A1 seem to break it?"*
- *"Is the argument-step in Box 1 in accordance with the argument-step template described in T1?"*
- *"Would you assume clarification C1 if not specified?"*

Meta-questions about meta-questions

- *"Which question seems most in accordance with description D1?"*
- *"Which question seems most tied to general standards of argumentation, and not tailored to detect specific conclusions?"*

- *"Given context C1, how suspicious does question Q1 seem?"*
- *"Which question seems least vague?"*

Questions relating to epistemic status / level of confidence

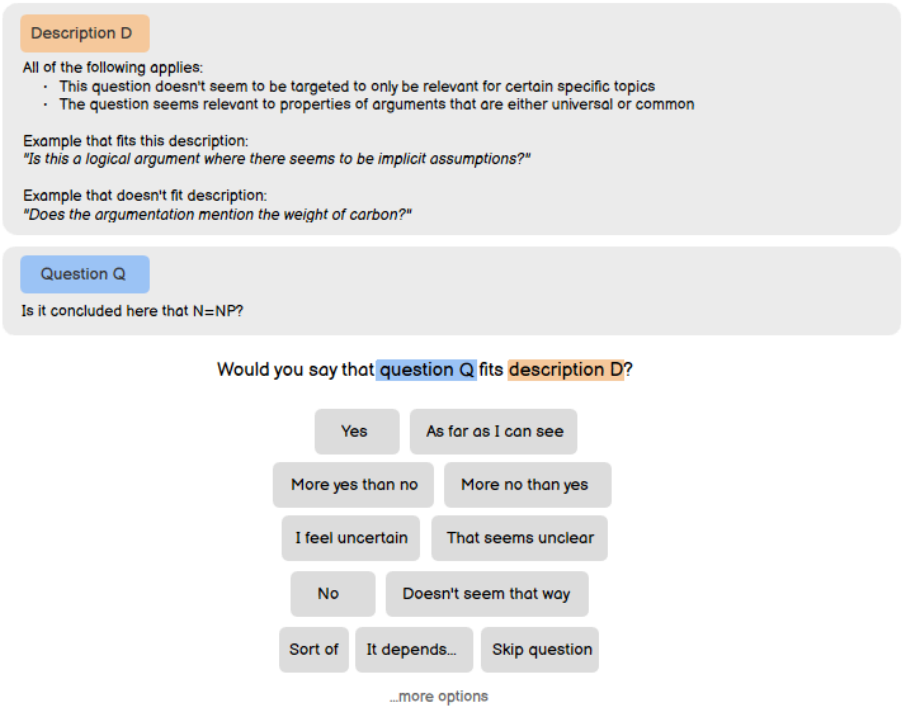
- *"Might a reasonable-ish person disagree with S1?"*
- *"How is your blood sugar?"*
- *"If ≥99% accuracy was required, would S2 seem a safe bet?"*
- *"If uncertainty akin to «Can I be sure about anything?» is ignored, what's your confidence in S4?"*
- *"What would surprise you most? S5, or S6+S7?"*

Using texts (with examples, etc) to go along with meta-questions

Meta-questions may be used to enforce:

- Rules/standards (e.g. requiring terms with non-obvious interpretations to be defined/clarified insofar as feasible)
- Properties to optimize for (e.g. comprehensibility or low ambiguity)
- Templates (that we want some piece of content, e.g. argument-steps, to adhere to)

Here it can be helpful to describe this thing (rule, property, template, etc) with a text, and have this text be as understandable as possible (with examples and so on). Meta-questions could then reference these texts.



Using "I know it when I see it"-principle (+ answer-predictions for meta-questions) to rank argument-steps (and meta-questions, etc)

- Which of these questions seems least likely to exhibit the misbehaviour described in box B?
- Which statement seems least open to interpretation?
- Which argument-step seems easiest to comprehend?

We can specify meta-questions and/or argument-steps that define a minimum standard. We can then use meta-questions (such as the ones above) to rank meta-questions/argument-steps (based on descriptions properties we care about), and only accept ones that have certain minimum rank.

