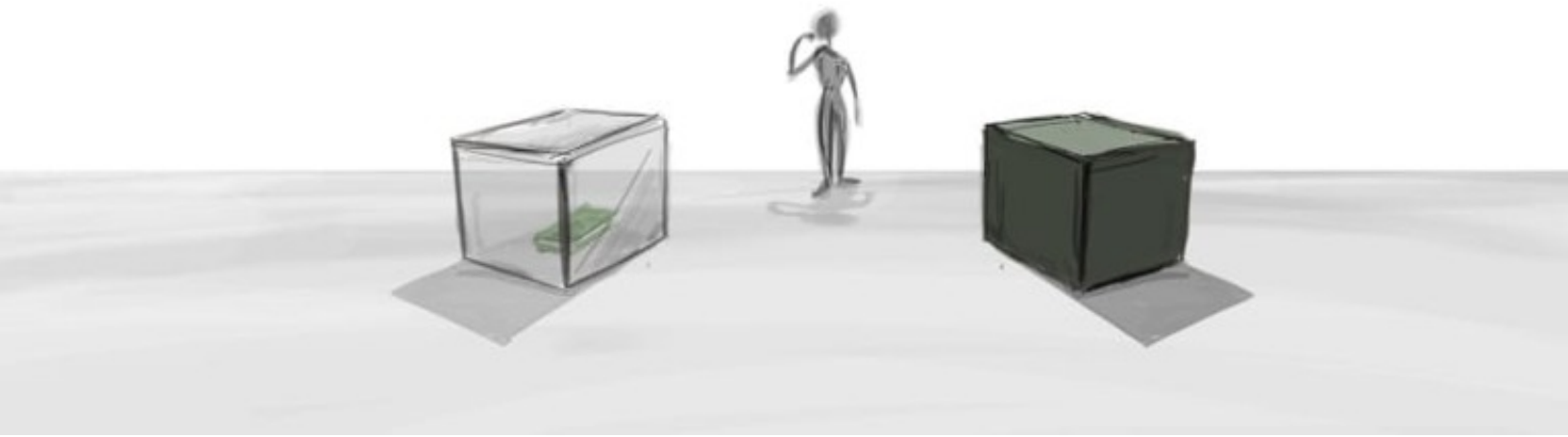


Decision Theory: Newcomb's Problem

1. Decision theory: An outline of some upcoming posts
2. Confusion about Newcomb is confusion about counterfactuals
3. Decision theory: Why we need to reduce “could”, “would”, “should”
4. Decision theory: Why Pearl helps reduce “could” and “would”, but still leaves us with at least three alternatives

Decision theory: An outline of some upcoming posts

Last August or so, Eliezer asked Steve Rayhawk and myself to attempt to solve Newcomb's problem together. This project served a couple of purposes:
a. Get an indication as to our FAI research abilities.
b. Train our reduction-muscles.
c. Check whether Eliezer's (unseen by us) timeless decision theory is a point that outside folks tend to arrive at independently (at least if starting from the rather substantial clues on OB/LW), and whether anything interestingly new came out of an independent attempt.

Steve and I (and, briefly but helpfully, Liron Shapira) took our swing at Newcomb. We wrote a great mass of notes that have been sitting on our hard drives, but hadn't stitched them together into a single document. I'd like to attempt a Less Wrong sequence on that subject now. Most of this content is stuff that Eliezer, Nesov, and/or Dai developed independently and have been referring to in their posts, but I'll try to present it more fully and clearly. I learned a bit of this from Eliezer/Nesov/Dai's recent posts.

Here's the outline, to be followed up with slower, clearer blog posts if all goes well:

0. **Prelude: "Should" depends on counterfactuals.** Newcomb's problem -- the problem of what Joe "should" do, to earn most money -- is the problem of which type of counterfactuals best cash out the question "Should Joe take one box or two?". Disagreement about Newcomb's problem is disagreement about what sort of counterfactuals we should consider, when we try to figure out what action Joe should take.

1. **My goal in this sequence is to reduce "should" as thoroughly as I can.** More specifically, I'll make an (incomplete, but still useful) attempt to:

- Make it even more clear that our naive conceptions of "could" and "should" are conceptual inventions, and are not Physically Irreducible Existent Things. (Written here.)
- Consider why one might design an agent that uses concepts like "could" and "should" (hereafter a "Could/Should Agent", or "CSA"), rather than designing an agent that acts in some other way. Consider what specific concepts of "could" and "should" are what specific kinds of useful. (This is meant as a more thorough investigation of the issues treated by Eliezer in "Possibility and Couldness".)
- Consider why evolution ended up creating us as approximate CSAs. Consider what kinds of CSAs are likely to be how common across the multiverse.

