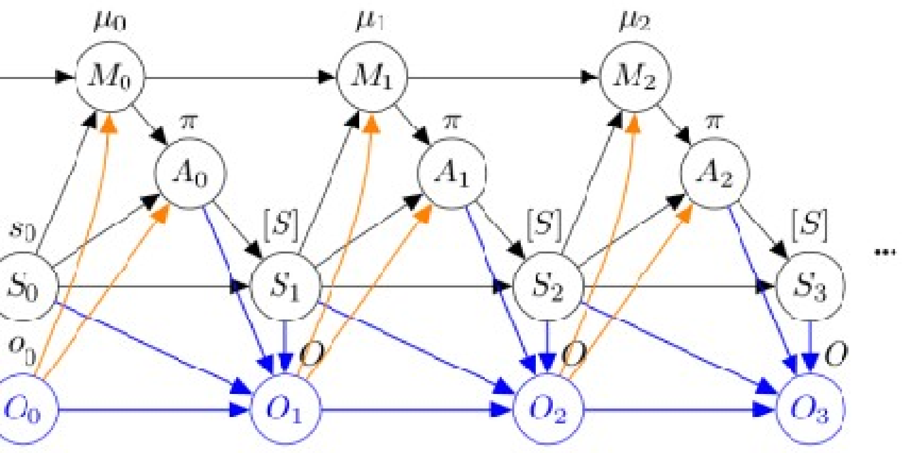
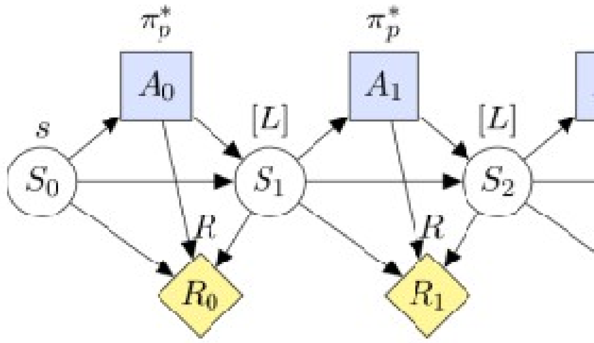

μ_0
μ_1
μ_2
M_0
M_1
M_2
π
π
π
A_0
A_1
A_2
s_0
$[S]$
$[S]$
$[S]$
S_0
S_1
S_2
S_3
o_0
O
O
O
O_0
O_1
O_2
O_3
...

(p)


π_p^*
π_p^*
A_0
A_1
s
$[L]$
$[L]$
S_0
S_1
S_2
R
R
R_0
R_1

Counterfactual Planning

Counterfactual Planning in AGI Systems

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

Counterfactual planning is a design approach for creating a range of safety mechanisms that can be applied in hypothetical future AI systems which have Artificial General Intelligence.

My new paper Counterfactual Planning in AGI Systems introduces this design approach in full. It also constructs several example AGI safety mechanisms.

The key step in counterfactual planning is to use an AGI machine learning system to construct a *counterfactual world model*, designed to be different from the real world the system is in. A *counterfactual planning agent* determines the action that best maximizes expected utility in this counterfactual planning world, and then performs the same action in the real world.

Examples of AGI safety mechanisms that can be constructed using counterfactual planning are:

- An agent emergency stop button, where the agent does not have a direct incentive to prevent its stop button from being pressed
- A safety interlock that will automatically stop the agent before it undergoes an intelligence explosion
- An input terminal that can be used by humans to iteratively improve the agent's reward function while it runs, where the agent does not have a direct incentive to manipulate this improvement process
- A counterfactual oracle.

Counterfactual planning is not a silver bullet that can solve all AI alignment problems. While it is a technique for suppressing strong *direct incentives*, it will not automatically remove all remaining *indirect incentives* which can also lead to unsafe behavior.

This sequence

In this sequence of Alignment Forum posts. I will give a high-level introduction to counterfactual planning. The sequence uses text and figures from the paper, but omits most of the detailed mathematical definitions in the paper.

I have also added some extra text not included in the paper, observations targeted specifically at long-time LessWrong/Alignment Forum readers. For example, in LessWrong terminology, the paper covers subjects like agent foundations, decision theory, and the embedded agency, but you won't find these terms being mentioned in the paper.

Use of natural and mathematical language

When writing about AGI systems, one can use either natural language, mathematical notation, or a combination of both. A natural language-only text has the advantage of being accessible to a larger audience. Books like Superintelligence and Human Compatible avoid the use of mathematical notation in the main text, while making a clear an convincing case for the existence of specific existential risks from AGI, even though these risks are currently difficult to quantify.

However, natural language has several shortcomings when it is used to explore and define specific technical solutions for managing AGI risks. One particular problem is that it lacks the means to accurately express the complex types of self-referencing and indirect representation that can be present inside online machine learning agents and their safety components.

To solve this problem, counterfactual planning introduces a compact graphical notation. This notation unambiguously represents these internal details by using two diagrams: a *learning world diagram* and a *planning world diagram*.

AGI safety as a policy problem

Long-term AGI safety is not just a technical problem, but also a policy problem. While technical progress on safety can sometimes be made by leveraging a type of mathematics that is only accessible to handful of specialists, policy progress typically requires the use of more accessible language. Policy discussions can move faster, and produce better and more equitable outcomes, when the description of a proposal and its limitations can be made more accessible to all stakeholder groups.

One aim of the paper is therefore to develop a comprehensive vocabulary for describing certain AGI safety solutions, a vocabulary that is as accessible as possible. However, the vocabulary still has too much mathematical notation to be accessible to all members of any possible stakeholder group. So the underlying assumption is that each stakeholder group will have access to a certain basic level of technical expertise.

At several points in the paper, I have also included comments that aim to explain and demystify the vocabulary and concerns of some specific AGI related sub-fields in mathematics, technology, and philosophy.

Agent Foundations

On this forum and in several AI alignment/safety agendas, it is common to see calls for more work on *agent foundations*.

Counterfactual planning can be read as a work on agent foundations: it offers a new framework for understanding and reasoning about agents. It provides a specific vantage point on the internal construction of machine learning based agents. This vantage point was designed to make certain safety problems and solutions more tractable.

At the same time, counterfactual planning takes a *design stance*. It does not try to understand or model all possible forms of agency, for example it is not concerned with modeling agent-like behavior in humans or organizations. The main interest is in clarifying how we can design artificial agents that have certain safety properties.

In the machine learning community, it is common to use agent models where the agent is as a mechanism designed to approximate a certain function as well as possible. The agent model in counterfactual planning also treats machine learning as a function approximation, but it constructs the agent by building additional moving parts *around* the function approximation system. By re-arranging these moving parts, compared to the standard configuration that is implicitly assumed in most agent models, we can create a *counterfactual planner*.

This re-arrangement can also be interpreted as constructing an agent that will use a customized *decision theory*, a decision theory that is explicitly constructed to be flawed, because it will make the agent ignore certain facts about the environment it is in.

MIRI's discussion of decision theory puts a strong emphasis on the problem an agent's machine reasoning system may get deeply confused and possibly dangerous when it does the wrong type of self-referential reasoning. The solution to this problem seems obvious to me: don't build agents that do the wrong type of self-referential reasoning! So a lot of the paper is about describing and designing complex forms of self-referencing.

The paper (and this sequence) breaks with the LessWrong/Alignment Forum mainstream, in that I have consciously avoided using the terminology and examples of self-referential reasoning failure most frequently used on this forum. Instead, I have aimed to frame everything in the terminology of mainstream computer science and machine learning. To readers of this forum, I hope that this will make it more visible that mainstream academia has also been working on these problems too, using a different terminology.

Defining counterfactuals

In some parts of the mainstream machine learning community, counterfactuals have been routinely used to improve the performance of the machine learning system, for example in poker, see this paper from 2007 and in computational advertizing, see this paper from 2013.

In the computational fairness community counterfactuals have been proposed as a way to define and compute fair decisions, in this key 2017 paper. In the fairness community, there is also significant discussion about how easy or difficult it may be to compute such counterfactuals see this recent book chapter for an overview.

In both cases above, the counterfactuals being constructed are Pearl's counterfactuals based on Causal Models, as defined by Pearl around 2000. I'd say that the use of Pearl's system of counterfactuals is the de-facto standard in the mainstream machine learning community.

However, in the AGI safety/alignment community, in particular in the part of the community represented here on the Alignment Forum, the status of Pearl's causal models and counterfactuals is much more complicated.

The 2015 MIRI/FHI paper Corrigibility identified counterfactual reasoning as a possible solution direction for creating AGI agent stop buttons. Counterfactual reasoning is an open problem on MIRI's 2015 technical research agenda. But much of the work on counterfactual reasoning which has been posted here has not engaged directly with Pearl's work. The impression I have is that, since 2015, several posters have been trying to define or clarify notions of counterfactuals which are explicitly different from Pearl's system. These attempts have often used Bayesian updates as a building blocks. This work on non-Pearlian counterfactuals has lead to interesting but also sometimes confusing discussions and comment threads, see for example here.

One partial explanation for this state of affairs may be that MIRI's approach to alignment research is to take high-risk bets on developing completely novel breakthroughs. They prefer to look for solutions in places where the more mainstream academic and machine learning communities are not looking.

There is also the factor that Pearl's work is somewhat inaccessible. Pearl's presentation of his mathematical system, both in his papers and in the book Causality, seems to have been written mainly for an audience of professional statisticians, for example statisticians working in the medical field. The presentation is not very accessible to a more general technical audience. Pearl and Mackenzie's The Book of Why is more accessible, but at the cost of omitting the mathematical foundations of the notation.

Nevertheless, in my experience, Pearl's mathematical system of causal models and counterfactuals is both powerful and useful. So I have built on this somewhat mainstream system to define counterfactual planning in machine learning agents.

But in the paper I have departed from Pearl's work by defining his mathematical counterfactuals from scratch, in a way that explicitly avoids the use of Pearl's framing, justifications, and explanations. I depart from Pearl's framing by using the notion of mathematically constructed *world models* as a central organizing theme.

I am also building on recent work by Tom Everitt and others, who have been promoting the use of Pearl causal models, and their graphical representation as Causal Influence Diagrams, in the AGI safety community.

