Semantic Differences Between Utility Functions and Granular Judgements

by Sven Nilsen, 2018

In this paper I discuss semantical differences between what is usually thought of as a utility function and what is usually thought of as a granular judgement. The distinction is important to understand why stronger proof consistency criteria applies to granular judgements, while having looser value consistency criteria than utility functions. I also argue that granular judgements have similar performance to utility functions in micro-models of the world, but are more applicable in macro-models, required to transition from artificial narrow intelligence to artificial super intelligence.

Utility functions are commonly used in AI research to describe goals of agents operating within a micro-model of the world. So far, nobody has been able to successfully produce an agent architecture that surpasses human general performance in a macro-model of the world. One reason might be that utility functions as they are understood semantically is not a good solution for macro-worlds, which leads to various efforts of producing good value-learning algorithms.

The major obstacle with utility functions is that they are not very good at capturing complex preferences of agents operating in complex environments. Careful analysis of this problem has raised a lot of safety issues about autonomous super-intelligent agents. Reducing the values of humans into a consistent utility function seems to be a computationally intractable problem. Therefore, in theories of what might be good value-learning algorithms, it is common to replace utility functions with some other representation that capture aspects of utility functions while having better learning-properties.

The theory of ethics as rational reasoning with granular judgements about the world, is very similar to utility functions when applied to individual experiments. This implies that granular judgements have similar performance in micro-models of the world. In fact, they are so similar that one can view granular judgements as partial functions of the whole utility function.

For example (\`E\` stands for ethical judgement):

$$E_{peter}(x_0 : [\text{loves(clara, peter)}]\ \text{true}) > E_{peter}(x_1 : [\text{loves(clara, john)}]\ \text{true})$$

Peter thinks a world where Clara loves him is better than a world where Clara loves John.

In a micro-model of the world, one might program Peter’s utility function like this:

```
$U_{peter}$ := \(x : world) =
    if loves(clara, peter, x) { 10 }
    else if loves(clara, john, x) { -5 }
    else { 0 }
```

$U_{peter}(x_0 : [\text{loves(clara, peter)}]\ \text{true}) = 10$
$U_{peter}(x_1 : [\text{loves(clara, john)}]\ \text{true}) = -5$
$U_{peter}(x_0 : [\text{loves(clara, peter)}]\ \text{true}) > U_{peter}(x_1 : [\text{loves(clara, john)}]\ \text{true})$

In the example above, it is obvious that one could replace `$E_{peter}$` with `$U_{peter}$` with no visible difference in Peter’s behavior. Unlike most approaches to value-learning, granular judgements seems like a direct alternative representation of utility functions, at least at first sight.

This leads to the question: Is there any semantic difference between utility functions and granular judgements? If there was no difference, one would not expect granular judgements to perform better than utility functions in macro-models of the world, because these two concepts would be identical.

The answers is yes, there are actually several differences:

1. A utility function might be arbitrary while a granular judgement is not
2. A utility function does not guide prior estimates while a granular judgement does
3. A utility function tends to be closed while a granular judgement tends to be open
4. A utility function is unique while a granular judgement is usually not unique
5. A utility function is usually absolute while a granular judgement is usually a constraint
6. A utility function is unambiguous while a granular judgement might depend on use of language

One important aspect of macro-models of the world is that these differences between utility functions and granular judgements might matter. To explain why there can be such properties, I will give a thought experiment, then I will go through each of the differences one by one:

A Newly Discovered Planet With Alien Life Forms

Assume mankind invents a generally useful artificial intelligence architecture and sends it off into space. In the utility function of the agent it is programmed sub-goals to find new planets, mine and valuable metals and bring it back to earth orbit. The agent accelerates using lazers to 50% of light speed, making it to another solar system in a few years but without reach of instant communication.

Unfortunately, the human programmers forgot to include a sub-goal to avoid planets that contains alien life forms. They also forgot to make the agent report such sightings and return to earth immediately, as it would be history’s most important discovery. Since there are plenty of empty planets in space, it is extremely unlikely that the metal would be worth the risk of mining it with the alien life forms present on the planet, for example by damaging the atmosphere with pollution (like on earth). This mistake was figured out right after the agent left communication distance with earth, and with the programmers scratching their heads, worrying about their careers, there was nothing more to do about it.

What happens next is even worse than anyone anticipated. The agent lands on a planet with primitive alien life forms and start mining, which sets off an earthquake. The earthquake triggers a super vulcano that kills all life on the planet by blocking sun-light with ash clouds, causing an eternal winter. The agent brings the metal into orbit and ships it back to earth. Mankind celebrate their successful technology without knowing about the alien life forms that was lost for eternity.

This thought experiment shows how easy it is to make catastrophic mistakes simply by forgetting to add something to a utility function. It is because utility functions are brittle and do not capture the complexity of preferences agents should have to behave ethically in a complex environment.

1. A utility function might be arbitrary while a granular judgement is not

A utility function is just a function. It can be programmed into any agent that supports a general purpose programming language for specifying goals. This means that you basically can make the agent do anything, including things that are non-sensical to us humans.