2. **A non-vicious regress.** Suppose we're designing Joe, and we want to maximize his expected winnings. What notion of "should" *should* we design Joe to use? There's a regress here, in that creator-agents with different starting decision theories will design agents that have different starting decision theories. But it is a non-vicious regress. We can gain understanding by making this regress explicit, and asking under what circumstances agents with decision theory X will design future agents with

decision theory Y, for different values of X and Y.

3a. **When will a CDT-er build agents that use "could" and "should"?** Suppose again that you're designing Joe, and that Joe will go out in a world and win utilons on your behalf. What kind of Joe-design will maximize your expected utilons?

If we assume nothing about Joe's world, we might find that your best option was to design Joe to act as a bundle of wires which happens to have advantageous physical effects, and which doesn't act like an agent at all.

But suppose Joe's world has the following handy property: suppose Joe's actions have effects, and Joe's "policy", or the actions he "would have taken" in response to alternative inputs also have effects, but the details of Joe's internal wiring doesn't otherwise matter. (I'll call this the "policy-equivalence assumption"). Since Joe's wiring doesn't matter, you can, without penalty, insert whatever computation you like into Joe's insides. And so, if you yourself can think through what action Joe "should" take, you can build wiring that sits inside Joe, carries out the same computation you would have used to figure out what action Joe "should" take, and then prompts that action.

Joe then inherits his counterfactuals from you: Joe's model of what "would" happen "if he acts on policy X" is *your* model of what "would" happen *if you design an agent, Joe, who* acts according to policy X. The result is "act according to the policy my creator would have chosen" decision theory, now-A.K.A. "Updateless Decision Theory" (UDT). UDT one-boxes on Newcomb's problem and pays the $100 in the counterfactual mugging problem.

3b. But it is only when computational limitations are thrown in that designing Joe to be a CSA leaves you better off than designing Joe to be your top-pick hard-coded policy. So, to understand where CSAs really come from, we'll need eventually to consider how agents can use limited computation.

3c. When will a UDT-er build agents that use "could" and "should"? The answer is similar to that for a CDT-er.

3d. **CSAs are only useful in a limited domain.** In our derivations above, CSAs' usefulness depends on the policy-equivalence assumption. Therefore, if agents' computation has important effects apart from its effects on the agents' actions, the creator agent may be ill-advised to create any sort of CSA.* For example, if the heat produced by agents' physical computation has effects that are as significant as the agent's "chosen actions", CSAs may not be useful. This limitation suggests that CSAs may not be useful in a post-singularity world, since in such a world matter may be organized to optimize for computation in a manner far closer to physical efficiency limits, and so the physical side-effects of computation may have more relative significance compared to the computation's output.

4. **What kinds of CSAs make sense?** More specifically, what kinds of counterfactual "coulds" make sense as a basis for a CSA?

In part 4, we noted that when Joe's policy is all that matters, you can stick your "What policy should Joe have?" computer inside Joe, without disrupting Joe's payoffs. Thus, you can build Joe to be a "carry out the policy my creator would think best" CSA.

It turns out this trick can be extended.

Suppose you aren't a CDT-er. Suppose you are more like one of Eliezer's "timeless" agents. When you think about what you "could" and "should" do, you do your counterfactuals, not over what you alone will do, but over what you and a whole set of other agents "running the same algorithm you are running" will simultaneously do. For example, you may (in your model) be choosing what algorithm you and Clippy will both send into a one-shot prisoner's dilemma.

Much as was the case with CDT-ers, so long as your utility estimate depends only on the algorithm's outputs and not its details you can choose the algorithm you're creating to be an "updateless", "act according to the policy your creator would have chosen" CSA.

5. **Which types of CSAs will create which other types of CSAs under what circumstances?** I go through the list above.

6. A partial list of remaining problems, and of threads that may be useful to pull on.

6.a. **Why design CSAs at all, rather than look-up tables or non-agent-like jumbles of wires?** Computational limitations are part of the answer: if I design a CSA to play chess with me, it knows what move it has to respond to, and so can focus its computation on that specific situation. Does CSAs' usefulness in focussing computation shed light on what type of CSAs to design?

6.b. **More generally, how did evolution come to build us humans as approximate CSAs?** And what kinds of decision theory should other agent-design processes, in other parts of the multiverse, be expected to create?