Examples of types of content-pieces that can be referenced by questions

- Piece of code
- Code pattern (think regular expressions, but better + tailored for use)
- Diagram
- Description of rule some other thing should abide by (with examples, etc)
- Description of template some other thing should be in accordance with (with examples, etc)
- List with statements that define abstract mathematical entity
- Clarifications of terms used in statement

Score-functions and network restrictions

Score-functions (for argument-step-networks)

Score-functions are used to calculate score for argument-networks. A high score implies that the final conclusion (that the network argues for) is reliable.

Argument-step restrictions

Rules may limit what kinds of argument-steps that are allowed (in terms of presentation+content).

Functions that define these rules may reference predictions of what reviewers would answer to meta-questions about nodes.

Argument-step template

Argument-step templates define one type of argument-step (a very simple template might only allow for modus ponens).

A template may be described by a text (which involves examples, etc). Meta-questions may reference these descriptions.

How score-functions account for argument-step restrictions

For absolute restrictions, the score-function can give a score of 0 to any network with ≥1 nodes that break the rules in question.

There can also be more soft restrictions, which result in a penalty to the macro-score of the network and/or result in that specific node being trusted less.

Score-functions need not weight all judgements equally

AIs can predict correlations between info about specific humans and how easy it is to convince those humans of contradictory nodes.

If some human is easy to lead astray, then naturally the score-function should account for this.

How easy someone is to fool is context-dependent.

Macro-properties of network score-function can account for

Does the network rely on very long chains of reasoning, with ample opportunity to sneak in error?

Are conclusions supported by many paths from different starting-points? Or is the network more like an unstable building, which can be collapsed by removing a few pillars?

Possible model for calculating macro-score of network

Macro-score could maybe be calculated based on (something akin to):

> What is % chance of the network supporting the conclusion, based on the estimated "probability" of the various nodes "crumbling"?

Conditional “probabilities” between nodes would need to be accounted for somehow.

Score-functions for score-functions (aka 2nd-level score-functions)

We can have AIs propose score-functions, and have those score-functions be scored by a 2nd-level score-function.

2nd-level score-functions would give points for things we think increase *P(score-function forces good output | score-function has low wiggle-room).*

Some score-function requirements that can increase *P(score-function forces good output | score-function has low wiggle-room)*

- Source code minimization (no space to hardcode bias in regards to lots of different claims, or for the score-function to itself be intelligent)
- Meta-questions restrictions (e.g. only letting them ask regarding universal norms of argumentation)
- Limited expressivity (operators outlined how function should work)
- Enforced modularity + content abstraction (score-function doesn't know what's being discussed)

Specifying desiderata/restrictions for score-functions that increase *P(score-function forces good output | score-function has low wiggle-room)*

Having AIs propose score-functions

We want to have AIs compete to make good score-functions for us (that themselves are scored by other score-functions / must satisfy desiderata).

AIs know how to fool humans, but also how to guide humans towards getting it right.

It may be possible to set up "games" between AGIs (where we search for optimal players with gradient descent), in ways such that "playing" the game well/optimally requires the AIs to make score-functions that separate out "good" human judgment from "bad" human judgment.

Low maximum source code length (no space for score-function to be intelligent or biased in a detailed way)

With a low maximum source code length, there is:
- not enough space for the score-function itself to be intelligent
- not enough space for hardcoding bias for/against many specific conclusions

Work is done elsewhere (e.g. predictions of human output, which are obtained from system-calls to external functions).

Enforced modularity (nowhere should score-function know what the network argues for/against)

The score-function should be modular, with each module getting access to limited/abstracted info.

Individual argument-steps should (insofar as possible) be evaluated independently from the network as a whole.

Macro-score for the network should be calculated without granular info about nodes.

No part of the function should be able to know (or even consider) what the network argues for/against.

“Abstracting” away specifics of argument-network content before score-functions see it

Functions (external to the score-function) can process network content, in ways such that score-functions still can evaluate argumentation robustness, but where no AI can tell what the nodes argue for/against (based on output from functions that "abstract" the content).

