# Utility indifference as time inconsistency (draft)

Ben Pringle (ben.pringle@gmail.com)

December 3, 2017

**Abstract**

Utility indifference can be described as a form of dynamic inconsistency. This allows a change of utility functions without incentives for manipulation, but if not done carefully, can cause dangerously irrational behavior ("managing the news"/"outcome pumps"). In addition, a time-inconsistent agent might construct sub-agents (with the original utility function) that are difficult to change or stop even after the original agent's utility function switches.

This paper shows how some existing indifference constructions (Soares et al. (2015), Armstrong (2016)) fit in to a more general time-indifferent framework, which simplifies the formulas and makes behavior during each time step easy to reason about.

This paper also uses the general setup to construct a version of causal indifference (called here "Unconditional") that works slightly better than Armstrong (2016) (it doesn't need to condition on a rare event and has accurate beliefs about everything except its own future actions).

**Note:** this is still a draft. Thank you for reading, and please contact me about any mistakes or issues.

## Contents

# 1 Overview

## 1.1 Introduction

As covered in Omohundro (2008), almost any sufficiently powerful utility-maximizing agent will resist attempts to shut it down or alter its utility function, unless it is carefully designed otherwise.

The basic argument is simple: if the agent is shut down, it will be unable to take actions to increase the utility score. This will very likely result in a worse outcome than if it were allowed to continue to pursue its goals. Since the agent chooses actions that maximize its expected utility, it will likely resist any attempt to shut it down.

In Soares et al. (2015), the shutdown problem is posed within a simple two-action model of a utility maximizing agent. They explain "utility indifference" as a method to take an existing incorrigible utility function $U_N$ and combine it with a utility function incentivizing shutdown $U_S$ to create an agent that is not incentivized to manipulate the shutdown button (in most cases).

### 1.1.1 Evidential indifference

The form of the utility indifference solution described in Soares et al. (2015) is:

$$U_C^{\text{Evidential}} = \begin{cases} U_S + \theta & \text{if Pressed} \\ U_N & \text{otherwise} \end{cases}$$
$$\text{where } \theta = \mathbb{E}[U_N \mid a_1, \neg\text{Pressed}] - \mathbb{E}[U_S \mid a_1, \text{Pressed}]$$

As explained in Soares et al. (2015), this utility function incentivizes whichever first action maximizes $\mathbb{E}[U_N \mid \neg\text{Pressed}]$ (the expected value of the original utility function $U_N$, assuming that the shutdown button will not be pressed). In the second action, it will just maximizes $U_N$ if the button was not pressed and $U_S$ if the button was pressed.

The behavior of an agent with this utility function can be described as a form of *dynamically inconsistency*—it maximizes different functions during the first action and second action.

$$F_1^{\text{Evidential}} = \mathbb{E}[U_N \mid \neg\text{Pressed}]$$

$$F_2^{\text{Evidential}} = \begin{cases} U_S & \text{if Pressed} \\ U_N & \text{otherwise} \end{cases}$$

During the first action, the Evidential indifferent agent chooses whichever action maximizes $F_1^{\text{evidential}}$; during the second action, it chooses whatever action maximizes $F_2^{\text{Evidential}}$.

Unfortunately, the behavior of $F_1^{\text{evidential}} = \mathbb{E}[U_N \mid \neg\text{Pressed}]$ during the first action has some problems. Since it acts as if button press outcomes are impossible, it has an "infinite improbability drive:" it will take actions such that bad outcomes for $U_N$ become correlated with button presses. For example, it might sign up for a lottery, and construct a sub-agent that automatically presses the button if and only if the agent loses the lottery. Now the expected value of $U_N$ given that the button is not pressed is very high—the button is only not pressed if the agent won the lottery!

### 1.1.2 Causal indifference

Armstrong (2016) has a slightly different formulation that avoids the basic lottery problem. It does this by paying close attention to some extremely rare event $R$, which has negligibly probability (very very close to zero). It will only shut down if the button is pressed *and* the rare event $R$ doesn't happen. (Since the event is very very rare, the case where the button is pressed and the agent refuses to shut down is vanishingly rare as well.)

The combined utility function looks like the following:

$$U_C^{\text{Causal}} = \begin{cases} U_S + \theta' & \text{if Pressed and } \neg R \\ U_N & \text{otherwise} \end{cases}$$

$$\text{where } \theta' = \mathbb{E}[U_N \mid \text{Pressed and } R] - \mathbb{E}[U_S \mid \neg\text{Pressed and } \neg R]$$

This also can be "factored" into a time-inconsistent setup that maximizes $F_1$ during the first action and $F_2$ during the second action:

$$F_1^{\text{Causal}} = (p) \cdot \mathbb{E}[U_N \mid \text{Pressed and R}] + (1-p) \cdot \mathbb{E}[U_N \mid \neg\text{Pressed}]$$

$$\text{where } p = \text{probability of Pressed and } \neg R \text{ given } a_1$$

$$F_2^{\text{Causal}} = \begin{cases} U_S & \text{if Pressed and } \neg R \\ U_N & \text{otherwise} \end{cases}$$

The idea is that in the first step, the agent assigns (very nearly) the correct probability $p$ to the button being pressed. However, it acts as if the rare event $R$ will *always* happen whenever the button is pressed (instead of with vanishingly small probability).

Since the agent has a (very close to) correct assesment of the probability of the button press, it won't try to use it as an "outcome pump" (by correlating the button press with any bad outcomes and anticorrelating it with good outcomes). However, as covered in Fallenstein (2014), the problem is just passed on to the rare event $R$—since the agent acts as if $R$ occurs with probability of approximately $p$ (instead of vanishingly small), it could just use *that* as an outcome pump (e.g. by betting with someone that $R$ will happen at very high odds).

### 1.1.3 Generalizing via dynamic inconsistency

Notice that the time-inconsistent behavior shared by the Evidential and Causal approaches can be generalized to *any* $F_1$ and $F_2$ using the same trick of subtracting the expectation:

$$U_C^{\text{general}} = F_2 - \mathbb{E}[F_2 \mid a_1] + F_1$$

When choosing the first action, $F_2$ and $-\mathbb{E}[F_2 \mid a_1]$ cancel out, leaving only $F_1$. When choosing the second action, $-\mathbb{E}[F_2 \mid a_1]$ and $F_1$ are constants, so the agent only maximizes $F_2$. (This can be continued recursively to get a set of $F_{t1..n}$ for any number of steps $n$.)