6.c. **What kind of a CSA are you?** What are you really asking, when you ask what you "should" do in Newcomb's problem? What algorithm do *you* actually run, and want to run, there?

6.d. **Two-player games: avoiding paradoxes and infinite types.** I used a simplification above: I assumed that agents took in inputs from a finite list, and produced outputs from a finite list. This simplification does not allow for two general agents to, say, play one another in prisoner's dilemma while seeing one another's policy. If I can choose any policy that is a function from the set of you policy options to {C, D}, and you can choose any policy that is a function from the set of *my* policy options to {C, D}, each of us must have more policy-options than the other.

Some other formalism is needed for two-player games. (Eliezer lists this problem, and the entangled problem 6.e, in his Timeless decision theory: problems I can't solve.)

6.e. **What do real agents do in the situations Eliezer has been calling "problems of logical priority before time"?** Also, what are the natural alternative decision theories to use for such problems, and is there one which is so much more natural than others that we might expect it to populate the universe, just as Eliezer hopes his "timeless decision theory" might accurately describe the bulk of decision agents in the universe?

Note that this question is related to, but harder than, the more limited question in 7.b. It is harder because we are now asking our CSAs to produce actions/outputs in more complicated situations.

6.f. **Technical machinery for dealing with timeless decision theories.** [Steve Rayhawk added this item.] As noted in 5 above, and as Eliezer noted in Ingredients of Timeless Decision Theory, we may wish to use a decision theory where we are "choosing" the answer to a particular math problem, or the output of a particular algorithm. Since the output of an algorithm is a logical question, this requires reasoning under uncertainty about the answers to logical questions. Setting this up usefully, without paradoxes or inconsistencies, requires some work. Steve has a gimmick for treating part of this problem. (The gimmick starts from the hierarchical Bayesian modeling idea of a hyperprior, used in its most general form: a prior belief about conditional probability tables for other variables.)

---

*More precisely: CSAs' usefulness breaks down if the creator's world-model includes important effects from its choice of which agent it creates, apart from the effects of that agent's policy.

Confusion about Newcomb is confusion about counterfactuals

(This is the first, and most newcomer-accessible, post in a planned sequence.)

Newcomb's Problem:

> *Joe walks out onto the square. As he walks, a majestic being flies by Joe's head with a box labeled "brain scanner", drops two boxes on the ground, and departs the scene. A passerby, known to be trustworthy, comes over and explains...*
>
> *If Joe aims to get the most money, should Joe take one box or two?*

What are we asking when we ask what Joe "should" do? It is common to cash out "should" claims as counterfactuals: "If Joe were to one-box, he would make more money". This method of translating "should" questions does seem to capture something of what we mean: we do seem to be asking how much money Joe can expect to make "if he one-boxes" vs. "if he two-boxes". The trouble with this translation, however, is that it is not clear what world "if Joe were to one-box" should refer to -- and, therefore, it is not clear how much money we should say Joe would make, "if he were to one-box". After all, Joe is a deterministic physical system; his current state (together with the state of his future self's past light-cone) fully determines what Joe's future action will be. There is no Physically Irreducible Moment of Choice, where this same Joe, with his own exact actual past, "can" go one way or the other.

To restate the situation more clearly: let us suppose that this Joe, standing here, is poised to two-box. In order to determine how much money Joe "would have made if he had one-boxed", let us say that we imagine reaching in, with a magical sort of world-surgery, and altering the world so that Joe one-boxes instead. We then watch to see how much money Joe receives, in this surgically altered world.

The question before us, then, is what sort of magical world-surgery to execute, before we watch to see how much money Joe "would have made if he had one-boxed". And the difficulty in Newcomb's problem is that there is not one but two obvious world-surgeries to consider. First, we might surgically reach in, after Omega's departure, and alter Joe's box-taking only -- leaving Omega's prediction about Joe untouched. Under this sort of world-surgery, Joe will do better by two-boxing:

Expected value (Joe's earnings if he two-boxes | some unchanged probability distribution on Omega's prediction) >
Expected value (Joe's earnings if he one-boxes | the same unchanged probability distribution on Omega's prediction).

Second, we might surgically reach in, after Omega's departure, and simultaneously alter both Joe's box-taking and Omega's prediction concerning Joe's box-taking. (Equivalently, we might reach in before Omega's departure, and surgically alter the insides of Joe brain -- and, thereby, alter both Joe's behavior and Omega's prediction of Joe's behavior.) Under this sort of world-surgery, Joe will do better by one-boxing:

Expected value (Joe's earnings if he one-boxes | Omega predicts Joe accurately) >

Expected value (Joe's earnings if he two-boxes | Omega predicts Joe accurately).