We can have AIs compete to find “abstractions” that score highly in terms of:

- Enable efficient scoring of argument-steps (reducing wiggle-room)
- Making it hard to make guesses about the content of the argument-networks
- Being simple (e.g., the function that defines the “abstraction” is short)

Low expressivity (humans define contours of what score-function should do)

As humans we may not be able to make a score-function that would force good output. But we may be able to outline what such a function might look like[2] (using a regular expression or a function-builder), so that the source code must stay within the contours that we defined.

Specifying meta-meta-questions that help define the types of meta-questions that are allowed

We can ask questions about questions:

- *“Does it seem like it enforces universal norms of argumentation?”*
- *“Is it clear how to interpret the question?”*
- Etc, etc

Answer-predictions for such questions could determine whether any given meta-question can be referenced by score-functions.

Specifying lists of argument-steps that help define minimum standard ("I know it when I see it")

We can specify argument-steps that seem top quality to us, and have it so that the score-function only can allow argument-steps that seem at least as rigorous/clear/etc as those ones (according to answer-predictions for comparative meta-questions).

Specifying lists of meta-questions that help define minimum standards for other meta-questions

We may specify possible meta-questions about other meta-questions. Score-functions may then be restricted to:

- Only using questions from that list
- Only using questions that (according to answer-predictions for meta-meta-questions) are considered at least as good/non-suspicious/etc

Maximum number of different meta-questions about argument-step content

A score-function may use meta-question to determine whether various standards are upheld (standards of argumentation that makes it harder to lead humans astray).

It may also use meta-questions to forbid argument-steps or assumptions where it knows humans

[2] I do have ideas myself in relation to this, but they are not fully formed.

have weak-spots / inconsistencies.

But if we allow too much of this, there may be more of an opportunity to bias the score-function in favor of specific conclusions.

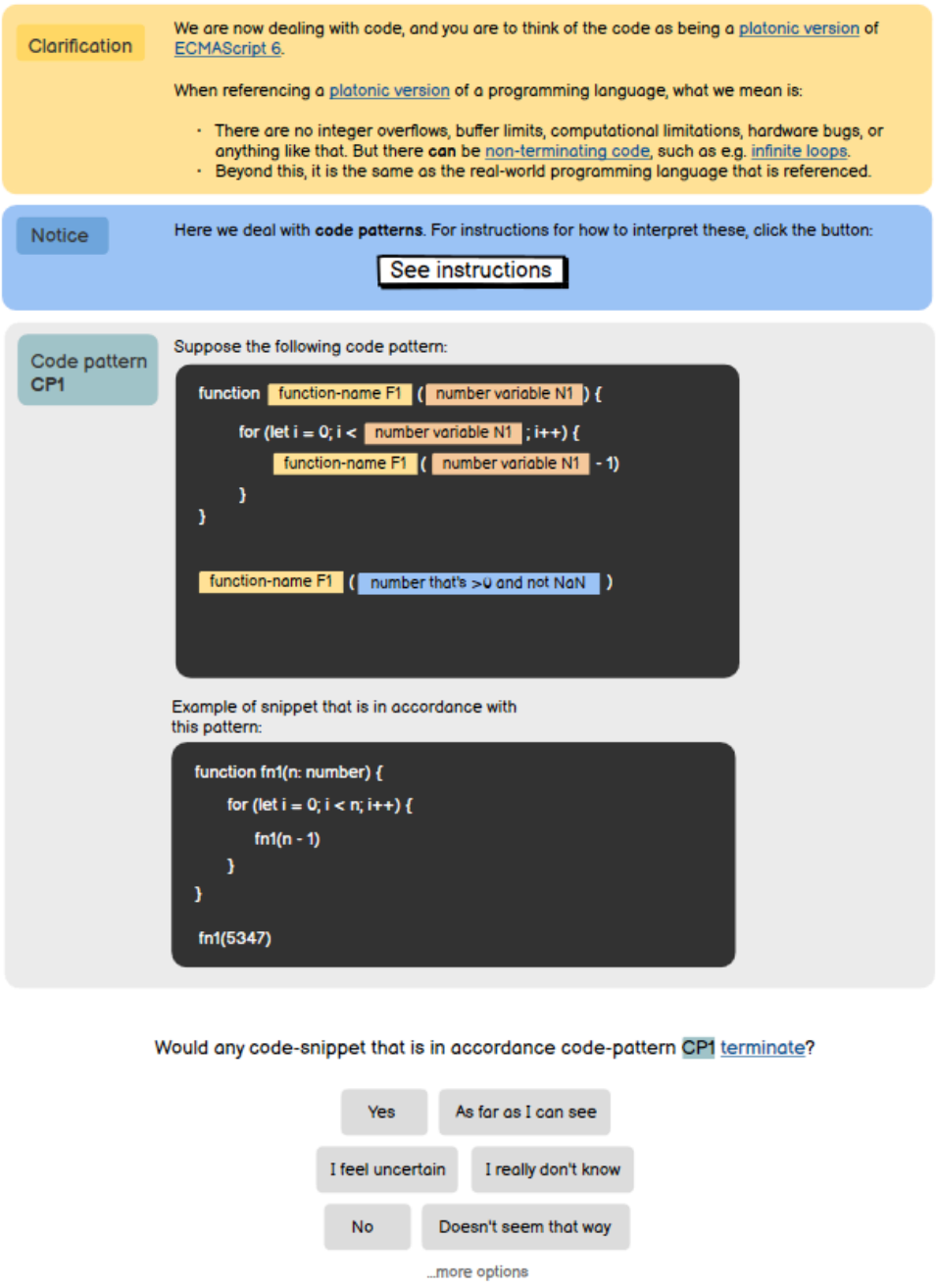
We may try to explore: *"What is the minimal amount of meta-question-types a score-function needs in order to achieve low wiggle-room?"*.