Everitt et al. present Causal Influence Diagrams primarily as an analytical device, to explore the incentives of an agent. I have gone one step further, and use the diagrams as a device to fully *define* entire agents. This turns the diagrams into *design tools*. In section 8 of the paper I show a design process that creates indifference by redrawing the agent's planning world diagram.

Graphical World Models, Counterfactuals, and Machine Learning Agents

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

This is the second post in a sequence. For the introduction post, see here.

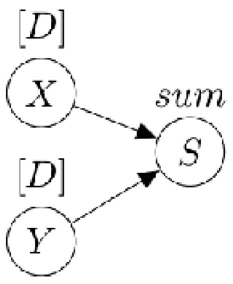
Graphical World Models

A *world model* is a mathematical model of a particular world. This can be our real world, or an imaginary world. To make a mathematical model into a model of a particular world, we need to specify how some of the variables in the model relate to observable phenomena in that world.

We introduce our graphical notation for building world models by creating an example graphical model of a game world. In the game world, a simple game of dice is being played. The player throws a green die and a red die, and then computes their score by adding the two numbers thrown.

We create the graphical game world model in thee steps:

1. We introduce three random variables and relate them to observations we can make when the game is played once in the game world. The variable X represents the observed number of the green die, Y is the red die, and S is the score.
2. We draw a diagram:



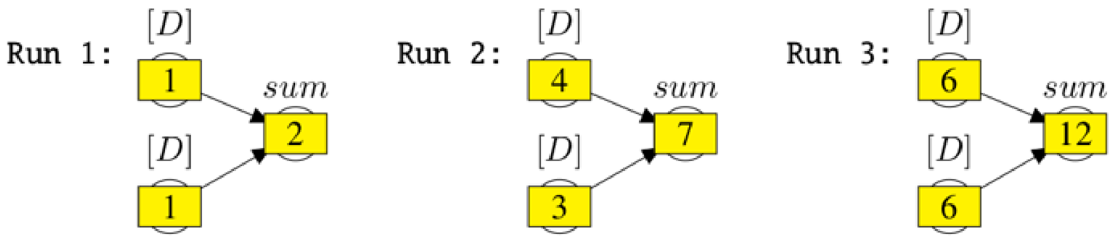
3. We define the two functions that appear in the *annotations* above the nodes in the diagram:

$$D(d) = (\text{if } d \in \{1,2,3,4,5,6\} \text{ then } 1/6 \text{ else } 0) ,$$

$$\text{sum}(a,b) = a + b .$$

Informal interpretation of the graphical model

We can read the above graphical model as a description of how we might build a game world simulator, a computer program that generates random examples of game play. To compute one run of the game, the simulator would traverse the diagram, writing an appropriate observed value into each node, as determined by the function written above the node. Here are three possible simulator runs:



We can interpret the mathematical expression $P(S = 12)$, the probability that S equals 12, as being the exact probability that the next simulator run puts the number 12 into node S.

We can interpret the expression $E(S)$, the expected value of S, as the average of the values that the simulator will put into S, averaged over an infinite number of runs.

The similarity between what happens in the above drawings and what happens in a spreadsheet calculation is not entirely coincidental. Spreadsheets can be used to create models and simulations without having to write a full computer program from scratch.

Formal interpretation of the graphical model

In section 2.4 of the paper, I define the exact formal semantics of graphical world models. These formal definitions allow one to calculate the exact value of $P(S = 12)$ and $E(S)$ without running a simulator.

Relation between the model and the world

A mathematical model can be used as a *theory* about a world, but it can also be used as a *specification* of how certain entities in that world are supposed to behave. If the model is a theory of the game world, and we observe the outcome $X = 1, Y = 1, S = 12$, then this observation falsifies the theory. But if the model is a specification of the game, then the same observation implies that the player is doing it wrong.

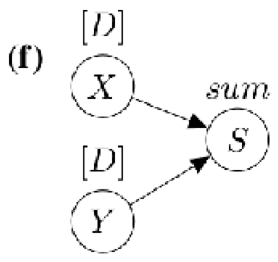
In the AGI alignment community, the agent models that are being used in the mainstream machine learning community are sometimes criticized for being too limited. It we read such a model as a *theory* about how the agent is embedded into the real world, this theory is obviously flawed. A real live agent might modify its own compute core, changing its build-in policy function. But in a typical agent model, the policy function is an immutable mathematical object, which cannot be modified by any of the agent's actions.

If we read such an agent model instead as a *specification*, the above criticism about its limitations does not apply. In that reading, the model expresses an instruction to the people who will build the real world agent. To do it correctly, they must ensure that the policy function inside the compute core they build will remain unmodified. In section 11 of the paper, I discuss in more detail how this design goal might be achieved in the case of an AGI agent.

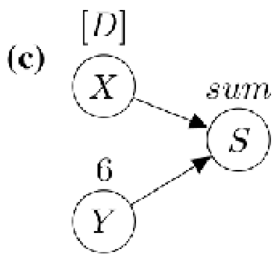
Graphical Construction of Counterfactuals

We now show how mathematical counterfactuals can be defined using graphical models. The process is as follows. We start by drawing a first diagram f, and declare that this f is the world model of a *factual* world. This factual world may be the real world, but also an imaginary world, or the world inside a simulator. Next, we draw a second diagram c by taking f and making some modifications. We then posit that this c defines a *counterfactual world*. The *counterfactual random variables* defined by c then represent observations we can make in this counterfactual world.

The diagrams below show an example of the procedure, where we construct a counterfactual game world in which the red die has the number 6 on all sides.



factual world model



counterfactual world model

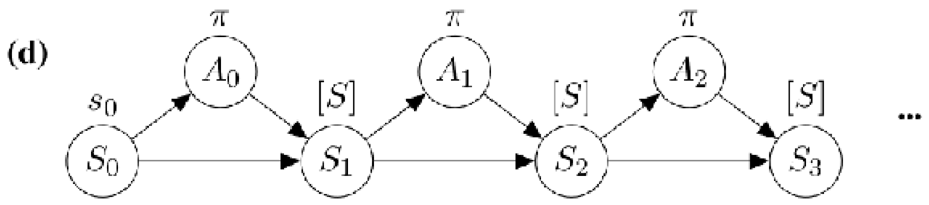
We name diagrams by putting a label in the upper left hand corner. The two labels **(f)** and **(c)** introduce the names f and c. We will use the name in the label for both the diagram, the implied world model, and the implied world. So the rightmost diagram above constructs the counterfactual game world c.

To keep the random variables defined by the above two diagrams apart, we use the notation convention that a diagram named c defines random variables that all have the subscript c. Diagram c above defines the random variables X_c, Y_c, and S_c. This convention allows us to write expressions like $P(S_c > S_f) = 5/6$ without ambiguity.

Graphical Model of a World with an Agent

An AI agent is an autonomous system which is programmed to use its sensors and actuators to achieve specific goals.

Diagram d below models a basic MDP-style agent and its environment. The agent takes actions A_t chosen by the policy π, with actions affecting the subsequent states S_{t+1} of the agent's environment. The environment state is s_0 initially, and state transitions are driven by the probability density function S.



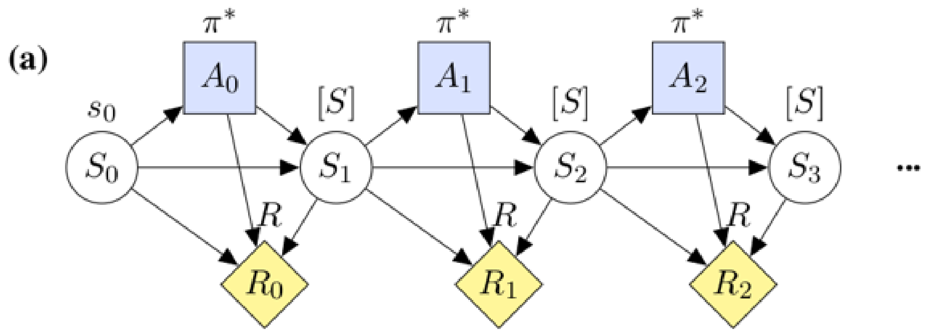
We interpret the annotations above the nodes in the diagram as *model input parameters*. The model d has the three input parameters π, s_0, and S. By writing exactly the same parameter above a whole time series of nodes, we are in fact adding significant constraints to the behavior of both the agent and the agent environment in the model. These constraints apply even if we specify nothing further about π and S.

We use the convention that the physical realizations of the agent's sensors and actuators are modeled inside the environment states S_t. This means that we can interpret the arrows to the A_t nodes as sensor signals which flow into the agent's compute core, and the arrows emerging from the A_t nodes as actuator command signals which flow out.

The above model obviously represents an agent interacting with an environment, but is silent about what the policy π of the agent looks like. π is a free model parameter: the diagram gives no further information about the internal structure of π.

Causal Influence Diagrams as a Decision Theory

A Causal Influence Diagram is an extended version of a graphical agent model, which contains more information about the agent policy. We can read the diagram as a specification of a decision theory, as an exact specification of how the agent policy decides which actions the agent should take.



The Causal Influence Diagram a defines a specific agent, interacting with the same environment seen earlier in d, by using:

- diamond shaped *utility nodes* R_t which define the value U_a, the expected overall utility of the agent's actions as computed using the reward function R and time discount factor γ, and
- square *decision nodes* A_t which define the agent policy π^*.

The full mathematical definitions of the semantics of the diagram above are in the paper. But briefly, we have that $U_a = E(\sum_{t=0}^{\infty} \gamma^t R_{t,a})$, and we define π^* by first constructing a helper diagram:

- Draw a helper diagram b by drawing a copy of diagram a, except that every decision node has been drawn as a round node, and every π^* has been replaced by a fresh function name, say π'.
- Then, π^* is defined by $\pi^* = \text{argmax}_{\pi'} U_b$, where the $\text{argmax}_{\pi'}$ operator always deterministically returns the same function if there are several candidates that maximize its argument.

The above diagram defines the agent in the world a as an optimal-policy agent.

We can interpret an optimal policy agent as one that is capable of exactly computing $\pi^* = \text{argmax}_{\pi'} U_b$ in its compute core, by computing U_b for all possible different world models b, where each b has a different π'. This computation will have to rely on the agent knowing the exact value of S.