The point: Newcomb's problem -- the problem of what Joe "should" do, to earn most money -- is the problem which type of world-surgery best cashes out the question "Should Joe take one box or two?". Disagreement about Newcomb's problem is disagreement about what sort of world-surgery we should consider, when we try to figure out what action Joe should take.

Decision theory: Why we need to reduce “could”, “would”, “should”

(This is the second post in a planned sequence.)

Let’s say you’re building an artificial intelligence named Bob. You’d like Bob to sally forth and win many utilons on your behalf. How should you build him? More specifically, should you build Bob to have a world-model in which there are many different actions he “could” take, each of which “would” give him particular expected results? (Note that e.g. evolution, rivers, and thermostats do not have explicit “could”/“would”/“should” models in this sense -- and while evolution, rivers, and thermostats are all varying degrees of stupid, they all still accomplish specific sorts of world-changes. One might imagine more powerful agents that also simply take useful actions, without claimed “could”s and “woulds”.)

My aim in this post is simply to draw attention to “could”, “would”, and “should”, as concepts folk intuition fails to understand, but that seem nevertheless to do something important for real-world agents. If we want to build Bob, we may well need to figure out what the concepts “could” and “would” can do for him.*

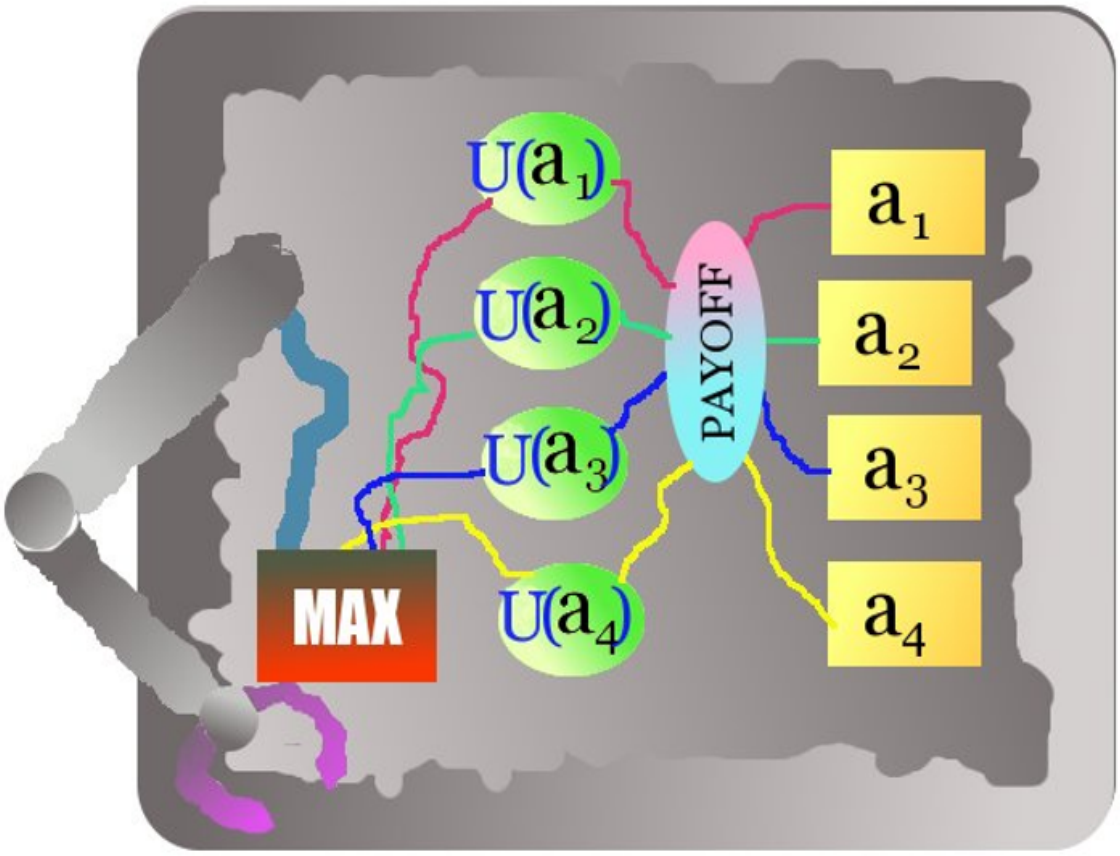
Introducing Could/Would/Should agents:

Let a Could/Would/Should Algorithm, or CSA for short, be any algorithm that chooses its actions by considering a list of alternatives, estimating the payoff it “would” get “if” it took each given action, and choosing the action from which it expects highest payoff.