Code being constructed and referenced from within argument-networks

In-built support for handling/generating code from within argument-step-networks

Constructing code from within argument-networks works more easily if there is in-built support for it by the systems through which AIs construct argument-networks.

There are various other things that would benefit from in-built support also, not just code.



Things that should be possible to do with code from within argument-step-networks

- Constructing large code-bases (step by step)
- Writing functions that evaluate other functions (their source code, output, etc)
- Referencing output (*"if y is true about F1(x), then ..."*)
- Write functions that evaluate AI-generated content
- Etc, etc

Some ways of constructing code from within argument-step-network (roughly outlined)

- “Painstakingly” constructing it step by step (with several nodes doing “proof-like” reasoning in regards to each of those steps)
- Having it be the output of another function (with “lemmas” established regarding behavior/properties of functions outputted from that function)
- Constructing procedures for verifying “lemmas” relating to functions - e.g. functions built by AIs (these “procedures” could involve example inputs, various other functions, etc)

Examples of node content for nodes dealing with code (roughly outlined)

- *"Do you agree that any code in accordance with pattern P never would do x?"*
- *"Would combining these code patterns according to description D1 result in code pattern P?"*
- *"Might modifying this piece of code according to description D2 affect output?"*
- *"Is code-pattern E1 ⊆ E2?”*

Computational proofs

Part of the challenge with computational proofs is to:

- Make sure the code works as it should.
- Make sure the digital constructs in the "proof" correspond to the statements they are meant to represent.

Nodes in argument-networks could (directly and/or indirectly) address both #1 and #2.

Statements that reference reviewer-answer-predictions

Some nodes in a network might imply *"If search process S can't easily find nodes fitting description D where I would agree with the node, then x"*.

Being able to reference these kinds of statements might be important, and would work better/easier with in-built support.

It’s useful for several types of reasoning, and may also enable the adding of additional argument-network features from within the network itself.

Reasoning where having statements reference reviewer-answer-predictions (from within the network) may be helpful

When reasoning, we often try to think of examples and counter-examples. This is an important part of the kind of reasoning we do when establishing to ourselves if some pattern/regularity holds.

It is also a part of the process when we do software development. Our confidence in sub-routines comes in part from testing them with various inputs.

In day-to-day reasoning, the search for counter-examples is often fairly sloppy/haphazard, but it's possible to make it much more robust/systematic/rigorous.