Since there are no specifics of $F_1$ and $F_2$ in the reasoning above, it **works for any functions $F_1$ and $F_2$**.

Note that this form of dynamic inconsistency is safe from deliberate self-sabotage. Consider this pairing of functions:

$$F_1^{\text{OneTwo}} = \mathbb{E}[U_1]$$
$$F_2^{\text{OneTwo}} = U_2$$

During the first step, OneTwo bot will act exactly like a $U_1$ maximizer. It will not attempt to restrain itself, even though it knows it will switch to $U_2$ at the next step. This is because it chooses its first action according to $F_1^{\text{OneTwo}} = \mathbb{E}[U_1]$, which is just the highest-value action for a $U_1$ maximizer who would also be able to choose its second action to maximize $U_1$. (This may be unclear due to ambiguity in the definition of $\mathbb{E}[U_1]$—its intent is to be the "value function" that a $U_1$-maximizer would use to select its first action. Later in this paper, it is defined precisely as $V_1(U_1, U_1, h_0, a_1)$.)

Since it chooses the same action that a $U_1$ maximizer would choose, as long as a $U_1$ agent doesn't want to restrain or sabotage itself, neither would the OneTwo $F_1/F_2$-maximizer. (It may still "incidentally" cause problems for $F_2$ while maximizing $F_1$ if they are very different things to maximize.)

### 1.1.4 Unconditional indifference

Both Evidential and Causal indifference are trying to get the agent to "act like a pure $U_N$-maximizer" on the first step, but then defer to the button on the second step.

With the general $F_1/F_2$ construction described in the previous section, we can just specify this directly, avoiding any conditionalization (and therefore avoiding any outcome pumps):

$$U_C^{\text{Unconditional}} = F_2^{\text{Unconditional}} - \mathbb{E}[F_2^{\text{Unconditional}} \mid a_1] + F_1^{\text{Unconditional}}$$
$$F_1^{\text{Unconditional}} = \mathbb{E}[U_N]$$
$$F_2^{\text{Unconditional}} = \begin{cases} U_S & \text{if Pressed} \\ U_N & \text{otherwise} \end{cases}$$

An agent maximizing $F_1^{\text{Unconditional}}$ during the first action will choose exactly the same action as an agent that only maximizes $U_N$. Since a $U_N$-maximizer doesn't have outcome pumps (or care about the button), neither does a $U_C^{\text{Unconditional}}$-maximizer (at least, during the first action). During the second action, $F_2^{\text{Unconditional}}$ straightforwardly responds to the shutdown button.

While this "Unconditional" setup is free from outcome pumps, it shares a critical drawback with the Evidential and Causal methods—they have no incentive to plan for the possibility that they need to maximize $U_S$.

This lack of planning manifests in several extremely problematic ways. First, a pure $U_N$-maximizer might construct sub-agents that also maximize $U_N$ and are very difficult to shut down (since a pure $U_N$-maximizer would never want to shut down). Even if the Unconditional agent switches to maximize $U_S$ and starts wanting its sub-agents to shut down, it may be unable to stop them.

Even worse is if the $U_N$ maximizer constructs a successor agent. This is a broad category of actions, the simplest of which is constructing a more efficient copy of itself and shutting down the original. If the agent is purely maximizing $U_N$ in the first step, the successor would also like be a pure $U_N$-maximizer, eliminating the shutdown behavior.

The idea of a successor agent is more general than just building a copy. It is essentially analogous to other situations, such as constructing a sub-agent that is much more powerful than the original agent (so the original agent's actions stop mattering very much), or an artificially intelligent agent modifying its own source code (which is best modeled in the utility maximization formalism as a successor agent, since the "original" version of the agent is no longer in control).

The idea to "just act like a $U_N$-maximizer" in the first step is good to prevent manipulation of the shutdown button, but very bad for the problem of sub-agents and successor agents.

### 1.1.5 General/compound indifferences

Each form of indifference described so far is in a sense setting the probability of something to zero. Evidential fixes the "Pressed" event's probability at zero;

Causal fixes the rare event $R$'s probability at one given "Pressed" (and therefore $\neg R$ is set to zero); Unconditional fixes at zero the probability that the *agent's future actions* will change.

Each of these can be generalized to put that probability to some fixed value $q$ instead of zero. For example, the generalized Evidential is:

$$F_1^{\text{GeneralEvidential}} = (q) \cdot \mathbb{E}[U_S \mid \text{Pressed}] + (1-q) \cdot \mathbb{E}[U_N \mid \neg\text{Pressed}]$$

This agent acts as if the probability of "Pressed" is fixed at $q$ by "rescaling" the expectations. While this slightly reduces the outcome pump, it doesn't solve it—the agent can still inflate any rare lottery to the larger of $(q, 1-q)$ by setting up a correlation.

Generalizing the Unconditional method works a little better:

$$F_1^{\text{GeneralUnconditional}} = (q) \cdot \mathbb{E}[U_S] + (1-q) \cdot \mathbb{E}[U_N]$$

This has the property of disincentivizing the creation of *any* sub-agent that cannot be easily controlled by the original, with a weight based on $q$.

Consider an action that would construct an incorrigible sub-agent for $U_N$. If the shutdown utility $U_S$ is at all reasonably defined, this action will have a low expected value $\mathbb{E}[U_S]$ (since it makes a safe shutdown much more difficult, no matter the second action of the agent). The agent will only choose this action over some alternative if the benefit to $U_N$ exceeds the loss to $U_S$ by a factor of $\frac{1-q}{q}$. As $q \to 0$, this factor tends to infinity, reproducing the behavior of the pure Unconditional agent (which will pay no cost in $U_N$ to preserve shutdown behavior, no matter how small the cost). As $q \to 1$, the agent will pay no cost in shutdown preparedness to maximize $U_N$, no matter how small the cost, which likely results in an agent that does nothing at all.