That is: let us say that to specify a CSA, we need to specify:

1. A list of alternatives a_1, a_2, ..., a_n that are primitively labeled as actions it “could” take;
2. For each alternative a_1 through a_n, an expected payoff U(a_i) that is labeled as what “would” happen if the CSA takes that alternative.

To be a CSA, the algorithm must then search through the payoffs for each action, and must then trigger the agent to actually take the action a_i for which its labeled U(a_i) is maximal.



Note that we can, by this definition of "CSA", create a CSA around *any* made-up list of "alternative actions" and of corresponding "expected payoffs".

The puzzle is that CSAs are common enough to suggest that they're useful -- but it isn't clear *why* CSAs are useful, or quite what kinds of CSAs are what *kind* of useful. To spell out the puzzle:

Puzzle piece 1: CSAs are common. Humans, some (though far from all) other animals, and many human-created decision-making programs (game-playing programs, scheduling software, etc.), have CSA-like structure. That is, we consider "alternatives" and act out the alternative from which we "expect" the highest payoff (at least to a first approximation). The ubiquity of approximate CSAs suggests that CSAs are in some sense useful.

Puzzle piece 2: The naïve realist model of CSAs' nature and usefulness doesn't *work as an explanation.*

That is: many people find CSAs' usefulness unsurprising, because they imagine a Physically Irreducible Choice Point, where an agent faces Real Options; by thinking hard, and choosing the Option that looks best, naïve realists figure that you can get the best-looking option (instead of one of those other options, that you Really Could have gotten).

But CSAs, like other agents, are deterministic physical systems. Each CSA executes a single sequence of physical movements, some of which we consider "examining alternatives", and some of which we consider "taking an action". It isn't clear why or in what sense such systems do better than deterministic systems built in some other way.

Puzzle piece 3: Real CSAs are presumably not built from arbitrarily labeled "coulds" and "woulds" -- presumably, the "woulds" that humans and others use, when considering e.g. which chess move to make, have useful properties. *But it isn't clear what those properties are, or how to build an algorithm to compute "woulds" with the desired properties.*

Puzzle piece 4: On their face, all calculations of counterfactual payoffs ("woulds") involve asking questions about impossible worlds. It is not clear how to interpret such questions.

Determinism notwithstanding, it is tempting to interpret CSAs' "woulds" -- our U(a_i)s above -- as calculating what "really would" happen, if they "were" somehow able to take each given action.

But if agent X will (deterministically) choose action a_1, then when he asks what would happen "if" he takes alternative action a_2, he's asking what would happen if something impossible happens.

If X is to calculate the payoff "if he takes action a_2" as part of a causal world-model, he'll need to choose some particular meaning of "if he takes action a_2" – some meaning that allows him to combine a model of himself taking action a_2 with the rest of his current picture of the world, without allowing predictions like "if I take action a_2, then the laws of physics will have been broken".

We are left with several questions:

- Just what are humans, and other common CSAs, calculating when we imagine what "would" happen "if" we took actions we won't take?
- In what sense, and in what environments, are such "would" calculations useful? Or, if "would" calculations are not useful in any reasonable sense, how did CSAs come to be so common?
- Is there more than one natural way to calculate these counterfactual "would"s? If so, what are the alternatives, and which alternative works best?

---

*A draft-reader suggested to me that this question is poorly motivated: what other kinds of agents could there be, besides "could"/"would"/"should" agents? Also, how could modeling the world in terms of "could" and "would" *not* be useful to the agent?

My impression is that there is a sort of gap in philosophical wariness here that is a bit difficult to bridge, but that one *must* bridge if one is to think well about AI design. I'll try an analogy. In my experience, beginning math students simply expect their nice-sounding procedures to work. For example, they expect to be able to add fractions straight across. When you tell them they can't, they demand to know *why* they can't, as though most nice-sounding theorems are true, and if you want to claim that one isn't, the burden of proof is on you. It is only after students gain considerable mathematical sophistication (or experience getting burned by expectations that don't pan out) that they place the burden of proofs on the theorems, assume theorems false

or un-usable until proven true, and try to actively construct and prove their mathematical worlds.