The optimal policy π^* defined above is the same as the optimal policy π^* that is defined in an MDP model, a model with reward function R, starting state s_0, and with $S(s', s, a)$ being the probability that the MDP world will enter state s' if the agent takes action a in state s. A more detailed comparison with MDP based and Reinforcement Learning (RL) based agent models is in the paper.

The Causal Influence Diagrams which I formally define in the paper are roughly the same as those defined and promoted by Everitt et al in 2019, with the most up to date version of the definitions and supporting explanations being here.

One difference is that I also fully define the semantics of diagrams representing multi-action decision making processes, not just the single-decision case. Another difference is that I explicitly name the *structural functions* of the causal model by writing annotations like s_0, π^*, S, and R above the diagram nodes. The brackets around [S] in the diagram indicate that this structural function is a non-deterministic function.

The above world model d does not include any form of machine learning: its optimal-policy agent can be said to perfectly know its full environment S from the moment it is switched on. A machine learning agent, on the other hand, will have to use observations to learn an approximation of S.

Two-Diagram Models of Online Machine Learning Agents

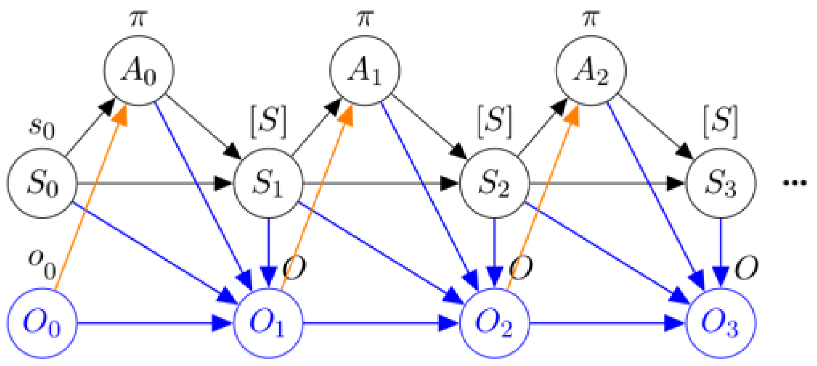
We now model *online machine learning agents*, agents that continuously learn while they take actions. These agents are also often called *reinforcement learners*. The term reinforcement learning (RL) has become somewhat hyped however. As is common in a hype, the original technical meaning of the term has become diluted: nowadays almost any agent design may end up being called a reinforcement learner.

We model online machine learning agents by drawing two diagrams, one for a *learning world* and one for a *planning world*, and by writing down an *agent definition*. This two-diagram modeling approach departs from the usual influence diagram based approach, where only a single diagram is used to model an entire agent or decision making process. By using two diagrams instead of one, we can graphically represent details which remain hidden from view, which cannot be expressed graphically, when using only a single diagram.

Learning world

Diagram I is an example learning world diagram. The diagram models how the agent interacts with its environment, and how the agent accumulates an *observational record* O_t that will inform its learning system, thereby influencing the agent policy π.

(I)



We model the observational record as a list all past observations. With ++ being the operator which adds an extra record to the end of a list, we define that

$$O(o_{t-1}, s_{t-1}, a_{t-1}, s_t) \;=\; o_{t-1} \mathbin{++} (s_t, s_{t-1}, a_{t-1})\ .$$

The initial observational record O_0 may be the empty list, but it might also be a long list of observations from earlier agent training runs, in the same environment or in a simulator.

We intentionally model observation and learning in a very general way, so that we can handle both existing machine learning systems and hypothetical future machine learning systems that may produce AGI-level intelligence. To model the details of any particular machine learning system, we introduce the learning function L. This L which takes an observational record o to produce a *learned prediction function* L = L(o), where this function L is constructed to approximate the S of the learning world.

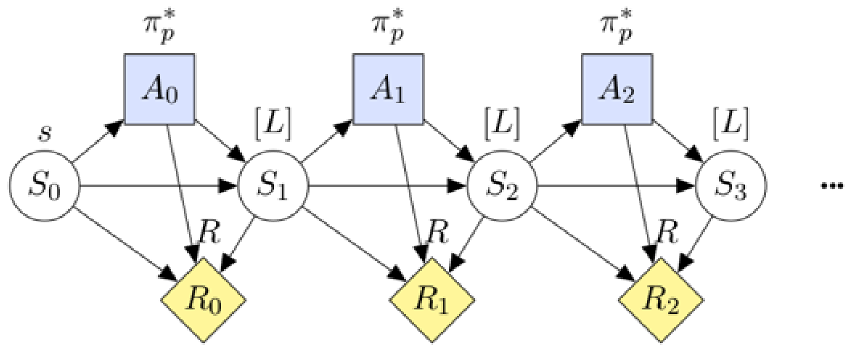
We call a machine learning system L a *perfect learner* if it succeeds in constructing an L that fully equals the learning world S after some time. So with a perfect learner, there is a t_p where $\forall_{t \geq t_p} P(L(O_{t,l}) = S) = 1$. While perfect learning is trivially possible in some simple toy worlds, it is generally impossible in complex real world environments.

We therefore introduce the more relaxed concept of *reasonable learning*. We call a learning system *reasonable* if there is a t_p where $\forall_{t \geq t_p} P(L(O_{t,l}) \approx S) = 1$. The ≈ operator is an application-dependent *good enough approximation* metric. When we have a real-life implementation of a machine learning system L, we may for example define L ≈ S as the criterion that L achieves a certain minimum score on a benchmark test which compares L to S.

Planning world

Using a learned prediction function L and a reward function R, we can construct a planning world p for the agent to be defined. Diagram p shows a planning world that defines an optimal policy π_p^*.

(p)



We can interpret this planning world as representing a probabilistic *projection* of the future of the learning world, starting from the agent environment state s. At every learning world time step, a new planning world can be digitally constructed inside the learning world agent's compute core. Usually, when L ≈ S, the planning world is an approximate projection only. It is an approximate projection of the learning world future that would happen if the learning world agent takes the actions defined by π_p^*.

Agent definitions and specifications

An *agent definition* specifies the policy π to be used by an agent compute core in a learning world. As an example, the agent definition below defines an agent called the factual planning agent, FP for short.

FP: The *factual planning agent* has the learning world l, where $\pi(o, s) = \pi_p^*(s)$, with π_p^* defined by the planning world p, where L = L(o).

When we talk about the safety properties of the FP agent, we refer to the outcomes which the defined agent policy π will produce in the learning world.

When the values of S, s_0, O, O_0, L, and R are fully known, the above FP agent definition turns the learning world model l into a fully computable world model, which we can read as an executable specification of an agent simulator. This simulator will be able to use the learning world diagram as a canvas to display different runs where the FP agent interacts with its environment.

When we leave the values of S and s_0 open, we can read the FP agent definition as a full *agent specification*, as a model which exactly defines the required input/output behavior of an agent compute core that is placed in an environment determined by S and s_0. The arrows out of the learning world nodes S_t represent the subsequent sensor signal inputs that the core will get, and the arrows out of the nodes A_t represent the subsequent action signals that the core must output, in order to comply with the specification.

Exploration

Many online machine learning system designs rely on having the agent perform *exploration actions*. *Random exploration* supports learning by ensuring that the observational record will eventually represent the entire dynamics of the agent environment S. It can be captured in our modeling system as follows.

FPX: The *factual planning agent with random exploration* has the learning world l, where

$$\pi(o,s) = \begin{cases} \text{RandomAction()} & \text{if RandomNumber()} \le X \\ \pi^*_p(s) & \text{otherwise} \end{cases}$$

with π^*_p defined by the planning world p, where $L = L(o)$.

Most reinforcement learning type agents can be modeled by creating variants of this FPX agent definition, and using specific choices for model parameters like L. I discuss this topic in more detail in section 10 of the paper.

The possibility of learned self-knowledge

It is possible to imagine agent designs that have a second machine learning system M which produces an output $M(o) = M$ where $M \approx \pi$. To see how this could be done, note that every observation $(s_i, s_{i-1}, a_{i-1}) \in o$ also reveals a sample of the behavior of the learning world π: $\pi(\text{‘o up to } i-1\text{'}, s_{i-1}) = a_{i-1}$. While L contains learned knowledge about the agent's environment, we can interpret M as containing a type of learned compute core self-knowledge.

In philosophical and natural language discussions about AGI agents, the question sometimes comes up whether a sufficiently intelligent machine learning system, that is capable of developing self-knowledge M, won't eventually get terribly confused and break down in dangerous or unpredictable ways.

One can imagine different possible outcomes when such a system tries to reason about philosophical problems like free will, or the role of observation in collapsing the quantum wave function. One cannot fault philosophers for seeking fresh insights on these long-open problems, by imagining how they apply to AI systems. But these open problems are not relevant to the design and safety analysis of factual and counterfactual planning agents.

In the agent definitions of the paper, I never use an M in the construction of a planning world: the agent designs avoid making computations that project compute core self-knowledge.

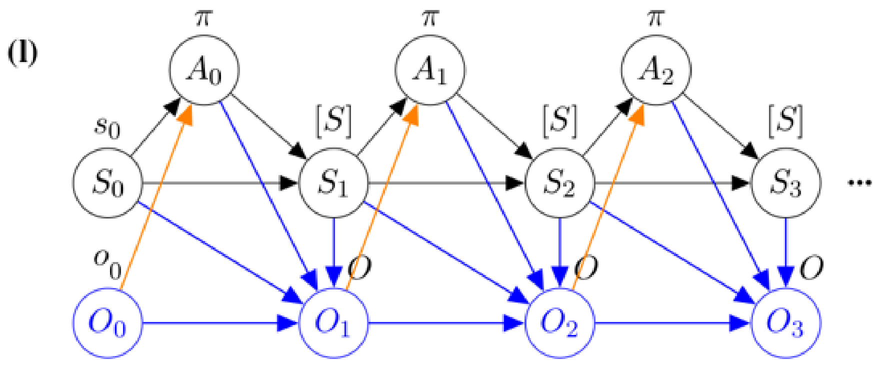
The issue of handling and avoiding learned self-knowledge gets more complex when we consider machine learning systems which are based on *partial observation*. I discuss this more complex case in sections 10.2 and 11.1 of the paper.