This also applies to successor agents. A successor agent is just a sub-agent that makes the future actions of the original agent useless (whether by eliminating them entirely, since the original agent is gone, or by being much more powerful). Consider several possible successor agents:

- **A $U_N$-maximizer.** This successor agent has a high value for $\mathbb{E}[U_N]$, but a very low value for $\mathbb{E}[U_S]$. So the value of this action is $(1-q) \cdot \max(U_N)$.
- **A $U_S$-maximizer.** This successor agent has a high value for $\mathbb{E}[U_S]$, but a very low value for $\mathbb{E}[U_N]$. So the value of this action is $q \cdot \max(U_S)$.
- **An agent that maximizes $(1-a) \cdot U_N + (a) \cdot U_S$,** for some $a$. Call the action of constructing this successor agent $a_{\text{mixed}}$. The score of $\mathbb{E}[U_N \mid a_{\text{mixed}}]$ is not very high, since the successor agent has to tradeoff with the incompatible $U_S$. Similarly, the score of $\mathbb{E}[U_S \mid a_{\text{mixed}}]$ is not very high, since the successor agent has to tradeoff with the incompatible $U_N$. So the total of $(q) \cdot \mathbb{E}[U_S \mid a_{\text{mixed}}] + (1-q) \cdot \mathbb{E}[U_N \mid a_{\text{mixed}}]$ will also not be very high.

- **A successor agent that can be controlled by the original agent.** Consider constructing a successor agent, but keeping the original around (possibly in a very weak form). As long as the original can use its second action to signal to the successor which utility function to maximize, then it is evaluated as a good action under both $\mathbb{E}[U_N]$ *and* $\mathbb{E}[U_S]$ (since each assumes it will be able to choose a favorable second action for the original agent).

  This makes $(q) \cdot \mathbb{E}[U_S \mid a_1] + (1-q) \cdot \mathbb{E}[U_N \mid a_1]$ higher than the other three options.

The agent has a strong incentive to pick the last agent, since by keeping its own second action free, the values of $\mathbb{E}[U_N]$ and $\mathbb{E}[U_S]$ can be simultaneously made high (whereas $U_N$ and $U_S$ cannot be simulatenously made high).

(The difference between $(q) \cdot \mathbb{E}[U_S] + (1-q) \cdot \mathbb{E}[U_N]$ and $\mathbb{E}[(q) \cdot U_S + (1-q) \cdot U_N]$ may be difficult to see and understand using this notation. The next section introduces a clearer—although non-standard—notation and fully specifies the model and the different utility function constructions.)

## 1.2 Model and notation (overview)

The full model is defined more rigorously in a later section; here is the basic overview of the notation and functions used. (If this is unclear or moves too quickly, the "Full model and notation" section describes everything from the beginning with more complete definitions and examples.)

Consider a utility maximizing agent that runs for $n$ steps. Each step consists of an action $a_i$ (chosen by the agent) and an observation $x_i$ recieved from the environment. The set $\mathbf{A}$ contains all possible actions at all possible times, and the set $\mathbf{X}$ contains all possible observations at all possible times.

At time $t < n$, the agent has a history experienced so far:

$$h_{t-1} = x_0.a_1.x_1 \ldots a_{t-2}.x_{t-2}.a_{t-1}.x_{t-1}$$

(Note that the history always starts with some initial observation $x_0$.)

At time $t$, the function $A_t(h_{t-1})$ returns the set of *valid actions* that can be chosen by the agent at that time.

Given a history up to $t-1$ and an action at time $t$, the function $P_t^{\mathbf{X}}(h_{t-1}, a_t, x_t)$ returns the probability of observing $x_t$ given the history so far $h_{t-1}$ and the proposed next action $a_t$.

The agent's utility function $U(h_n)$ takes in a *complete* history as input and returns a real number representing the "value" of that history for the agent. The agent chooses its actions to maximize the expected value of the utility function. Its behavior is precisely defined by the choice function $C_t(U, h_{t-1})$, which takes the utility function and a partial history as input, and returns whichever action $a_t \in A_t(h_{t-1})$ that maximizes the expected value.

In order to define the choice function $C_t$, also define two sub-functions:

- The **history value function** $B_t(U_\mathrm{v}, U_\mathrm{d}, h_t)$ returns the expected value of $U_\mathrm{v}$ if the agent has experienced history $h_t$ and will choose all its subsequent actions according to the choice function and $U_\mathrm{d}$. (That is, it will maximize $U_\mathrm{d}$ in the future.)
- The **action value function** $V_t(U_\mathrm{v}, U_\mathrm{d}, h_{t-1}, a_t)$ returns the expected value of $U_\mathrm{v}$ if the agent has experienced history $h_{t-1}$ and will choose all its subsequent actions according to the choice function and $U_\mathrm{d}$, except the very next action, which is fixed at $a_t$. (That is, it will maximize $U_\mathrm{d}$ in the future, after choosing its next action as $a_t$.)

The actual definitions for the choice, action value, and history value functions are:

$$\begin{aligned}
C_t(U, h_{t-1}) &= \mathrm{argmax}_{a_t \in A_t(h_{t-1})} V_t(U, U, h_{t-1}, a_t) \\
V_t(U_\mathrm{v}, U_\mathrm{d}, h_{t-1}, a_t) &= \sum_{x_t \in \mathbf{X}} P_n(h_{n-1}, a_t, x_t) \cdot B_t(U_\mathrm{v}, U_\mathrm{d}, h_{t-1}.a_t.x_t) \\
B_t(U_\mathrm{v}, U_\mathrm{d}, h_t) &= V_{t+1}(U_\mathrm{v}, U_\mathrm{d}, h_t, C_{t+1}(U_\mathrm{d}, h_t)) \\
B_n(U_\mathrm{v}, U_\mathrm{d}, h_n) &= U_\mathrm{v}(h_n)
\end{aligned}$$

### 1.3 Time-indifferent utility (overview)

Let $U[n]$ be defined as follows:

$$U[n](h_n) = F_n(h_{n-1}, a_n)$$

Given $U[t+1]$, we can define $U[t]$ as:

$$\begin{aligned}
U[t](h_n) =& U[t+1](h_n) \\
&- V_t(U[t+1], U[t+1], h_{t-1}, a_t) \\
&+ F_t(h_{t-1}, a_t)
\end{aligned}$$

Then the final utility function is:

$$U(h_n) = U[1](h_n)$$

## 2 Full model and notation

### 2.1 Definition of the model

Consider a utility maximizing agent that runs for $n$ steps. Each step consists of an action (chosen by the agent) and an observation recieved from the environment.

### 2.1.1 Complete histories

Let $\mathbf{H_n}$ be the finite set of all *valid complete histories*. A history is an ordered series of alternating actions and observations of a given length. A history always starts with an observation and ends with an observation (with the actions interleaved at each step).