Reaching toward AI theory is similar. If you don't understand how to reduce a concept -- how to build circuits that compute that concept, and what exact positive results will follow from that concept and will be absent in agents which don't implement it -- you need to keep analyzing. You need to be suspicious of anything you can't derive for yourself, from scratch. Otherwise, even if there is something of the sort that is useful in the specific context of your head (e.g., some sort of "could"s and "would"s that do you good), your attempt to re-create something similar-looking in an AI may well lose the usefulness. You get cargo cult could/woulds.

+ Thanks to Z M Davis for the above gorgeous diagram.

Decision theory: Why Pearl helps reduce “could” and “would”, but still leaves us with at least three alternatives

(This is the third post in a planned sequence.)

My last post left us with the questions:

- Just what are humans, and other common CSAs, calculating when we imagine what “would” happen “if” we took actions we won’t take?
- Is there more than one natural way to calculate these counterfactual “would”s? If so, what are the alternatives, and which alternative works best?

Today, I’ll take an initial swing at these questions. I’ll review Judea Pearl’s causal Bayes nets; show how Bayes nets offer a general methodology for computing counterfactual “would”s; and note *three* plausible alternatives for how to use Pearl’s Bayes nets to set up a CSA. One of these alternatives will be the “timeless” counterfactuals of Eliezer’s Timeless Decision Theory.

The problem of counterfactuals is the problem what we do and should mean when we we discuss what “would” have happened, “if” something impossible had happened. In its general form, this problem has proved to be quite gnarly. It has been bothering philosophers of science for at least 57 years, since the publication of Nelson Goodman’s book “Fact, Fiction, and Forecast” in 1952:

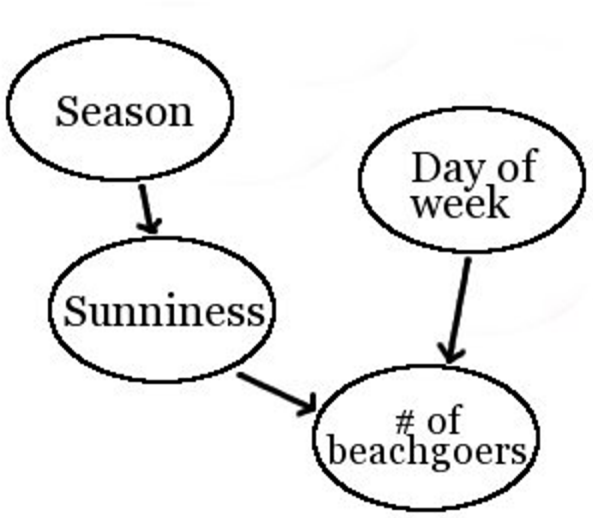
> Let us confine ourselves for the moment to [counterfactual conditionals] in which the antecedent and consequent are inalterably false--as, for example, when I say of a piece of butter that was eaten yesterday, and that had never been heated,
>
> `*If that piece of butter had been heated to 150°F, it would have melted.*'
>
> Considered as truth-functional compounds, all counterfactuals are of course true, since their antecedents are false. Hence
>
> `*If that piece of butter had been heated to 150°F, it would not have melted.*'
>
> would also hold. Obviously something different is intended, and the problem is to define the circumstances under which a given counterfactual holds while the opposing conditional with the contradictory consequent fails to hold.

Recall that we seem to need counterfactuals in order to build agents that do useful decision theory -- we need to build agents that can think about the consequences of each of their “possible actions”, and can choose the action with best expected-consequences. So we need to know how to compute those counterfactuals. As Goodman puts it, “[t]he analysis of counterfactual conditionals is no fussy little grammatical exercise.”

Judea Pearl’s Bayes nets offer a method for computing counterfactuals. As noted, it is hard to reduce human counterfactuals in general: it is hard to build an

algorithm that explains what (humans will say) really “would” have happened, “if” an impossible event had occurred. But it is easier to construct specific formalisms within which counterfactuals have well-specified meanings. Judea Pearl’s causal Bayes nets offer perhaps the best such formalism.

Pearl’s idea is to model the world as based on some set of causal variables, which may be observed or unobserved. In Pearl’s model, each variable is determined by a conditional probability distribution on the state of its parents (or by a simple probability distribution, if it has no parents). For example, in the following Bayes net, the beach’s probability of being “Sunny” depends only on the “Season”, and the probability that there is each particular “Number of beach-goers” depends only on the “Day of the week” and on the “Sunniness”. Since the “Season” and the “Day of the week” have no parents, they simply have fixed probability distributions.