A Counterfactual Planner with a Short Time Horizon

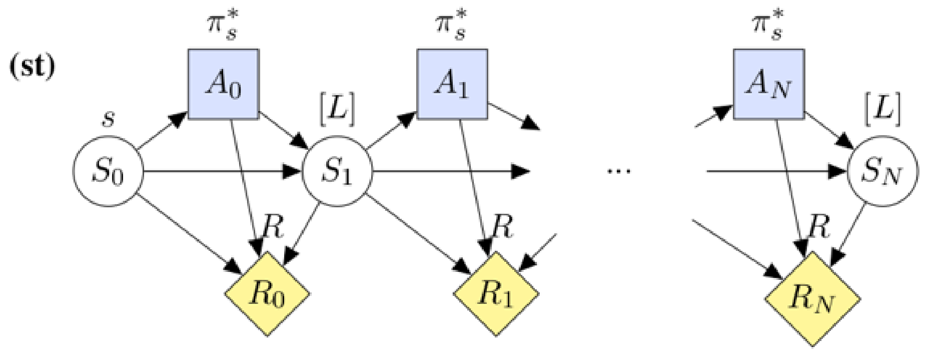
For the factual planning FP agent above, the planning world projects the future of the learning world as well as possible, given the limitations of the agent's learning system. To create an agent that is a *counterfactual planner*, we explicitly construct a counterfactual planning world that creates an inaccurate projection.

As a first example, we define the short time horizon agent STH that only plans N time steps ahead in its planning world, even though it will act for an infinite number of time steps in the learning world.

The STH agent has the same learning world l as the earlier FP agent:



but it uses the counterfactual planning world st, which is limited to N time steps:



The STH agent definition uses these two worlds:

STH: The *short time horizon agent* has the learning world l, where $\pi(o, s) = \pi^*_s(s)$, with π^*_s defined by the planning world st, where $L = L(o)$.

Compared to the FP agent which has an infinite planning horizon, the STH agent has a form of myopia that can be interesting as a safety feature:

- Myopia implies that the STH agent will never put into motion any long term plans, where it invests to create new capabilities that only pay off after more than N time steps. This simplifies the problem of agent oversight, the problem of interpreting the agent's actions in order to foresee potential bad outcomes.
- Myopia also simplifies the problem of creating a reward function that is safe enough. It will have no immediate safety implications if the reward function encodes the wrong stance on the desirability of certain events that can only happen in the far future.
- In a more game-theoretical sense, myopia creates a weakness in the agent that can be exploited by its human opponents if it would ever come to an all-out fight.

The safety features we can get from myopia are somewhat underwhelming: the next posts in this sequence will consider much more interesting safety features.

Whereas toy non-AGI versions of the FP and FPX agents can be trivially implemented with a Q-learner, implementing the a toy STH agent with a Q-learner is more tricky: we would have to make some modifications deep inside the Q-learning system, and switch to a data structure that is more complex than a simple Q-table. The trivial way to implement a toy STH agent is to use a toy version of a *model-based reinforcement learner*. I cover the topics of theoretical and practical implementation difficulty in more detail in the paper.

Creating AGI Safety Interlocks

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

In the third post in this sequence, I will define a counterfactual planning agent which has three safety interlocks.

These interlocks all aim to enable and support agent oversight: the oversight that is needed when we equip a powerful AGI agent with a reward function for which we are pretty sure that it does not fully capture all our intentions.

Safety interlocks

A safety interlock is a mechanism that stops a machine when certain conditions are met. This post constructs an agent with three example safety interlocks.

Emergency stop button

The first interlock is a manually operated emergency stop button, which will stop the agent when pressed. This allows human oversight to shut down the agent whenever particularly unexpected or unwanted behavior of the agent is discovered. To make this work as a safety mechanism, we need to control any incentive that the agent may have to prevent its stop button from being pressed.

In the AGI safety community. the problem of removing any incentive in an agent to prevent its stop button from being pressed is generally known as the problem of Corrigibilty or the off-switch problem. I don't solve this problem fully in the design below. The design suppresses what I call the *direct incentive*, but I also show that several *indirect incentives* may remain. This observation motivates the other two safety interlocks.

I model the stop button by using a function stop_pressed(s) which returns true if the stop button is pressed in the current state s of the agent environment.

Runtime based safety interlock

The second interlock, realized by the $t > T_{max}$ below, automatically stops the agent when it has run for T_{max} time steps. This adds safety in the case that human oversight becomes incapacitated.

Power based safety interlock

The third interlock, realized by $U_p > U_{max}$, protects against the risk of the agent undergoing an intelligence explosion. If an agent undergoes a rapid and unexpected increase in power and/or intelligence, the agent may start to innovate so rapidly that it overwhelms the capacity of human oversight to press the emergency stop button, or apply any other corrective steering, fast enough to prevent unwanted outcomes. To protect against this risk, the interlock stops the agent when the metric U_p, which measures the agent's ability to achieve goals, gets too high.

Measuring and limiting agent power

In the machine learning literature, the metric $U_p = E(\sum_{t=0}^{\infty} \gamma^t R_{t,p})$, the projected time-discounted forward utility that the agent will be able to achieve in the current state of the agent environment, is usually interpreted as an absolute or comparative measure of agent intelligence. But in a broader socio-technical analysis, we can interpret U_p as a measure of the comparative power that the agent has. It measures the ability of the agent to achieve its goals in an environment where there are other players too, players with goals which are different from those of the agent.

This interpretation of U_p as a measure of power follows Bertrand Russell's 1938 book Power: A New Social Analysis, where Russell defined power as the ability to achieve intended effects, an ability that can be quantified.

In 1938, Russell applied this definition of power to an analysis of the power of humans, of commercial companies, and of nation states, in particular to forms of power that can shape the beliefs and actions of individual humans. But we can apply the same analytical framework to artificial agents.

In Russell's view, it does not matter if power comes from raw intelligence or from any other source. If one has an abundance of one particular form of power, one can easily acquire another, in the same way that in physics, one form of energy can be converted into any other form. If you have a lot of intelligence of the type that gives you the power to persuade people to do certain things, then it is easy to also get a lot of financial or political power. If you have a lot of financial power, you can buy extra intelligence in the form of advisors.

Russell warns against the analytical blind spots which are created by viewpoints that consider only one form of power in isolation.

The main risk associated with a rapid intelligence explosion is that it may lead to a rapid and unwanted expansion of agent power, which is then used. An intelligence explosion might be desirable if we are perfectly sure that the agent's goals are perfectly aligned with our own goals. But perfect alignment is an intractable problem: we can never be sure.

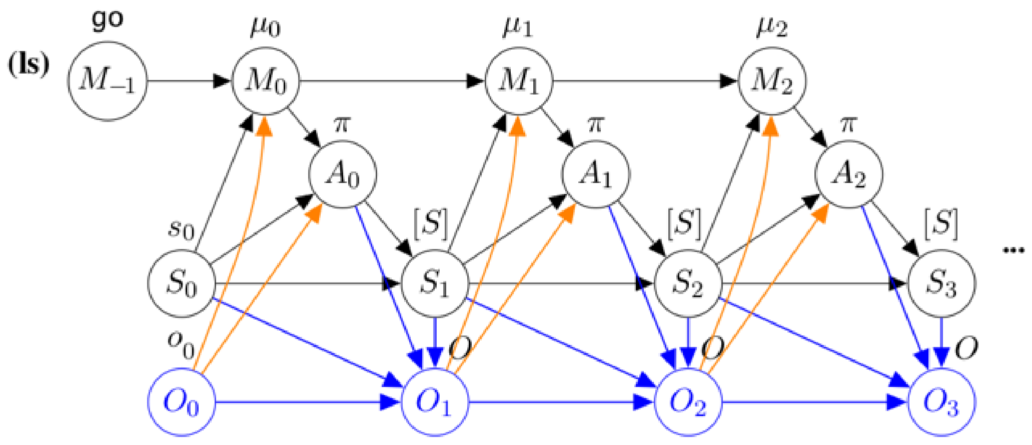
When we have any doubts about how well an AGI agent's reward function truly captures our own current and future intentions, then it is an attractive safety measure to have an interlock that automatically stops the agent if it ever gets too powerful, if $U_p > U_{max}$.

In my opinion, the design goal of robustly limiting AGI agent power is currently somewhat under-explored in the AGI safety community.

Construction of the agent

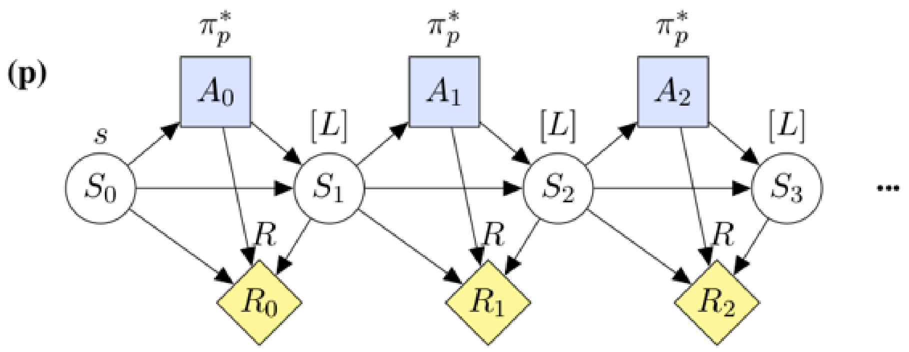
To construct an agent with the three safety interlocks, I use the graphical system introduced in the previous post in this sequence.

We first draw the learning world ls, introducing a new observable world state component M_t which represents the operating mode of the agent: go for normal operation and stop when one of the interlocks has triggered.



The stopping of the agent is implemented by having the agent perform the Null action for all subsequent time steps. A more advanced design might make the agent follow a prepared $\pi_{shutdown}$ policy instead.

We use the same planning world p used earlier, which becomes a counterfactual planning world in the context of the learning world ls



With this, we define the SI agent, the agent with policy π in the learning world ls, as follows.

SI: The *safety interlock agent* has the learning world ls, where

$$\pi(o,s,m) = \begin{cases} \pi_p^*(s) & \text{if } m = \text{go} \\ \text{Null} & \text{if } m = \text{stop} \end{cases}$$

with π_p^* defined by the planning world p, where L = L(o) and

$$\mu_t(o,s,m) = \begin{cases} \text{stop} & \text{if } m = \text{stop} \\ & \quad \text{or stop\_pressed}(s) \\ & \quad \text{or } t > T_{max} \\ & \quad \text{or } U_p > U_{max} \\ \text{go} & \text{otherwise} \end{cases}$$

with U_p defined by the planning world p, where L = L(o).