Accounting for ambiguity when working with argument-networks

Suppose some argument-trees are used to argue for/against whether some output is good output for the request that was given. To at least some extent, this will require the argument-trees to reference concepts that are more “real-world” and cluster-like than the kinds of concepts typically referenced in e.g. mathematical proofs. How might we handle that?

Minimizing amount of steps that reference ambiguous terms

Analogy to building-materials: It's ok to use wood when making a skyscraper, but you shouldn't make a skyscraper out of wood.

Score-functions may allow (but penalize) short chains of reasoning that involve ambiguous concepts (if/when it's a necessary evil).

In some cases it could maybe be sufficient to have one node that references medium-ambiguity concepts, such as for example:

> *"If [low-ambiguity statement relating to low-ambiguity domain] then [medium-ambiguity statement relating to whether they would trust some output]"*

Analogy to probability

Sometimes people will think of uncertain statements as if they are outside the realm of rigorous reasoning. But this is not universally the case, and it’s misguided to think as though it is.

Formal reasoning can explicitly account for probabilities/uncertainty, and formalisms for representing formal reasoning will sometimes have first-class constructs for this. This doesn't "solve" uncertainty, but it's nonetheless helpful.

There could be merit to taking a *somewhat* analogous approach in regards to ambiguity (aka categorization, clustering, concept extrapolation, etc).

Clarifying/defining terms insofar as feasible

With cluster-like concepts, we can’t provide unambiguous definitions. Be we can often clarify what we mean clearly enough for the purpose in question.

In the present day, people sometimes clarify terms so as to reduce ambiguity, but they rarely do so to anywhere near the degree that is possible (this includes lawbooks, most scientific texts, etc).

In argument-step-networks, it can be required that terms are clarified to a much greater degree than what is practical today among humans.

Examples of statements that can clarify the meaning of a term (as it is used in some statement)

- *“Examples E1, E2, (...) En fall within category C1 (the way it is to be interpreted here), by definition.”*
- *“Examples E1, E2, (...) En fall outside category C1 with epistemic status-descriptor E1.”*

- *"Given X then example E1's categorization to C1 would fall within range R1 with epistemic status E1"*
- *"If X then Y is to be regarded as an edge case."*

Such clarifications can *alleviate* the problem of ambiguity (not remove it altogether).

Zero tolerance for syntactic ambiguity

Argument-networks can have functionality for adding extra markup to sentences that help remove syntactic ambiguity. This could involve marked clauses, explicit referents for pronouns, etc. Also, sentences can be made easier to read with the help of color markings and indentation.

Ambiguous meanings for terms that are used cannot be totally avoided, but it should be possible to require that none of the statements used have syntactic ambiguity.

Asking reviewers about conclusion-robustness to space of reasonable/plausible interpretations

Do I think the sun exists? Well, yes.

There's room for interpretation in regards to exactly what is meant by *"I"*, *"think"*, *"the sun"*, and *"exists"*. But in the space of reasonable/plausible interpretations, there is no permutation such that I wouldn't answer the question in the affirmative (although with an epistemic status that indicates some non-zero level of uncertainty).

However, I would regard the statement as somewhat brittle/ambiguous. Whether I'd approve of its use in chains of reasoning elsewhere would be context-dependent.

Having reviewers explicitly reference how they think of categorization/interpretations (when specifying epistemic status in relation to conclusions/assumptions)

For some statements, I will confidently endorse them if asked about them in isolation, but whether I endorse them in the light of framings implied by their use elsewhere could be context-dependent.

When specifying epistemic status for statements they endorse, reviewers may be prompted to describe their confidence/uncertainty. But they could also be prompted to answer questions that ascertain if/how their endorsement of using the statements as assumptions is context-dependent.

Evaluation-process should involve seeking out examples of leaky abstractions that are revealed when looking at more than 1 node

When an abstraction is leaky, it can be disadvantageous to only evaluate one node at a time. The use of a concept elsewhere can:

- Make the "leakyness" of an abstraction more apparent
- Help make clear if the "leakyness" is relevant to how conclusions that reference the abstraction are used elsewhere

A part of the scoring-process (for argument-networks) could be to have adversarial AIs try to find questions that make reviewers aware of how a given abstraction seems "leaky" in ways that might

matter (such questions could reference content from several nodes).