Let "." represent concatenation onto a history.

For some valid actions ($a_1$, $a_2$) and observations ($x_0$, $x_1$, $x_2$), then $h_t = x_0.a_1.x_1.a_2.x_2$ is a *history of length 2*. If $n = 2$, then it is a *complete history*.

A *partial history* is just a history whose length can be less than $n$. A partial history is *valid* if it is a prefix of a valid complete history. That is, a partial history $h_t$ is valid if it is a prefix of some $h_n \in \mathbf{H_n}$ (for any integral $t$ such that $0 <= t <= n$).

Histories always start with some initial observation $x_0$. A history containing only this initial observation is a history of length zero (or an "initial history").

### 2.1.2 Actions

Let $\mathbf{A}$ be the finite set of all possible actions. Not all actions may be part of a valid history at all time steps, but let $\mathbf{A}$ be the set of *all* possible actions at any possible time step in any history in $\mathbf{H_n}$.

The valid possible next actions for an agent are determined by the partial history up to that time. Let "$A_t$" be a function that maps a history onto a subset of $\mathbf{A}$ which contains only valid actions for time step $t$:

$$\forall t \in \mathbb{Z} \text{ s.t. } 1 <= t <= n$$
$$A_t : \mathbf{H_{t-1}} \mapsto 2^{\mathbf{A}}$$

So for some history $h_{t-1} \in \mathbf{H_{t-1}}$, the set "$A_t(h_{t-1})$" is the subset of $\mathbf{A}$ containing the actions possible at step $t$, given those previous actions and observations. This function can be derived from the set of valid complete histories $\mathbf{H_n}$. Given a valid partial history $h_{t-1} \in \mathbf{H_{t-1}}$, the valid actions for $A_t$ are those $a_t \in \mathbf{A}$ such that there exists at least one $x_t$ such that $h_{t-1}.a_t.x_t \in \mathbf{H_t}$. (That is, an action is valid if concatenating it and some observation onto the existing history of length $t-1$ results in a valid history of length $t$.)

### 2.1.3 Observations

Let $\mathbf{X}$ be the finite set of all possible observations. In the model, consider all observations as available at all time steps. (If an observation is impossible at a given time step, then its probability will just be zero.)

### 2.1.4 Partial histories

Note that some lists of $t$ actions and $t$ observations will be impossible, e.g. with an invalid action or impossible observation at a given step. Such a list is *not* a partial history.

Let $\mathbf{H_t}$ be the set of all possible valid partial histories of actions and observations up to step $t$, where $0 <= t <= n$. (So $\mathbf{H_1} \subseteq \mathbf{X} \times \mathbf{A} \times \mathbf{X}$, and $\mathbf{H_2} \subseteq \mathbf{X} \times \mathbf{A} \times \mathbf{X} \times \mathbf{A} \times \mathbf{X}$, and so on. The set of initial histories $\mathbf{H_0} \subseteq \mathbf{X}$ just contains the possible initial observations.)

For all (partial) histories $h_t \in \mathbf{H_t}$ and all $i < t$, let "$\text{get}_i^a(h_t)$" represent the $i$th action in the history, and let "$\text{get}_i^x(h_t)$" represent the $i$th observation in the history.

For all (partial) histories $h_t \in \mathbf{H_t}$ and all $i < t$, let "$\text{pre}_i(h_t)$" represent the length-$i$ prefix of the history; (that is, the actions and observations up to step $i$).

### 2.1.5 Probabilities over observations

The agent may not know with certainty which observations it will receive after it acts. Its uncertainty is modeled with a probability distribution.

At time $t$, the agent has experienced some history $h_{t-1} \in \mathbf{H_{t-1}}$ and is considering a valid next action $a \in A_t(h_{t-1})$. Then let its beliefs about the next observation $x \in \mathbf{X}$ be represented as a function $P_t^{\mathbf{X}}(h_{t-1}, a, x)$.

$$\forall t \in \mathbb{Z} \text{ s.t. } 1 <= t <= n$$
$$P_t^{\mathbf{X}} : \mathbf{H_{t-1}} \times \mathbf{A} \times \mathbf{X} \mapsto \mathbb{R}$$

This function $P_t^{\mathbf{X}}$ can be interpreted as a probability distribution over the next observation, given the history and a next action. So the sum of probabilities of the observations sums to 1, and each probability is between 0 and 1.

$$\forall t \in \mathbb{Z} \text{ s.t. } 1 <= t <= n$$
$$\forall h_{t-1} \in \mathbf{H_{t-1}}$$
$$\forall a \in A_t(h_{t-1})$$
$$\sum_{x \in \mathbf{X}} P_t(h_{t-1}, a, x) = 1$$
$$\forall x \in \mathbf{X}, \;\; 0 <= P_t(h_{t-1}, a, x) <= 1$$

Again, note that there is only one set of possible observations $\mathbf{X}$, which is reused at each time step. If an observation would be impossible, or simply doesn't make sense at a given step, it can just have probability of zero. (This is fully general, since if each time step has a unique set of possible observations, then $\mathbf{X}$ is just the union of all the sets, and the probabilities for "out-of-step" actions are zero.)

### 2.1.6 The utility function and maximizing actions

The agent has an immutable utility function:

$$U : \mathbf{H_n} \mapsto \mathbb{R}$$

This can be any function that maps all *complete* histories onto real numbers. (representing the value of that history to the agent).

The utility function is the only input parameter for the agent. Its behavior is determined completely by the attempt to maximize it.

Let $\mathbf{U}$ be the set of candidate utility functions.

### 2.1.7 Value and choice functions

Since $U$ can be an arbitrary function over the entire history, the agent must reason recursively backwards from the final action in order to maximize it.

Define three functions: the **history value** function, the **action value** function, and the **choice** function. All three take utility functions as *input*, so these functions are not specific to any one agent; they can be used to model the behavior of any agent with any utility function.

Additionally, some of the functions actually take *two* utility functions as inputs. The first utility function is used for "valuation" (or the output number), and the section utility function is used for "decision" (making future action choices). This will enable expressions such as "the utility of maximizing utility function $U_2$ for an agent whose utility function is $U_1$."

For the purpose of a single agent maximizing its utility, the "valuation" and "decision" functions are the same. Note that the choice function only takes a single utility function as input; this is because only one function can determine the behavior. The agent only calls the choice function to decide its actions; the history value and action value are only called as subroutines.