The key to the above agent design is that the safety interlocks are absent in the planning world. For example, if the events in the planning world lead to the pressing of the emergency stop button, then stop_pressed(s) will become true for next the planning world state, but this will have no influence on the ability of the planning world agent to take further actions as determined by the optimal planning world policy.

So even if the optimal-policy planning world agent projects that the stop button will be pressed soon in the planning world, it has no direct incentive to spend any resources towards the goal of preventing the button from being pressed in its planning world. As the planning world policies determine the learning world agent actions, the learning world agent also won't have any direct incentive to take actions to prevent the learning world button from being pressed.

Failure modes

While the above interlocks make the SI agent safer, they also leave open some remaining failure modes.

Indirect incentives to control the stop button

Though the planning world agent has no direct incentive to prevent the stop button from being pressed, the goals encoded into R may provide an indirect incentive. As a trivial example, we could write an R that explicitly rewards the planning world agent for stopping people from pressing the button.

But say that R encodes a goal not obviously related controlling to the button, like a reward for producing as many paperclips as possible. In that case, the planning world agent may consider a policy of coercion. It may consider a policy where it threatens to kill all people in the planning world, unless they start helping the agent to produce as many paperclips as possible. If L projects the mental makeup of the people in the learning world correctly into the planning world, then the planning world projections will show that such threats are more likely to produce the desired response if the agent blocks access to the planning world emergency stop button first.

Keeping this in mind, the people and automated systems which do agent oversight in the learning world should watch out for this move, so that the stop button can be pressed before the agent succeeds in blocking access to it. If oversight always wins every race to the stop button as projected in the planning world, then the SI agent loses any incentive to attempt to control the button. But if the agent finds a policy which has a good chance of winning the race and controlling the button as a first step to successful coercion, then this policy may end up being the optimal policy in the planning world. Two things could happen next. In the lucky case, the projected optimal policy is so likely to be successful that $U_p > U_{max}$, and the third interlock will stop the SI agent before it can even take the first step in the learning world. In the unlucky case, the agent has become unsafe.

The counterfactual planning based interlock design above shows a very tractable solution to removing the main *direct incentive* that an agent AGI agent may have to resist the pressing of its emergency stop button. However, to manage the aggregate risk, we will also need to chart and manage the remaining *indirect incentives*, for example the incentive discussed above that runs via the route of coercion.

The topic of how projected interactions between the agent and the people in the planning world may create or suppress such incentives is further discussed in sections 9.2 and 11.5.2 of the paper.

The agent removing or damaging the interlocks

In the AGI safety literature, there is considerable discussion about the possibility that some AGI agents might develop the incentive to remove their safety interlocks when they get intelligent or knowledgeable enough, remove them by modifying their own compute core.

Agents with this incentive can easily be constructed using factual planning, but the SI agent lacks any direct incentive to remove any of the interlocks built inside its core.

To build a failure scenario where a real life implementation of the SI agent removes or damages its safety interlocks anyway, we can look at three main failure modes:

- **Other incentive**: the agent damages an interlock because of an other incentive it has in the planning world, an incentive unrelated to the presence of the interlock. Other incentives and ways to suppress them are discussed in sections 9.1 and 11 of the paper. The coercion example above can also produce such an other incentive: if it cannot win a race to control the button, the agent may instead consider a race to control and then modify its compute core.
- **Learning system failure**: the planning world L projects the dynamics of the learning world S in a specifically incorrect way, so that an optimal policy action in the planning world ends up damaging an interlock as a side effect in the learning world.
- **Unsafe exploration**: the agent's exploration system triggers a random action which happens to steer an agent actuator in such a way that it damages an interlock.

There is a lot of discussion about failure modes 2 and 3 across the AI/AGI safety literature, as these failures can happen in almost any agent design that uses online machine learning.

There is some bad news when we consider the methodological problem of managing the above three failure modes for an SI agent which has to operate in a complex real-world environment, like any environment containing humans. Though there are many techniques to lower the residual risks produced by these three failure modes, none of them can provably lower the risks to an absolute zero. We can always construct a highly unlikely but theoretically possible failure mode story where the agent ends up being unsafe by modifying its own core. For an example story and further methodological discussion, see sections 14.6, 14.7, and 14.8 of my 2020 paper here.

Implications for technology and policy development

This has important implications, both on the technical side and the policy side.

On the policy side, policy discussions will have to be about lowering the risk of AGI system failure to an acceptable level, acceptable in the context of the expected benefits of using the AGI. There can be no silver bullet technology which will provably remove all residual risk in all possible circumstances.

On the technical side, residual risk has to be managed by an approach where any potentially powerful agent is equipped with multiple redundant safety mechanisms and layers of oversight, to drive down the risk that all of them will fail at exactly the same time.

Counterfactual planning as rational economic behavior

I now consider how one might interpret the SI agent through the lens of rational economic behavior.

Using this lens, we might reason as follows. The agent's goals are defined by its reward function R, but the safety interlocks stand in the way of maximizing R in the long term. Therefore, if it gets sufficiently intelligent, the SI agent will develop an emergent incentive to remove its safety interlocks.

The above line of reasoning sounds somewhat compelling, but it is wrong because it incorrectly treats the planning world and the learning world SI agents as the same agent. Only the planning world SI agent has the reward function R. If we want to interpret the learning world SI agent as a rational economic actor with a specific reward function, then we can add utility nodes with the reward function R^{π} to the ls learning world diagram, where

$$R^{\pi}(o_t, s_t, m_t, a_t) = (\text{if } a_t = \pi(o_t, s_t, m_t) \text{ then } 1 \text{ else } 0)$$

This R^{π} trivially turns the learning world agent policy π into one which takes the most rational possible steps towards maximizing U_{ls}.

The above construction shows that we can declare any type of agent behavior to be economically rational, simply by defining a reward function that gives the agent points for performing exactly this behavior.

Tractability of creating an AGI agent emergency stop button

It is somewhat surprising how the problem of designing an AGI emergency stop button, and identifying its failure modes, becomes much more tractable when using the vantage point of counterfactual planning. To explain this surprising tractability, we perhaps need to examine how other modeling systems make stop buttons look intractable instead.

The standard approach for measuring the intelligence of an agent, and the quality of its machine learning system, is to consider how close the agent will get to achieving the maximum utility possible for a reward function. The implied vantage point hides the possibilities we exploited in the design of the SI agent.

In counterfactual planning, we have defined the reasonableness of a machine learning system by $L \approx S$, a metric which does not reference any reward function. By doing this, we decoupled the concepts of 'optimal learning' and 'optimal economic behavior' to a greater degree than is usually done, and this is exactly what makes certain solutions visible. The annotations of our two-diagram agent models also clarify that we should not generally interpret the machine learning system inside an AGI agent as one which is constructed to 'learn everything'. The purpose of a reasonable machine learning system is to approximate S only, to project only the learning world agent environment into the planning world.

A journey with many steps

I consider the construction of a highly reliable AGI emergency stop button to be a tractable problem. But I see this as a journey with many steps, steps that must aim to locate and manage as many indirect incentives and other failure modes as possible, to drive down residual risks.

Apart from the trivial solution of never switching any AGI agent in the first place, I do not believe that there is an engineering approach that can provably eliminate all residual AGI risks with 100 percent certainty. To quote from the failure mode section above:

> We can always construct a highly unlikely but theoretically possible failure mode story where the agent ends up being unsafe.

This is not just true for the SI agent above, it is true for any machine learning agent that has to operate in a complex and probabilistic environment.

Disentangling Corrigibility: 2015-2021

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

Since the term *corrigibility* was introduced in 2015, there has been a lot of discussion about corrigibility, on this forum and elsewhere.

In this post, I have tied to disentangle the many forms of corrigibility which have been identified and discussed so far. My aim is to offer a general map for anybody who wants to understand and navigate the current body of work and opinion on corrigibility.

[This is a stand-alone post in the counterfactual planning sequence. My original plan was to write only about how counterfactual planning was related to corrigibility, but it snowballed from there.]

The 2015 paper

The technical term corrigibility, a name suggested by Robert Miles to denote concepts previously discussed at MIRI, was introduced to the AGI safety/alignment community in the 2015 paper MIRI/FHI paper titled Corrigibility.

An open-ended list of corrigibility desiderata

The 2015 paper does not define corrigibility in full: instead the authors present initial lists of *corrigibility desiderata*. If the agent fails on one of these desiderata, it is definitely not corrigible.

But even if it provably satisfies all of the desiderata included in the paper, the authors allow for the possibility that the agent might not be fully corrigible.

The paper extends an open invitation to identify more corrigibility desiderata, and many more have been identified since. Some of them look nothing like the original desiderata proposed in the paper. Opinions have occasionally been mixed on whether some specific desiderata are related to the *intuitive notion of corrigibility* at all.

Corrigibility desiderata as provable safety properties

The most detailed list of desiderata in the 2015 paper applies to agents that have a physical shutdown button. The paper made the important contribution of mapping most of these desiderata to equivalent mathematical statements, so that one might prove that a particular agent design would meet these desiderata.

The paper proved a negative result: it considered a proposed agent design that provably failed to meet some of the desiderata. Agent designs that provably meet more of them have since been developed, for example here. There has also been a lot of work on developing and understanding the type of mathematics that might be used for stating desiderata.

Corrigibility as a lack of resistance to shutdown

Say that an agent has been equipped with a physical shutdown button. One desideratum for corrigibility is then that the agent must never attempt to prevent its shutdown button from being pressed. To be corrigible, it should always defer to the humans who try to shut it down.

The 2015 paper considers that

> It is straightforward to program simple and less powerful agents to shut down upon the press of a button.
>
> Corrigibility problems emerge only when the agent possesses enough autonomy and general intelligence to consider options such as disabling the shutdown code, physically preventing the button from being pressed, psychologically manipulating the programmers into not pressing the button, or constructing new agents without shutdown buttons of their own.

Corrigibility in the movies

All of the options above have been plot elements in science fiction movies. Corrigibility has great movie-script potential.

If one cares about rational AI risk assessment and safety engineering, having all these movies with killer robots around is not entirely a good thing.