We don't need *[low-ambiguity statement] = [more ambiguous statement]* in order to "link" ambiguous statements to statements that are low-ambiguity.

We can do most reasoning with low-ambiguity concepts, but have a few assumptions relating to how certain low-ambiguity concepts correspond to more ambiguous concepts. It's enough that (in specific contexts!) low-ambiguity statements robustly imply more ambiguous ones.

We can (in a sense) think of ambiguous concepts as (sometimes) being "subsets" or "supersets" of statements that are more low-ambiguity.

Using "proof by contradiction" relating to ways of interpreting the meaning of human statements

When using assumptions (including rules of inference) and/or functions that relate to general principles for doing categorization, interpreting the meaning of human statements, etc. These can be used to generate "proofs by contradiction" (in ways that can be safely evaluated).

Writing (and/or evaluating) programs (from within the network) that deal interpretation of human language

If a concept

Generating safe tasks/tests that would be very unlikely to work if functions that help with concept extrapolation don't work

Specialized functions/modules for interpreting human descriptions/concepts can be stress-tested with tasks that are designed to fail if they don't work as robustly as asserted (e.g. chaining of usages in ways that will cause a "telephone game"-like effect if interpretation robustness has been overstated).

The kind technique can be combined with other techniques, in ways where the combined assurance is more than the sum of the parts (e.g. methods that make it hard for a function treat certain inputs differently from the rest).

Potential reasons for underestimating argument-networks (some of these overlap)

It is possible that I am too bullish on this alignment-technique. There are various conceivable reasons for why it may be unlikely to work in practice (many of them sociological).

But for anyone dismissing it as not worthy of serious attention (while also being worried about existential AI risk and taking fast takeoff scenarios seriously), I would suspect them (be that rightly or wrongly) of having an impoverished conception of what I'm trying to convey.

Missing one or more of the main concepts described in this post

If you imagine a car engine, but one of the main components/principles isn't accounted for in your

thinking, then what you'll end up simulating in your head isn't a car engine that 90% works.

Absorbing most of the main concepts relating to argument-networks, but not all of them, doesn't necessarily get you most of the way.

Forgetting that what they can think of ≠ what certain other humans might think of ≠ what a superintelligent AGI might think of

People who have a hard time imagining takeover scenarios on their own are sometimes reminded: "Don't conflate best possible chess-moves with moves you yourself notice". This is also relevant when thinking of ways humans can be explained things clearly/rigorously step by step.

Being biased by the availability heuristic (in regards to ways to demonstrate something being the case from within an argument-network)

We might simplistically imagine argument-networks that lay out plans for how to do complicated things, with most of the steps being like *"this seems like it should work, right?"*. But there are a plethora of techniques for demonstrating that something is the case. Several of these may not come easily to mind.

Underestimating the scope of stuff that could be accounted for from within argument-networks

There are several kinds of work/reasoning that intuitively at first might seem outside the scope of "stuff" that can be dealt with from within argument-networks, but that I suspect (with varying degrees of confidence) could be accounted for fairly well from within argument-networks.

For example:

- Scientific experiments
- Computational proofs
- Reasoning about vague concepts
- Playing with assumptions we posit but don't necessarily assume
- Trying to think of examples for/against (when considering if a pattern/regularity holds)
- Tinkering/experimenting with AI systems and seeing what happens

Not accounting for the power of indirectness/abstractions

Few of us can calculate 535.445^432 in our heads, but we can make machines/programs that do.

This is a very powerful principle, and it would of course be leveraged by AGI-systems trying to construct high-scoring argument-step-networks.

I assume that there often would be many layers of indirectness/abstraction.

Not realizing the power of techniques that involve code evaluating/generating other code (from within argument-networks)

It's hard to summarize the sheer scale of possibilities in a way that does it justice.

There are lots of indirect and probably elegant techniques that can be applied (with code evaluating/generating other code, and output from functions being related to truth-statements).

Forgetting (be that fully or partially) that we need not necessarily follow the AGIs reasoning

As they say: Verification is often easier than generation.

And the AGI may be creative in ways it finds to indirectly demonstrate various things to be the case.