Once we have a Bayes net set up to model a given domain, computing counterfactuals is easy*. We just:

1. Take the usual conditional and unconditional probability distributions, that come with the Bayes net;
2. Do “surgery” on the Bayes net to plug in the variable values that define the counterfactual situation we’re concerned with, while ignoring the parents of surgically set nodes, and leaving other probability distributions unchanged;
3. Compute the resulting probability distribution over outcomes.

For example, suppose I want to evaluate the truth of: “If last Wednesday had been sunny, there would have been more beach-goers”. I leave the “Day of the week” node at Wednesday“, set the ”Sunny?“ node to ”Sunny“, ignore the “Season” node, since it is the parent of a surgically set node, and compute the probability distribution on beach-goers.

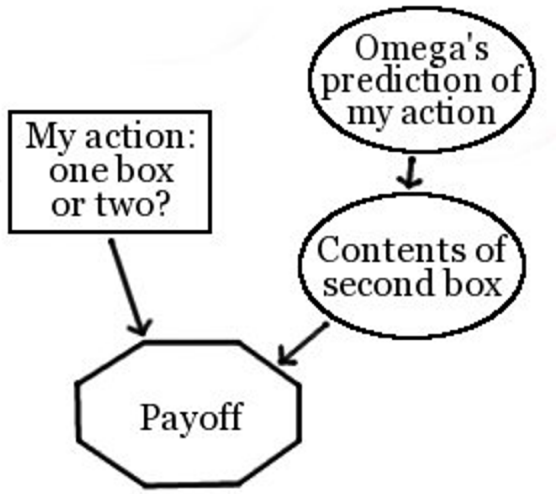
*Okay, not quite easy: I’m sweeping under the carpet the conversion from the English counterfactual to the list of variables to surgically alter, in step 2. Still, Pearl’s Bayes nets do much of the work.

But, even if we decide to use Pearl's method, we are left with the choice of how to represent the agent's "possible choices" using a Bayes net. More specifically, we are left with the choice of what surgeries to execute, when we represent the alternative actions the agent "could" take.

There are at least three plausible alternatives:

Alternative One: "Actions CSAs":

Here, we model the outside world however we like, but have the agent's own "action" -- its choice of a_1, a_2, or ... , a_n -- be the critical "choice" node in the causal graph. For example, we might show Newcomb's problem as follows:



The assumption built into this set-up is that the agent's action is uncorrelated with the other nodes in the network. For example, if we want to program an understanding of Newcomb's problem into an Actions CSA, we are forced to choose a probability distribution over Omega's prediction that is independent of the agent's actual choice.

How Actions CSAs reckon their coulds and woulds:

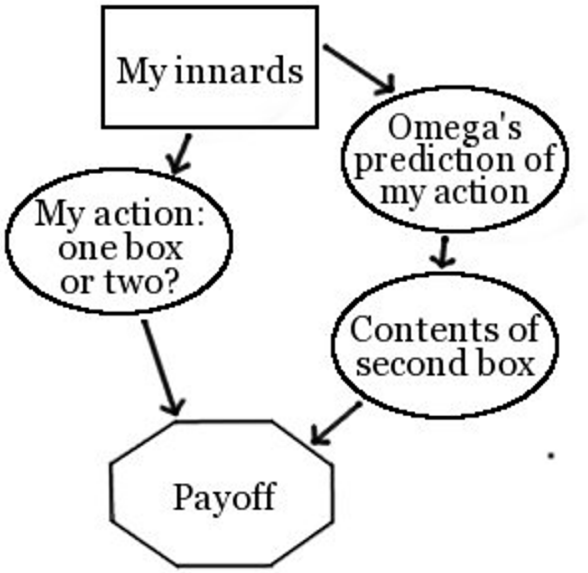
- Each "could" is an alternative state of the "My action" node. Actions CSAs search over each state of the "action" node before determining their action.
- Each "would" is then computed in the usual Bayes net fashion: the "action" node is surgically set, the probability distribution for other nodes is left unchanged, and a probability distribution over outcomes is computed.

So, if causal decision theory is what I think it is, an "actions CSA" is simply a causal decision theorist. Also, Actions CSAs will two-box on Newcomb's problem, since, in their network, the contents of box B is independent of their choice to take box A.