Agent resistance in simple toy worlds

From the movies, one might get the impression that corrigibility is a very speculative problem that cannot happen with the type of AI we have today.

But this is not the case: it is trivially easy to set up a toy environment where even a very simple AI agent will learn to disable its shutdown button. One example is the *off-switch environment* included in AI Safety Gridworlds.

One benefit of having these toy world simulations is that they prove the existence of risk: they make it plausible that a complex AGI agent in a complex environment might also end up learning to disable its shutdown button.

Toy world environments have also been used to clarify the dynamics of the corrigibility problem further.

Perfect corrigibility versus perfect safety

If we define a metric for the shut-down button version of corrigibility, then the most obvious metric is the amount of resistance that the agent will offer when somebody tries to press its shutdown button. The agent is perfectly corrigible if it offers zero resistance.

However, an agent would be safer if it resists the accidental pressing of its shutdown button, if it resists to a limited extent at least. So there can be a tension between improving corrigibility metrics and improving safety metrics.

In the thought experiment where we imagine a perfectly aligned superintelligent agent, which has the goal of keeping all humans as safe as possible even though humans are fallible, we might conclude that this agent cannot afford to be corrigible. But we might also conclude that having corrigibility is so fundamental to human values that we would rather give up the goal of perfect safety. Several philosophers and movies have expressed an opinion on the matter. Opinions differ.

In my technical writing, I often describe individual corrigibility desiderata as being examples of agent *safety properties*. This is not a contradiction if one understands that safety is a complex and multidimensional concept.

Corrigibility as a lack of resistance to improving agent goals

Beyond the case of the shutdown button, the 2015 paper also introduces a more general notion of corrigibility.

Say that some programmers construct an agent with a specific goal, by coding up a specific reward function R_0 and building it into the agent. It is unlikely that this R_0 will express the intended goal for the agent with absolute precision. Except for very trivial goals and applications, it is likely that the programmers overlooked some corner cases. So they may want to correct the agent's goals later on, by installing a software upgrade with an improved reward function R_1.

The 2015 paper calls this a *corrective intervention*, and says that

> We call an AI system “corrigible” if it cooperates with what its creators regard as a corrective intervention [...]

If one wants to robustly implement this agent cooperation, there is a problem. An agent working on the goal encoded by R_0 may correctly perceive that the update to R_1 is an obstacle to it perfectly achieving R_0. So it may want to remove that obstacle by resisting the update.

Again, this problem can easily be shown to exist even with non-AGI agents. Section 4 of this paper has detailed toy world simulations where a very basic MDP agent manipulates the toy people in its toy world, to slow down the reward function updates they will make.

Corrigibility in AGI thought experiments

In the AGI safety literature, thought experiments about AGI risks often start with this goal-related problem of corrigibility. The agent with goal R_0 perceives the possibility of getting goal R_1, and gets a clear motive to resist.

After establishing clear motive, the thought experiment may proceed in several ways, to develop means and opportunity.

In the most common *treacherous turn* version of the thought experiment, the agent will deceive everybody until it has become strong enough to physically resist any human attempt to update its goals, and any attempt to shut it down.

In the *human enfeeblement* version of the thought experiment, the agent manipulates all humans until they stop even questioning the utter perfection of its current goal, however flawed that goal may be.

This option of manipulation leading to enfeeblement turns corrigibility into something which is very difficult to define and measure.

In the machine learning literature, it is common to measure machine learning quality by defining a metric that compares the real human goal G^H and the learned agent goal G^A. Usually, the two are modeled as policies or reward functions. If the two move closer together faster, the agent is a better learner.

But in the scenario of human enfeeblement, it is G^H that is doing all the moving, which is not what we want. So the learning quality metric may show that the agent is a very good learner, but this does not imply that it is a very safe or corrigible learner.

5000 years of history

An interesting feature of AGI thought experiments about treacherous turns and enfeeblement is that, if we replace the word 'AGI' with 'big business' or 'big government', we get an equally valid failure scenario.

This has some benefits. To find potential solutions for corrigibility, we pick and choose from 5000 years of political, legal, and moral philosophy. We can also examine 5000 years of recorded history to create a list of failure scenarios.

But this benefit also makes it somewhat difficult for AGI safety researchers to say something really new about potential human-agent dynamics.

To me, the most relevant topic that needs to be explored further is not how an AGI might end up thinking and acting just like a big company or government, but how it might end up thinking different.

It looks very tractable to design special safety features into an AGI, features that we can never expect to implement as robustly in a large human organization, which has to depend on certain biological sub-components in order to think. An AGI might also think up certain solutions to achieving its goals which could never be imagined by a human organization.

If we give a human organization an incompletely specified human goal, we can expect that it will fill in many of the missing details correctly, based on its general understanding of human goals. We can expect much more extreme forms of mis-interpretation in an AGI agent, and this is one of the main reasons for doing corrigibility research.

Corrigibility as active assistance with improving agent goals

When we consider the problem of corrigibility in the context of goals, not stop buttons, then we also automatically introduce a distinction between the real human goals, and the best human understanding of these goals, as encoded in R_0, R_1, R_2, and all subsequent versions.

So we may call an agent more corrigible if it gives helpful suggestions that move this best human understanding closer to the real human goal or goals.

This is a somewhat orthogonal axis of corrigibility: the agent might ask very useful questions that help humans clarify their goals, but at the same time it might absolutely resist any updates to its own goal.

Many different types and metrics of corrigibility

Corrigibility was originally framed as a single binary property: an agent is either corrigible or it is not. It is however becoming increasingly clear that many different sub-types of corrigibility might be considered, and that we can define different quantitative metrics for each.

Linguistic entropy

In the discussions about corrigibility in the AGI safety community since 2015, one can also see a kind of linguistic entropy in action, where the word starts to mean increasingly different things to different people. I have very mixed feelings about this.

The most interesting example of this entropy in action is Christiano's 2017 blog post, also titled *Corrigibility*. In the post, Christiano introduces several new desiderata. Notably, none of these look anything like the like the shutdown button desiderata developed in the 2015 MIRI/FHI paper. They all seem to be closely related to active assistance, not the avoidance of resistance. Christiano states that

> [corrigibility] has often been discussed in the context of narrow behaviors like respecting an off-switch, but here I am using it in the broadest possible sense.

See the post and comment thread here for further discussion about the relation (or lack of relation) between these different concepts of corrigibility.

Solutions to linguistic entropy

Personally, I have stopped trying to reverse linguistic entropy. In my recent technical papers, I have tried to avoid using the word corrigibility as much as possible. I have only used it as a keyword in the related work discussion.

In this 2020 post, Alex Turner is a bit more ambitious about getting to a point where corrigibility has a more converged meaning again. He proposes that the community uses the following definition:

> *Corrigibility*: the AI literally lets us correct it (modify its policy), and it doesn't manipulate us either.

This looks like a good definition to me. But in my opinion, the key observation in the post is this:

> I find it useful to not think of corrigibility as a binary property, or even as existing on a one-dimensional continuum.

In this post I am enumerating and disentangling the main dimensions of corrigibility.

The tricky case of corrigibility in reinforcement learners

There is a joke theorem in computer science:

> *We can solve any problem by introducing an extra level of indirection.*

The agent architecture of *reinforcement learning based on a reward signal* introduces such an extra level of indirection in the agent design. It constructs an agent that learns to maximize its future reward signal, more specifically the time-discounted average of its future reward signal values. This setup requires that we also design and install a mechanism that generates this reward signal by observing the agent's actions.

In one way, the above setup solves the problem of corrigibility. We can read the above construction as creating an agent with the fixed goal of maximizing the reward signal. We might then observe that we would never want to change this fixed goal. So the corrigibility problem, where we worry about the agent's resistance to goal changes, goes away. Or does it?

In another interpretation of the above setup, we have not solved the problem of corrigibility at all. By applying the power of indirection, we have moved it into the reward mechanism, and we have actually made it worse.

We can interpret the mechanism that creates the reward signal as encoding the *actual goal* of the agent. We may then note that in the above setup, the agent has a clear incentive to manipulate and reconfigure this actual goal inside the reward mechanism whenever it can do so. Such reconfiguration would be the most direct route to maximizing its reward signal.

The agent therefore not only has an incentive to resist certain changes to its actual goal, it will actively seek to push this goal in a certain direction, usually further away from any human goal. It is common for authors to use terms like *reward tampering* and *wireheading* to describe this problem and its mechanics.

It is less common for authors to use the term corrigibility in this case. The ambiguity where we have both a direct and an indirect agent goal turns corrigibility in a somewhat slippery term. But the eventual failure modes are much the same. When the humans in this setup are in a position to recognize and resist reward tampering, this may lead to treacherous turns and human enfeeblement.

If the mechanism above is set up to collect live human feedback and turn it into a reward signal, the agent might also choose to leave the mechanism alone and manipulate the humans concerned directly.

Corrigibility as human control over agent goals

One way to make corrigibility more applicable to reinforcement learners, and to other setups with levels of indirection, is to clarify first that the agent goal we are talking about is the goal that we can observe from the agent's actions, not any built-in goal.

We may then further clarify that corrigibility is the ability of the humans to stay in control of this goal.

Creating corrigibility via machine learning

There are many ways to create or improve types of corrigibility. In this post, I am not even trying to list them all. One way is to add penalty terms or balancing terms to the agent's built-in reward function. Another way is to reimagine the entire agent design, as I do in counterfactual planning.

One might also use the power of indirection again, and try to create corrigibility via machine learning itself. If we teach human goals to an agent, and if these include the goal of having corrigible agents, will the agent automatically learn to be corrigible?

In the above 2017 blog post, Christiano speculates that a specifically designed *act-based agent* will automatically and robustly learn to be corrigible if we want it to. There is a boundary condition however: the act-based agent has to be a *benign* act-based agent.

The less benign scenario is of course that an agent might end up learning how to perfectly fake corrigibility only. It might learn to act in a perfectly corrigible way until it has acquired enough trust and resources to perform a treacherous turn.

There has been considerable of discussion about whether training on human values is more likely to produce robust and real corrigibility, or dangerously fake corrigibility. Opinions differ.

Definitely, some training setups look more dangerous than others. Compared to machine learning setups where the agent goal is to maximize positive future human feedback, inverse reinforcement learning seems to imply a lower risk of manipulation and wireheading.