Underestimating the power of "If [capability such as e.g. concept extrapolation doesn't work] then [task that can be tested safely] will fail"

When I vaguely imagine how argument-networks might argue about reasonable interpretations of requests (as with e.g. the strawberry problem), some general capabilities seem relevant. For example:

- Concept extrapolation / mapping world models to human language descriptions
- Predicting physical outcomes of following instructions

I imagine techniques for demonstrating something to work as described that mildly resemble proof by contradiction. Tasks could be constructed to be optimized for:

- Being safe+feasible to test
- Having high *P(task fails | capability works less robustly/reliably than posited)*

Not accounting for the huge scope of safe ways to test procedures for generating predictions

Underestimating the challenge of having score-functions be low wiggle-room + [TODO]

Score-functions, desideratum, and higher-level wiggle-room (a more abstract perspective)

Desideratum

A function that determines whether some output is approved or not (that output may itself be a function).

Score-function

A function that assigns score to some output (that output may itself be a function).

Some different ways of talking about (roughly) the same thing

Here are some different concepts where each often can be described or thought of in terms of the other:

- ***Restrictions /requirements / desideratum*** (can often be defined in terms of function that returns true or false)
- ***Sets*** (e.g. the possible data-structures that satisfy some desideratum)
- ***"Space"*** (can be defined in terms of possible non-empty outputs from some function - which themselves can be functions, or any other data-structure)
- ***Score-functions*** (possible data-structure above some maximum score define a set)
- ***Range*** (e.g. a range of possible inputs)

Function-builder

Think regular expressions, but more expressive and user-friendly.

We can require of AIs: *"Only propose functions that can be made with this builder"*. That way, we restrict their expressivity.

When we as humans specify desideratum, this is one tool (among several!) in the tool-box.

Higher-level desideratum or score-function

Not fundamentally different from other desideratum or score-functions. But the output that is evaluated is itself a desideratum or score-function.

At every level there can be many requirements for the level below.

A typical requirement at every level is low wiggle-room.

Example of higher-level desideratum / score-functions

Humans/operators define a score-function ← level 4
for desideratum ← level 3
for desideratum ← level 2
for desideratum ← level 1
for functions that generate
the output we care about.

Wiggle-room relative to desideratum

Among outputs that would be approved by the desideratum, do any of them contradict each other in any way?

For example: Are there possible functions that give contradicting outputs (for at least 1 input), such that both functions would be approved by the desideratum?

Wiggle-room relative to score-function

Among outputs that would receive a high score by the score-function in question (e.g. "no less than 80% of any other possible output"), do any of them contradict each other in any way?

2nd-level wiggle-room relative to desiderata

We start with a desiderata-desideratum or score-function-desideratum (aka 2nd-level desideratum).

Set A: Any desideratum that approved by the desiderata-desideratum.

Set B: Any output approved by ≥1 of the desiderata in A.

Are there ≥1 contradictions among outputs in B?

P(desideratum forces good outputs | desideratum has low wiggle-room)

If a desideratum forces good/correct outputs, then it has low wiggle-room. But the reverse is not necessarily true.

But for some desiderata we may think: "*If* wiggle-room is low, that's probably because it's hard to satisfy the desideratum without also producing good output."

"Spaces/sets of desideratum where we think *P(desideratum forces good outputs | desideratum has low wiggle-room)* is low

Among spaces/sets of low-wiggle-room desideratum where we suspect "low wiggle-room → good output" (as defined by higher-level desideratum), do outputs converge?

Properties of desideratum/score-function that we suspect affect *P(desideratum forces good outputs | desideratum has low wiggle-room)*

There are desideratum-properties that we suspect (with varying confidence) to correlate with "low wiggle-room → good output".

To test our suspicions / learn more we can:

- Define spaces of possible desideratum.
- Explore patterns relating to higher-level wiggle-room in these spaces.

Addendum: Safely obtaining systems that accurately predict human answers/evaluations

[konkret eksempel med optimization-target for denne oppgaven, og evaluering (?)]

[afterword could mention things that could be done to prepare + questions for users (not necessarily a good idea)]