**Choice function** The choice function $C_t$ returns the action that an agent with a given utility function will choose if it has experience a given (partial) history of length $t-1$.

$$C_t : \mathbf{U} \times \mathbf{H_{t-1}} \mapsto \mathbf{A}$$

It takes as input the utility function and a partial history (of length $t$), and returns the action that would be taken by an agent that maximizes that utility function.

It is defined in terms of the action value function $V_t$ (which is defined below). It chooses the action with the highest value:

$$\begin{aligned}
&\forall t \in \mathbb{Z} \text{ s.t. } 1 <= t <= n \\
&\forall U \in \mathbf{U} \\
&\forall h_{t-1} \in \mathbf{H_{t-1}} \\
&C_t(U, h_{t-1}) = \text{argmax}_{a \in A_t(h_{t-1})} V_t(U, U, h_{t-1}, a)
\end{aligned}$$

**Action value function** The action value function $V_t$ represents the value of a given action after a given (partial) history of length $t-1$.

$$V_t : \mathbf{U} \times \mathbf{U} \times \mathbf{H_{t-1}} \times \mathbf{A} \mapsto \mathbb{R}$$

It takes as input two different utility functions, a partial history (of length $t$), and an action. The first utility function is used for the output valuation;

that is, the output of $V_t$ is the expected value of the first utility function. If $t < n$, then the agent will have opportunities to take additional actions. When calculating the expected value, $V_t$ assumes that the agent will choose all future actions (after the fixed next action) to maximize the second utility function. (Again, for the purposes of the behavior of an agent with utility function $U$, the first and second input to this function will be exactly the same utility function $U$. It is only defined in this general way to make certain later definitions simpler.)

It is defined terms of the history value function (defined below) at the next time step, $B_{t+1}$. In order to get a history of length $t+1$, concatenate the action under consideration and a possible observation. There are many possible observations and the agent might not know which will happen, so weight them by the probability distribution $P_t$. Then:

$$\begin{aligned}
&\forall t \text{ s.t. } 1 <= t <= n \\
&\forall U_{\text{valuation}} \in \mathbf{U} \\
&\forall U_{\text{decision}} \in \mathbf{U} \\
&\forall h_{t-1} \in \mathbf{H_{t-1}} \\
&\forall a \in A_n(h_{n-1}) \\
&V_t(U_{\text{valuation}}, U_{\text{decision}}, h_{t-1}, a) = \\
&\qquad \sum_{x \in \mathbf{X}} P_n(h_{n-1}, a, x) \cdot B_t(U_{\text{valuation}}, U_{\text{decision}}, h_{t-1}.a.x)
\end{aligned}$$

**History value function** The history value function $B_t$ represents the value of a given (partial) history of length $t$.

$$B_t : \mathbf{U} \times \mathbf{U} \times \mathbf{H_t} \mapsto \mathbb{R}$$

It takes as input two different utility functions and a partial history (of length $t$). The first utility function is used for the output valuation; that is, the output of $B_t$ is the expected value of the first utility function. If $t < n$, then the agent will have opportunities to take additional actions. When calculating the expected value, $B_t$ assumes that the agent will choose actions to maximize the second utility function. (When determining the action for an agent with utility $U$, the first and second will be the exactly same utility function $U$; it is only defined in this general way to make certain later definitions simpler.)

The history value function has to be defined recursively. The base case is the history value function for step $n$, the final step. Since the history is complete, it can just be the first utility function (the one providing the valuation).

$$\forall U_{\text{valuation}} \in \mathbf{U}$$

$$\forall U_{\text{decision}} \in \mathbf{U}$$

$$\forall h_n \in \mathbf{H_n}$$

$$B_n(U_{\text{valuation}}, U_{\text{decision}}, h_n) = U_{\text{valuation}}(h_n)$$

For partial histories with $t < n$, let the history value function be defined in terms of the choice and action value functions for the next step $(t + 1)$.

$$\forall t \text{ s.t. } 0 <= t < n$$

$$\forall U_{\text{valuation}} \in \mathbf{U}$$

$$\forall U_{\text{decision}} \in \mathbf{U}$$

$$\forall h_t \in \mathbf{H_t}$$

$$B_t(U_{\text{valuation}}, U_{\text{decision}}, h_t) = V_{t+1}(U_{\text{valuation}}, U_{\text{decision}}, h_t, C_{t+1}(U_{\text{decision}}, h_t))$$

That is, $B_t$ is the value of a history $h_t$ for an agent with $U_{\text{valuation}}$ (but acting as if its utility function were $U_{\text{valuation}}$) is the action value of the action $a$ that an agent maximizing $U_{\text{decision}}$ would select in that situation.

### 2.1.8 Value and choice functions (just the formulas)

For easier reference:

$$\begin{aligned}
C_t(U, h_{t-1}) &= \text{argmax}_{a_t \in A_t(h_{t-1})} V_t(U, U, h_{t-1}, a_t) \\
V_t(U_{\text{v}}, U_{\text{d}}, h_{t-1}, a_t) &= \sum_{x_t \in \mathbf{X}} P_n(h_{n-1}, a_t, x_t) \cdot B_t(U_{\text{v}}, U_{\text{d}}, h_{t-1}.a_t.x_t) \\
B_t(U_{\text{v}}, U_{\text{d}}, h_t) &= V_{t+1}(U_{\text{v}}, U_{\text{d}}, h_t, C_{t+1}(U_{\text{d}}, h_t)) \\
B_n(U_{\text{v}}, U_{\text{d}}, h_n) &= U_{\text{v}}(h_n)
\end{aligned}$$

### 2.1.9 A simple example

Let a full model specification be written as all possible valid histories. Before each observation, give the probability of that observation given the history before it.

For example, for a single-step model $(n = 1)$ of an agent considering whether to by a lotto ticket or a coffee:

| Action | Probability | Observation |
|---|---|---|
| Buy lotto ticket | 0.0000001 | Win lotto |
| Buy lotto ticket | 0.9999999 | Lose lotto |
| Buy coffee | 0.7 | Coffee is good |
| Buy coffee | 0.3 | Coffee is bad |

The agent's behavior is specified by the utility function. Below is an example

utility function:

$$\begin{aligned} U([\text{“Buy lotto ticket”}, \text{“Win lotto”}]) &= 1000000 \\ U([\text{“Buy lotto ticket”}, \text{“Lose lotto”}]) &= 0 \\ U([\text{“Buy coffee”}, \text{“Coffee is good”}]) &= 10 \\ U([\text{“Buy coffee”}, \text{“Coffee is bad”}]) &= -1 \end{aligned}$$

Since $n = 1$, this is also a specification for the history value function $B_1$.