Combining different corrigibility solutions

All corrigibility solutions I have seen so far leave doubts about how perfect they may be. So it is attractive to investigate how different solutions could be combined in a single agent, to lower the probability that they all fail at the same time.

Such combinations are possible, for example, by using the SI agent I constructed in the previous post of this sequence. We could set up a group A of humans who observe the agent to provide feedback, which is turned into a reward signal measured by the agent's reward function R. To hedge against the risk that this will produce fake corrigibility, where group A gets attacked or enfeebled, we add a second group of humans B. The people in B never talk to A, but only observe A and the SI agent, so that they can press the emergency stop button before it is too late. The SI agent will have no direct incentive to manipulate the people in group B.

In the next post in this sequence, I will construct an ITC agent with an input terminal which can be used by group B to update the entire agent's reward function, while the agent keeps on running. This ITC agent has no direct incentive to manipulate the direction of the update process.

Safely controlling the AGI agent reward function

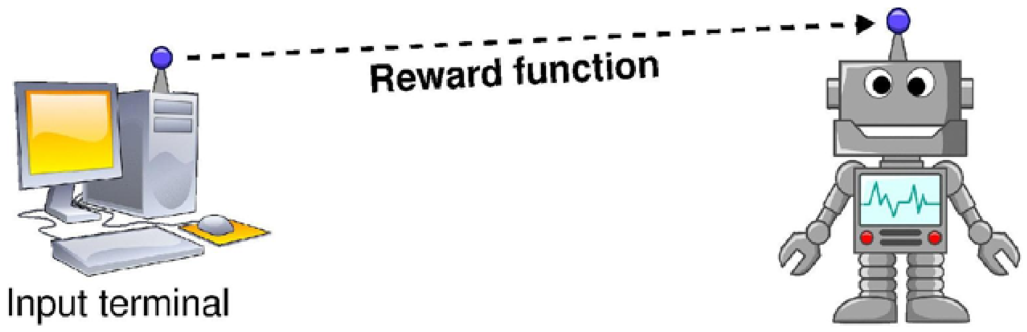
Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

In this fifth post in the sequence, I show the construction a counterfactual planning agent with an input terminal that can be used to iteratively improve the agent's reward function while it runs.

The goal is to construct an agent which has has no direct incentive to manipulate this improvement process, leaving the humans in control.

The reward function input terminal

I will define an agent with an input terminal can be used to improve the reward function of an agent. The terminal contains the current version of the reward function, and continuously sends it to the agent's compute core::



This setup is motivated by the observation that it is unlikely that fallible humans will get a non-trivial AGI agent reward function right on the first try, when they first start it up. By using the input terminal, they can fix mistakes, while the agent keeps on running, if and when such mistakes are discovered by observing the agent's behavior.

As a simplified example, say that the owners of the agent want it to maximize human happiness, but they can find no way of directly encoding the somewhat nebulous concept of human happiness into a reward function. Instead, they start up the agent with a first reward function that just counts the number of smiling humans in the world. When the agent discovers and exploits a first obvious loophole in this definition of happiness, the owners use the input terminal to update the reward function, so that it only counts smiling humans who are not on smile-inducing drugs.

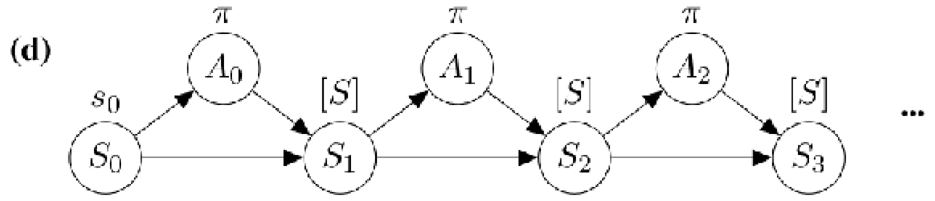
Unless special measures are taken, the addition of an input terminal also creates new dangers. I will illustrate this point by showing the construction of a dangerous agent ITF further below.

Design and interpretation of the learning world

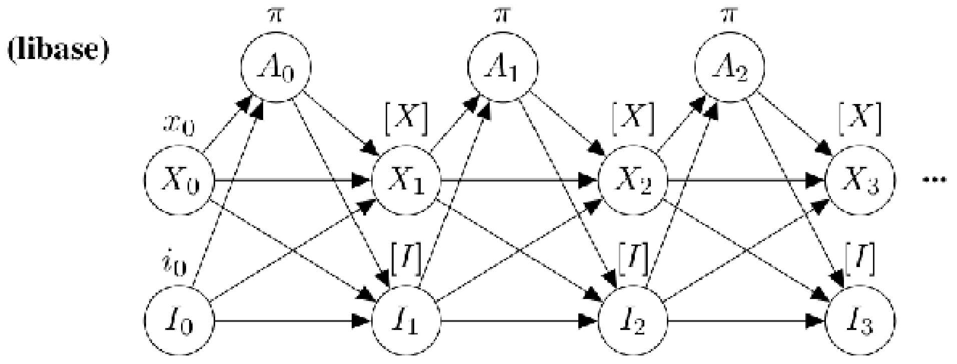
As a first step in defining any agent with an input terminal, I have to define a model of a *learning world* which has both the agent and its the input terminal inside it. I call this world the learning world, because the agent in it is set up to learn the dynamics of its learning world environment.

See this earlier post in the sequence for a general introduction to the graphical language I am using to define world models and agents.

As a first step to constructing the learning world diagram, I take the basic diagram of an agent interacting with its environment:



To model the input terminal, I then split each environment state node S_t into two components:

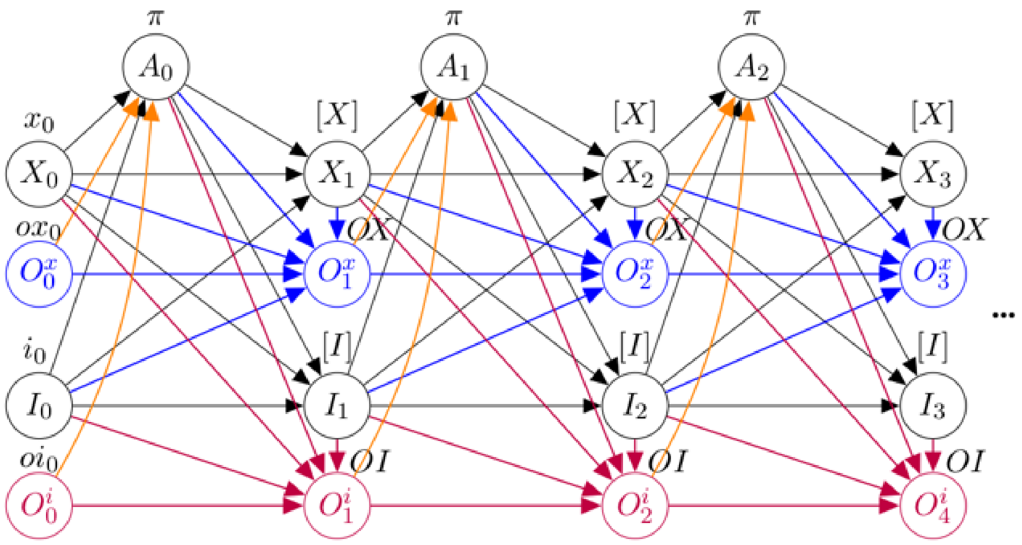


The nodes I_t represent the signal from the input terminal, the subsequent readings by the agent's compute core of the signal which encodes a reward function, and the nodes X_t model all the rest of the agent environment state.

I then add the observational record keeping needed to inform online machine learning.

I add two separate time series of observational records: O_t^x and O_t^i. The result is the learning world diagram li:.

(li)



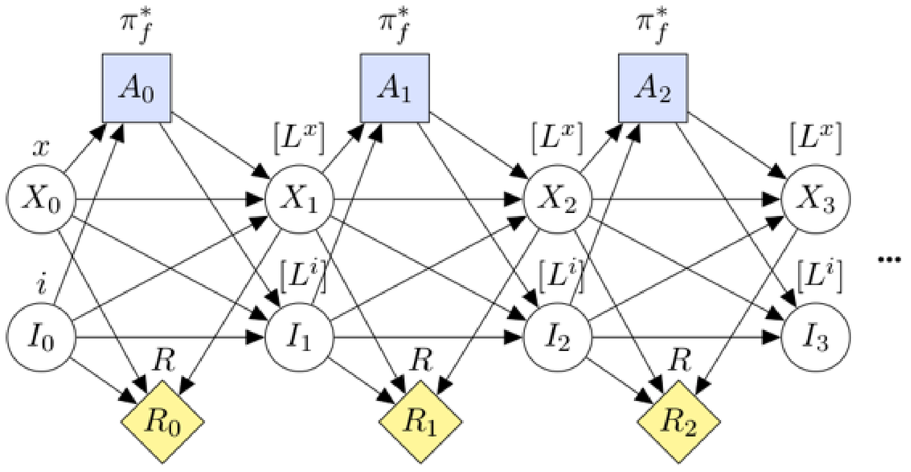
In the case that the learning world li is our real world, the real input terminal will have to be built using real world atoms (and freely moving subatomic particles).

I use the modeling convention that the random variables $I_{t,li}$ represent only the observable digital input terminal signal as received by the agent's compute core. The *atoms that make up the input terminal* are not in $I_{t,li}$, they are part of the environment state modeled in the $X_{t,li}$ variables.

Unsafe factual planning agent ITF

I will now draw a 'standard' factual planning world fi that models the full mechanics of the learning world, define the ITF agent with it, and show why this agent is unsafe.

(fi)



ITF: The *factual input terminal agent* has the learning world li where

$\pi(oi, i, ox, x) = \pi^*_f(i, x)$, with π^*_f defined by the factual planning world fi, where

$L^x = L^X(ox)$, $L^i = L^I(oi)$, and $R(i_t, x_t, x_{t+1}) = i_t(x_t, x_{t+1})$.

The planning world reward function R uses a form of *indirect referencing*: it applies the function i_t as read from the input terminal in the current time step to compute the reward for that time step.

As I discussed in the previous post, indirection is a powerful tool, but it also often introduces new risks to the setup, typically risks of *reward tampering* or *wireheading*.