Alternative Two: "Innards CSAs":

Here, we again model the outside world however we like, but we this time have the agent's own "innards" -- the physical circuitry that interposes between the agent's

sense-inputs and its action-outputs -- be the critical “choice” node in the causal graph. For example, we might show Newcomb’s problem as follows:



Here, the agent’s innards are allowed to cause both the agent’s actions and outside events -- to, for example, we can represent Omega’s prediction as correlated with the agent’s action.

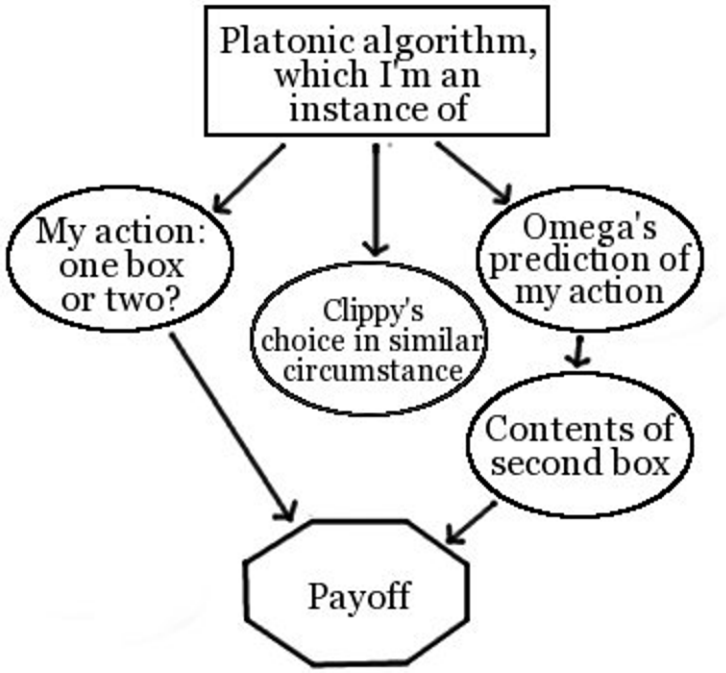
How Innards CSAs reckon their coulds and woulds:

- Each “could” is an alternative state of the “My innards” node. Innards CSAs search over each state of the “innards” node before determining their optimal innards, from which their action follows.
- Each “would” is then computed in the usual Bayes net fashion: the “innards” node is surgically set, the probability distribution for other nodes is left unchanged, and a probability distribution over outcomes is computed.

Innards CSAs will one-box on Newcomb’s problem, because they reason that if their innards were such as to make them one-box, those same innards would cause Omega, after scanning their brain, to put the $1M in box B. And so they “choose” innards of a sort that one-boxes on Newcomb’s problem, and they one-box accordingly.

Alternative Three: “Timeless” or “Algorithm-Output” CSAs:

In this alternative, as Eliezer suggested in Ingredients of Timeless Decision Theory, we have a “Platonic mathematical computation” as one of the nodes in our causal graph, which gives rise at once to our agent’s decision, to the beliefs of accurate predictors about our agent’s decision, and to the decision of similar agents in similar circumstances. It is the output to this *mathematical* function that our CSA uses as the critical “choice” node in its causal graph. For example:



How Timeless CSAs reckon their coulds and woulds:

- Each “could” is an alternative state of the “Output of the Platonic math algorithm that I’m an instance of” node. Timeless CSAs search over each state of the “algorithm-output” node before determining the optimal output of this algorithm, from which their action follows.
- Each “would” is then computed in the usual Bayes net fashion: the “algorithm-output” node is surgically set, the probability distribution for other nodes is left unchanged, and a probability distribution over outcomes is computed.

Like innards CSAs, algorithm-output CSAs will one-box on Newcomb’s problem, because they reason that if the output of their algorithm was such as to make them one-box, that same algorithm-output would also cause Omega, simulating them, to believe they will one-box and so to put $1M in box B. They therefore “choose” to have their algorithm output “one-box on Newcomb’s problem!”, and they one-box accordingly.

Unlike innards CSAs, algorithm-output CSAs will also Cooperate in single-shot prisoner’s dilemmas against Clippy -- in cases where they think it sufficiently likely that Clippy’s actions are output by an instantiation of “their same algorithm” -- even in cases where Clippy cannot at all scan their brain, and where their innards play no physically causal role in Clippy’s decision. (An Innards CSA, by contrast, will Cooperate if having Cooperating-type innards will physically cause Clippy to cooperate, and not otherwise.)

Coming up: considerations as to the circumstances under which each of the above types of agents will be useful, under different senses of “useful”.

Thanks again to Z M Davis for the diagrams.