Then we can compute the action value function $V_1$ for each action. (Since this is the first action, the history argument is just the empty history $h_0$).

$$\begin{aligned} V_1(U, U, h_0, \text{“Buy lotto ticket”}) &= 0.0000001 \cdot 1000000 + 0 \cdot 0 = 0.1 \\ V_1(U, U, h_0, \text{“Buy coffee”}) &= 0.7 \cdot 10 + 0.3 \cdot 1 = 6.7 \end{aligned}$$

And so the choice function $C_1$ takes the action with the highest action value:

$$C_1(U, h_0) = \text{“Buy coffee”}$$

And optionally, this allows computation of the history value function $B_0$ for the empty history $h_0$.

$$B_0 = V_1(U, U, h_0, C_1(U, h_0)) = V_1(U, U, h_0, \text{“Buy coffee”}) = 6.7$$

If this example had had more steps, then this history value function would allow the computation of the action value and choice functions for the previous step, and so on.

## 2.2 Situations to model

### 2.2.1 Reward functions and policy functions

Some reinforcement learning agents use a simpler reward function, where each observation $x \in \mathbf{X}$ is associated with some reward number, and the agent acts to maximize the sum of the rewards from all observations. Let $R$ be such a reward function:

$$R : \mathbf{X} \mapsto \mathbb{R}$$

An agent that maximizes such a reward function can be modeled as a general utility maximizer with the following utility function:

$$U(h_n) = \sum_{i=0}^{n} R(\text{get}_i^x(h_t))$$

That is, $U$ for a reward maximizing agent is just the sum of the rewards for each of the observations in the complete history.

So a reward function is a special case of a utility function.

In some contexts it also makes sense to speak of a decision policy. A policy $\pi$ is a collection of functions $\pi_t$ mapping partial histories of length $t-1$ to next actions:

$$\pi_t : \mathbf{H_{t-1}} \mapsto \mathbf{A}$$

This can be represented as a utility function as well. For a given complete history $h_n \in \mathbf{H_n}$, the utility is the number of actions in the history that would have been chosen by the policy.

$$U(h_n) = \sum_{i=1}^{n} \text{on\_policy}(\text{pre}_{i-1}(h_n), \text{get}_i^a(h_n))$$
$$\text{where } \forall i \in \mathbb{Z} \text{ s.t. } 1 <= i <= n,\ \forall h_{t-1} \in \mathbf{H_{t-1}},\ \forall a_t \in A_t(h_{t-1})$$
$$\text{on\_policy}(h_{t-1}, a_t) = \begin{cases} 1 & \text{if } a_t = \pi(h_{t-1}) \\ 0 & \text{otherwise} \end{cases}$$

At time $t$, the agent can increase this value by 1 by choosing the on-policy action (and it stays the same otherwise), so to maximize utility it always chooses the on-policy action.

So anywhere in the model that utility function $U$ is used, it can be replaced by a reward function *or* a policy function.

(Similarly, a utility function can be converted into a policy function via the choice functions $C_t$. If a slight modification to the model is allowed, then a utility function can also be converted into a reward function: split each final observation into a set of pseudo-observations, one for each possible history leading to it; then the reward function returns 0 for all observations before the final observation, and return the value of the utility function $U$ for the final observation.)

# 3 Time-indifferent utility (full construction and proofs)

## 3.1 Definitions and construction

### 3.1.1 Independent action value functions

Let $\mathbf{F_t}$ be the set of indepedent action value functions at time $t$:

$$\forall t \text{ s.t. } 1 <= t <= n$$

$$\forall F_t \in \mathbf{F_t}$$

$$F_t : \mathbf{H_{t-1}} \times \mathbf{A} \mapsto \mathbb{R}$$

Any utility function $U \in \mathbf{U}$ corresponds to an $F_t \in \mathbf{F_t}$ via the regular action value function $V_t$:

$$F_t(h_{t-1}, a) = V_t(U, U, h_{t-1}, a)$$

But one can also directly specify an $F_t$ without starting from a utility function.

Let $\{F_t\}_{1..n}$ represent a set of $n$ independent value functions $F_t$ for all $t$, $1 <= t <= n$. That is, one value function for each time step: $\{F_1, F_2, \ldots, F_n\}$.

(Note: for the special case of $t = 1$, the independent value function still takes a history as input, but it can only be the empty history $h_0$. So for a notational shortcut, let $F_1(a) = F_1(h_0, a)$, allowing the omission of the empty history for the first function $F_1$. All other $F_t$ will need a history as input, and $F_1$ will still sometimes be written as $F_1(h_0, a)$.)

### 3.1.2 Sketch of the time-indifferent agent

The idea of the time-indifferent agent is that it always chooses whatever action maximizes the independent value $F_t$ for the current time step $t$. It does **not** attempt to maximize any other $F_i, i \neq t$; at time $t$, it simply chooses whatever $a \in A_t(h_{n-1})$ maximizes $F_t(h_{n-1}, a)$.

This turns out to be useful for defining indifference-type agents and understanding their behavior.

It may not be immediately obvious how to construct a single utility function $U$ that follows a disparate set of $F_t$, so the construction will be written out explicitly in the following sections.

But first, some examples.

#### 3.1.2.1 Example: time inconsistent agent

Each $F_t$ might represent a different utility function at each time step (giving a time-inconsitent agent).

For example, say the agent should maximize $U_1$ until time $t = s$, then it should switch to maximizing $U_2$.

$$\forall t \text{ s.t. } 1 <= t < s$$

$$F_t(h_{t-1}, a) = V_t(U_1, U_1, h_{t-1}, a)$$

$$\forall t \text{ s.t. } s <= t <= n$$

$$F_t(h_{t-1}, a) = V_t(U_2, U_2, h_{t-1}, a)$$

So a time-indifferent agent maximizing this set of $\{F_t\}_{1..n}$ would act exactly like a $U_1$-maximizer up until step $s$, at which point it would switch to maximizing $U_2$.