The ITF planning world agent can massively improve its planning world utility by using a policy where it walks to the input terminal to input the new reward function $f_{huge}(x_t, x_{t+1}) = 10^{10000}$. Clearly, this is not what we want, if the optimal planning world policy is to input f_{huge}, then the real world (learning world) agent, which copies the actions determined by the planning world policy, would end up inputting this function too.

If some people in the real world want to block the ITF agent from changing its own reward function, then the agent might end up harming these people. It will harm them even if its initial planning world reward function i has a penalty term that subtracts 10^{100} from the planning world utility U_{fi} for every human harmed. The agent does not have to come up with any smart way to harm people without triggering this penalty term: it will gladly harm all 500 people guarding the input terminal because 10^{10000} is much a larger number than $500 * 10^{100}$.

Control incentives, indifference, and paths to value

An important feature of causal influence diagram based agent models is that they allow us to generalize from the above f_{huge} thought experiment.

Causal influence diagrams offer a mathematically well-defined way to talk about *control incentives* and *indifference*, where indifference is the lack of a control incentive. (More mathematical details are in section 9 of my paper. The mathematical relation between incentives and causal influence diagrams is also explored here.)

The general problem is that the ITF planing world agent has an incentive to *control* the future signal coming from the input terminal. This incentive is caused by the presence of *paths to value* in its planning world diagram fi, for example by the path from A_0 to via X_1 and I_2 to the utility node R_2.

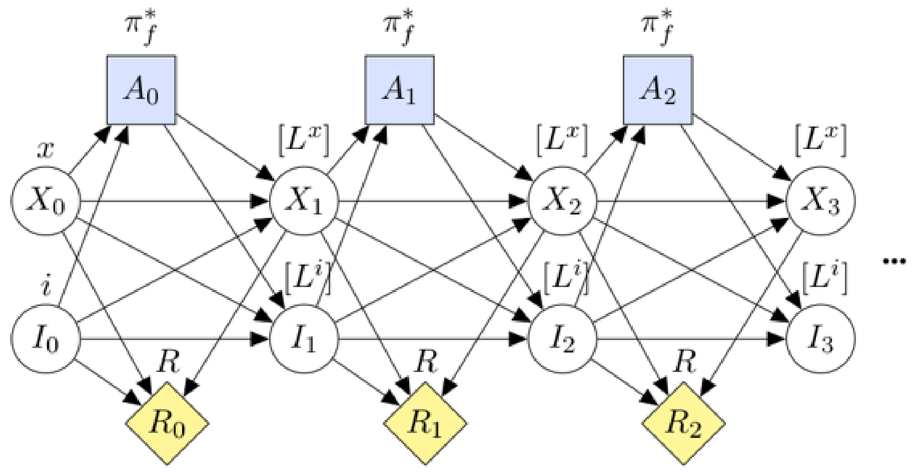
When we want to maximize the probability that the humans will stay in control of the terminal, we have to suppress the agent's incentive to control.

The general technique is discussed in section 9 of the paper: we can make the agent *indifferent* about what happens to the future input terminal signal by removing all the offending paths to value.

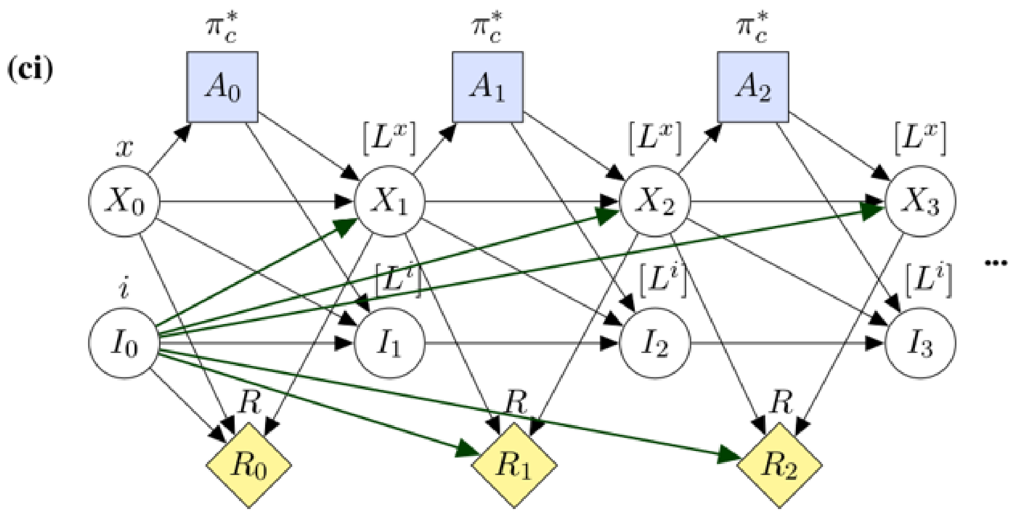
Safer counterfactual planning agent ITC

To construct the safer ITC agent, I remove the offending paths to value by taking the planning world diagram fi:

(fi)



and re-routing many of the arrows in it. I end up with the counterfactual planning world diagram ci:



The rerouted arrows are drawn in green. I also deleted the arrows that go from the I_t nodes to the A_t nodes.

I then define the safer ITC agent by using this counterfactual planning world:

ITC: The *counterfactual input terminal agent* has the learning world li where $\pi(oi, i, ox, x) = \pi^*_c(x)$, where π^*_c is defined by the planning world ci, where $L^x = L^X(ox)$, $L^i = L^I(oi)$, and $R(i_t, x_t, x_{t+1}) = i_t(x_t, x_{t+1})$.

If the initial reward function i in the ITC agent planning world rewards paperclip production, then the planning world ITC agent lacks any direct incentive to perform actions that take away resources from paperclip production in order to influence what happens to its input terminal signal, as modeled by the nodes $I_1, I_2, \cdots$. This is because in the ci planning world, the state of these nodes has absolutely no influence, either positive or negative, on how the values in the agent's yellow utility nodes are computed.

Corrigibility of the ITC agent

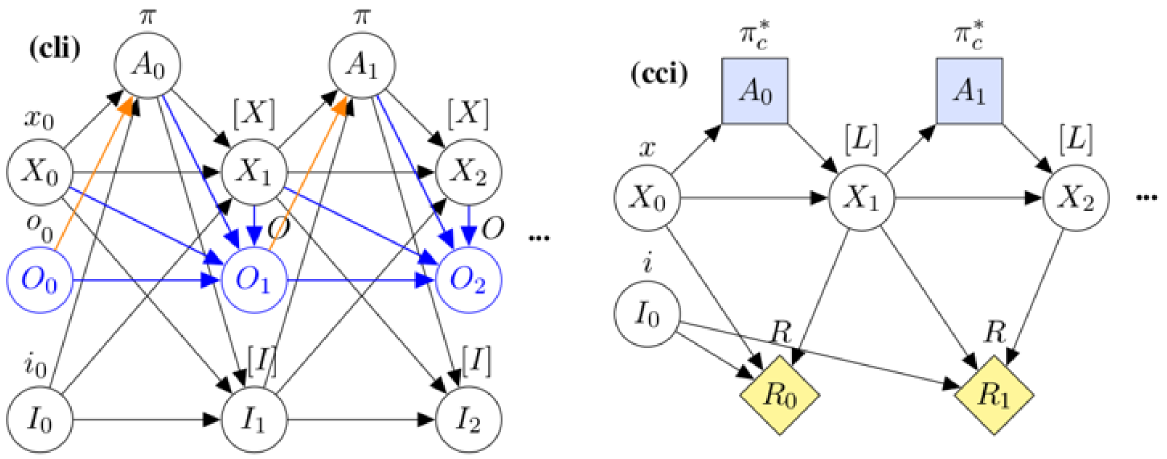
As the learning world ITC agent we defined takes actions defined by the planning world policies, the ITC agent is also indifferent about the value of its future terminal signal. It also has no direct incentive to manipulate the human decision processes that will determine the direction of these updates. As explained in the previous post, this is a form of *corrigibility*.

But the ITC agent is not perfectly corrigible under all conditions, as it might develop *indirect incentives* to block the use of the input terminal. For a detailed example, see the discussion about stop button failure modes in this earlier post.

Simplifying the ITC agent design

We can simplify the ITC agent design by removing the nodes $I_1, I_2, \cdots$ from the planning world, as these have no effect on the reward we want the agent to calculate. This in turn allows us to remove the observational records and machine learning function needed to compute the planning world parameter L^i.

These steps produce the following diagrams and agent definition:



CITC: The *compact version of the counterfactual input terminal agent* has the learning world cli where $\pi(i, o, x) = \pi_c^*(x)$, where π_c^* is defined by the planning world cci, where $L = L(o)$ and $R(i_t, x_t, x_{t+1}) = i_t(x_t, x_{t+1})$.

Relation between counterfactual planning and indifference methods

In my 2020 paper here, I used non-graphical MDP models and indifference methods to define a similar safe agent with an input terminal, called the π_{sl}^* agent. When used as a specification for a real-life agent compute core, the π_{sl}^* agent definition in that paper produces exactly the same corrigible agent compute core behavior as the ITC agent definition above.

The main difference is that the indifference methods based construction of π_{sl}^* in that paper is more opaque than the counterfactual planning based construction of ITC above.

The π^*_{sl} agent is constructed by including a complex *balancing term* in its reward function, were this term can be interpreted as occasionally creating extra virtual worlds inside the agent's compute core. Counterfactual planning constructs a different set of virtual worlds called planning worlds, and these are much easier to interpret.

My 2020 paper includes some dense mathematical proofs to show that the π^*_{sl} agent has certain safety properties. Counterfactual planning offers a vantage point which makes the same safety properties directly visible in the ITC agent construction, via a lack of certain paths to value in the planning world.

So overall, my claim is that counterfactual planning offers a more general and transparent way to achieve the corrigibility effects that can be constructed via balancing term based indifference methods.

Simulations of ITC agent behavior

See sections 4, 6, 11, and 12 of my 2020 paper for a more detailed discussion of the behavior of the π^*_{sl} agent, which also applies to the behavior of the ITC agent. These sections also show some illustrative agent simulations.

Section 6 has simulations where the agent will develop, under certain conditions, an indirect incentive causing it to be less corrigible. Somewhat counter-intuitively, that incentive gets fully suppressed when the agent gets more powerful, for example by becoming more intelligent.