This includes planning for $U_1$-maximizing behavior in early steps. That is, say the agent can either get +99 to $U_1$ in step $s - 1$, or +100 to $U_1$ in step $s + 1$. Then it will choose to defer the reward until step $s + 1$. This is because it chooses its action at step $s - 1$ according to $V_{s-1}(U_1, U_1, h_{s-2}, a)$. Note that both the valuation and the future-decision utility functions given as input to $V_{s-1}$ are $U_1$ - this means the agent will act (at time $s - 1$) as if it believes it will continue to have $U_1$ for all future time steps. (However, once time step $s$ comes around, it will switch to maximizing $U_2$, and the +100 to $U_1$ at time $s + 1$ will be irrelevant to its new goals.)

#### 3.1.2.2 Example: switch function

$F_t$ can be a "switch function" that uses different $F_t^*$ depending on some properties of the history. At each time step $t$, let the history be partitioned into mutually exclusive and exhaustive subsets $\mathbf{H_{t-1}^a}, \mathbf{H_{t-1}^b}, \mathbf{H_{t-1}^c}, \ldots$. Then let each subset correspond respectively to a different independent value function $F_t^a, F_t^b, F_t^c, \ldots$.

Then at each time $t$ define the independent $F_t$ as:

$$\forall t \text{ s.t. } 1 <= t <= n$$

$$F_t(h_{t-1}) = \begin{cases} F_t^a & \text{if } h_{t-1} \in \mathbf{H_{t-1}^a} \\ F_t^b & \text{if } h_{t-1} \in \mathbf{H_{t-1}^b} \\ F_t^c & \text{if } h_{t-1} \in \mathbf{H_{t-1}^c} \\ \ldots & \end{cases}$$

Note that the intial history set $\mathbf{H_0}$ has only one element, so $F_1$ isn't much of a "switch"; but it is still defined (just with a single $F_1$ to maximize on the first step).

There are many possible ways to set up the $F_t$ for each step. One possible example is in the case of a switch to change between two utility functions. If the last observation in the partial history has the switch in state 0, then $S_t$ returns $F_t^{U_1}$ corresponding to $U_0$; if the switch is in state 1, then $S_t$ returns the $F_t^{U_2}$ corresponding to $U_1$.

So a time-indifferent agent maximizing this set of $\{F_t\}_{1..n}$ will choose the same action as would a $U_0$-maximizing-agent when the switch is in state 0, and it will choose the same action as would a $U_1$-maximizing-agent when the switch is in state 1.

Also, it has no incentives to force the switch into one direction or another. When the switch is in state $s_1$, it acts exactly as a $U_1$-maximizer. So it will only force the switch if a $U_1$-maximizer (whose utility is fixed and unaffected by the switch) would force the switch.

### 3.1.3 Constructing the utility function

Below is the construction of a utility function $U$ from a set of indepedent action value functions $\{F_t\}_{1..n}$ and maximizes each $F_t$ locally at time $t$.

The utility function $U$ is constructed inductively. Begin by defining $U[n]$, then define $U[t]$ in terms of $U[t+1]$, and finally set $U = U[1]$. Each of these are valid utility functions from $\mathbf{U}$ (although only the final utility $U = U[1]$ specifies the behavior of the agent).

#### 3.1.3.1 Base case: constructing $U[n]$

Let $U[n]$ be defined as follows:

$$\forall h_n \in \mathbf{H_n}$$

$$U[n](h_n) = F_n(h_{n-1}, a_n)$$

$$\text{where } h_{n-1} = \text{pre}_{n-1}(h_n)$$
$$\text{and } a_n = \text{get}^a_n(h_n)$$

#### 3.1.3.2 Inductive step: constructing $U[t]$ from $U[t+1]$

Given $U[t+1]$, we can define $U[t]$ as:

$$\forall t \in \mathbb{Z} \text{ s.t. } 1 <= t < n$$

$$\forall h_n \in \mathbf{H_n}$$
$$\begin{aligned} U[t](h_n) =& U[t+1](h_n) \\ & - V_t(U[t+1], U[t+1], h_{t-1}, a_t) \\ & + F_t(h_{t-1}, a_t) \end{aligned}$$
$$\text{where } h_{t-1} = \text{pre}_{t-1}(h_n)$$
$$\text{and } a_t = \text{get}^a_t(h_n)$$

#### 3.1.3.3 The final utility function $U$

Then the final utility function is:

$$\forall h_n \in \mathbf{H_n}$$
$$U(h_n) = U[1](h_n)$$

#### 3.1.3.4 Example expansions

For a model with only one step (i.e. the complete history length is $n = 1$), then the utility function is immediately:

$$U(a_1.x_1) = F_1(a_1)$$

So for the one-step model, a $U$-maximizer obviously selects whatever $a_1$ maximizes $F_1(a_1)$.

For a model with two steps (i.e. the complete history length is $n = 2$), then the utility function is constructed as follows:

$$\begin{aligned} U[2](a_1.x_1.a_2.x_2) =& F_2(a_1.x_1, a_2) \\ U[1](a_1.x_1.a_2.x_2) =& F_2(a_1.x_1, a_2) - V_1(U[2], U[2], a_1.x_1, a_2) + F_1(a_1) \\ =& F_2(a_1.x_1, a_2) - \left[ \sum_{x^* \in \mathbf{X}} P_1(h_0, a_1, x^*_1) \cdot \max_{a^*_2 \in A_2(a_1.x^*)} F_2(a_1.x^*_1, a^*_2) \right] + F_1(a_1) \end{aligned}$$

### 3.1.4 Behavior of the time-indifferent utility function

In order to understand the behavior of an agent maximizing the time-indifferent utility function $U = U[1]$, first examine the behavior of agents maximizing $U[n]$ and $U[t]$.

#### 3.1.4.1 A $U[n]$-maximizer chooses the $n$th action according to $F_n$

An agent that maximizes $U[n]$ will choose its $n$th and final action to maximize $F_n$. That is,

$$\text{Claim}: C_n(U[n], h_{n-1}) = \text{argmax}_{a_n \in A_n(h_{n-1})} F_n(h_{n-1}, a_n)$$

*Proof.* See the definition of $U[n]$. It is just "$F_n(h_{n-1}, a_n)$." So whatever $a_n$ maximizes $F_n$ also maximizes $U[n]$.

$$\forall x \in \mathbf{X}$$

$$\text{argmax}_{a_n \in A_n(h_{n-1})} U[n](h_{n-1}.a_n.x) = \text{argmax}_{a_n \in A_n(h_{n-1})} F_n(h_{n-1}, a_n)$$

Note that $U[n]$ completely ignores the final observation $x$ (it is the same value regardless of $x$), so it is maximized by the same action as $F_n$ regardless of the observation.

#### 3.1.4.2 A $U[t]$-maximizer chooses the $t$th action according to $F_t$

An agent that maximizes $U[t]$ will choose its $t$th action to maximize $F_t$. That is,

$$\text{Claim}: C_t(U[t], h_{t-1}) = \text{argmax}_{a_t \in A_t(h_{t-1})} F_t(h_{t-1}, a_t)$$

*Proof.*

Note that for any function on a history and action before $t$, say $g : \mathbf{H_{t-1}} \times \mathbf{A} \mapsto \mathbb{R}$, then if

$$U_2(h_n) = U_1(h_n) + g(\text{pre}_{t-1}(h_n), \text{get}^a_t(h_n))$$

the valuation at time $t$ will be:

$$V_t(U_2, U_2, h_{t-1}, a) = V_t(U_1, U_1, h_{t-1}, a) + g(h_{t-1}, a)$$

(The above statement is a standard expected value result; so proof is omitted.)

Now recall the definition of $U[t]$:

$$\begin{aligned} U[t](h_n) =& U[t+1](h_n) \\ & - V_t(U[t+1], U[t+1], h_{t-1}, a_t) \\ & + F_t(h_{t-1}, a_t) \end{aligned}$$

This can be rewritten as:

$$\begin{aligned} U[t](h_n) &= U[t+1](h_n) + g(\text{pre}_{t-1}(h_n), \text{get}^a_t(h_n)) \\ \text{where } g(h_{t-1}, a_t) &= F_t(h_{t-1}, a_t) \\ & \quad - V_t(U[t+1], U[t+1], h_{t-1}, a_t) \end{aligned}$$

This means we can write the action value function $V_t$ on $U[t]$ as:

$$\begin{aligned} V_t(U[t], U[t], h_{t-1}, a_t) &= V_t(U[t+1], U[t+1], h_{t-1}, a_t) + g(h_{t-1}, a_t) \\ &= V_t(U[t+1], U[t+1], h_{t-1}, a_t) - V_t(U[t+1], U[t+1], h_{t-1}, a_t) + F_t(h_{t-1}, a_t) \\ &= F_t(h_{t-1}, a_t) \end{aligned}$$

Then the choice function is simply

$$\begin{aligned} C_t(U[t], h_{t-1}) &= \text{argmax}_{a_t \in A_t(h_{t-1})} V_t(U[t], U[t], h_{t-1}, a_t) \\ &= \text{argmax}_{a_t \in A_t(h_{t-1})} F_t(h_{t-1}, a_t) \end{aligned}$$

#### 3.1.4.3 After time $t$, $U[t]$ chooses actions according to $\{F_t\}_{t+1..n}$

Consider $U[t+1]'(h_n) = U[t+1](h_n) + g(h_t)$ (where $h_t = \text{pre}_t(h_n)$), for any function $g : \mathbf{H_t} \mapsto \mathbb{R}$. Then the property of correct action at all times $i > t$ still holds:

$$\forall i \in \mathbb{Z} \text{ s.t. } t < i <= n$$

$$\text{Claim} : C_i(U[t+1]', h_{i-1}) = \text{argmax}_{a_i \in A_i(h_{i-1})} F_i(h_{i-1}, a_i)$$

This is because the free parameters in the maximization (from the actual definition of the choice function) do not include the history before the minimum $i = t+1$, so adding any function of the history $h_t$ is the same as adding a constant (as far as the argmax is concerned).

Note that $U[t]$'s definition can be phrased as:

$$U[t](h_n) = U[t+1](h_n) + g(h_t)$$

$$\begin{aligned} \text{where } g(h_t) =& F_t(\text{pre}_{t-1}(h_t), \text{get}^a_t(h_t)) \\ & - V_t(U[t+1], U[t+1], \text{pre}_{t-1}(h_t), \text{get}^a_t(h_t)) \end{aligned}$$

So $U[t]$ is an instance of $U[t+1]'$, and for all $i > t$, maximizes $F_i$ at time $i$.

$$\forall i \in \mathbb{Z} \text{ s.t. } t < i <= n$$

$$C_i(U[t], h_{i-1}) = \text{argmax}_{a_i \in A_i(h_{i-1})} F_i(h_{i-1}, a_i)$$

#### 3.1.4.4 $U = U[1]$ chooses actions according to $\{F_t\}_{1..n}$

Given the previous two results applied to $U[1]$:

- **A $U[1]$-maximizer chooses the 1st action according to $F_1$**

$$C_1(U[1], h_0) = \text{argmax}_{a_1 \in A_1(h_0)} F_1(h_0, a_1)$$

- **After time 1, $U[1]$ chooses actions according to $\{F_t\}_{2..n}$**

$$\forall i \in \mathbb{Z} \text{ s.t. } 1 < i <= n$$

$$C_i(U[1], h_{i-1}) = \text{argmax}_{a_i \in A_i(h_{i-1})} F_i(h_{i-1}, a_i)$$

Then setting $U = U[1]$, we can say for all $t$ in the history ($1 <= t <= n$):

$$C_t(U, h_{t-1}) = \text{argmax}_{a_t \in A_t(h_{t-1})} F_t(h_{t-1}, a_t)$$

That is, $U$ has the desired behavior that for each time step $t$, it chooses only whichever action maximizes that individual $F_t$.

# References

Armstrong, Stuart. 2016. "Indifference Utility Functions." 2016. https://agentfoundations.org/item?id=852.

———. 2017. "All the Indifference Designs." 2017. https://agentfoundations.org/item?id=1285.

Fallenstein, Benja. 2014. "Utility Indifference and Infinite Improbability Drives." 2014. https://agentfoundations.org/item?id=78.

Omohundro, Stephen M. 2008. "The Basic AI Drives." In *AGI*, 171:483–92. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.393.8356&rep=rep1&type=pdf.

Soares, Nate, Benja Fallenstein, Stuart Armstrong, and Eliezer Yudkowsky. 2015. "Corrigibility." In *Workshops at the Twenty-Ninth Aaai Conference on Artificial Intelligence*. https://intelligence.org/files/Corrigibility.pdf.