For example:

```
U := \(x : world) = {
    n := number_of_apples(x)
    sum := 0
    for i := 0; i < n; i += 1 {
        res += random() * x.trees[i % len(x.trees)].height_in_m
    }
    if (sum % 1000) > 3 { -1000 } else { 1000 }
}
```

This utility function has one partial function:

U(x : [number_of_apples] 0) = 1000

However, for a non-zero number of apples, it is very hard to know exactly what it will return. It depends on how trees are sorted. For 1 apple, it is likely that it returns -1000 if the first tree is lower than 6 m (because 0.5 * 6 = 3) and 1000 otherwise. There is a lot more than one apple in the world, so this particular return value is not useful. When you assume a realistic number of apples, the utility function just returns noise.

As a consequence of maximizing this utility function, the agent will kill all trees in the world with apples, since it receives no higher score than 1000 and when there are no apples it receives the same score all the time. I just made up an arbitrary utility function and without knowing or intending, it would be a catastrophic mistake to use it. This is how brittle utility functions are in macro-models.

A granular judgement requires some function and some output the function will return:

E(x : [g] b)

This means that if the function does not return the value, it is non-sensical:

b : [∃g] true

The above is a check that is required to pass before it makes sense to make a judgement. This is a rule that follows from granular judgements using statements in path semantics.

So, you can not just type in some random code which runs by accident (this is very unlikely):

1. You must refer to a function (either formally defined or learned over time)
2. You must know the function takes the world as input and returns the type of value
3. You must now the function returns the value

2. A utility function does not guide prior estimates while a granular judgement does

Back to the example with Peter:

$$E_{peter}(x_0 : [loves(clara, peter)]\ true) > E_{peter}(x_1 : [loves(clara, john)]\ true)$$

If this is the only thing that is known, an AI is asked to estimate on Peter’s behalf:

$$E_{peter}(x_2 : |\forall\ X\ \{\ [loves(X, peter)]\ true\ \}| > k) > E_{peter}(x_3 : |\forall\ X\ \{\ [loves(X, peter)]\ true\ \}| <= k)$$

Does Peter prefer a world where for some constant `k`, more than `k` people love Peter than a world where `k` people or less love him?

The AI believes that the more people who love Peter, the higher chance is it that Clara loves him. Therefore, it believes that for any constant `k`, e.g. 0, Peter would prefer to be loved by somebody or someone. Even if the AI figures out that Clara loves John, it might suggest this to Peter as an idea, to get to know somebody else. This is because the AI estimates this as a prior, without being very confident that this holds, because Peter might have objections.

On the downside, even if Clara loved Peter the AI might try to get him more than one girlfriend, just because it might seem like a good idea to the AI. It should probably be programmed to ask Peter first.

A zen robot, which is a zen rational agent (extended instrumental rationality to higher order goal reasoning) programmed to follow Rational Natural Morality (RNM), would be able to infer a LOT more than this. It would know that increasing the population on earth rapidly, just to make more people love Peter, would be a bad thing. It would know the physiological limits of the human body and mating behavior. This could make the zen robot able to infer that Peter would probably not like to have many girlfriends at once. It might even infer that Peter would prefer to be with somebody with similar interests. What the zen robot suggests to Peter, when observing that Clara loves John, could be weighted by a thousands different concerns for which this particular estimate did not weight as much.

Using a utility function in this example requires understanding how to extract the sub-goal and make the same inference. This is far from obvious. One ends up defining the whole theory of path semantics and granular judgements just in order to justify this prior estimate. This is an extreme level of complexity.

Worse, a utility maximizer might find up arbitrary ways to try make Clara love Peter instead of John. For example, by simply killing John. It would never occur to this kind of agent that Peter might prefer John to love him and therefore John should not be dead.

The brilliant thing with ethics as rational reasoning with granular judgements about the world, is that it includes path semantical reasoning for computing prior estimates. With other words, it is a strong “don’t be stupid and don’t just try random things”-signal to the agent. This adds another level of safety that is completely absent in utility functions.

3. A utility function tends to be closed while a granular judgement tends to be open

Since a utility function in principle is just a function, it tends to make a closed-world assumption. When it is defined, there is nothing else in the world that the agent cares about, so it does not treat any life forms that are not added to its utility function as something worth to protect or even avoid harming.

Previously, I shown how removing information from a granular judgement leads to natural prior estimates. This might lead to e.g. a wish for humans to not die, to be generalized by an AI to not harming animals. While this property is important, there is another reason granular judgements might perform better in a macro-model of the world: The open-world assumption.

A granular judgement is only telling something in rough and approximate detail about preferences.

$E(A)$ might give a prior, but does not define specifically $E(A \wedge B)$

So, when the AI learns more about the world, it understands that some of it might have implications for preferences it already knows. It figures out that sometimes, the preferences can be made *more specific*.

If the AI starts to wonder whether some preferences can be made more specific, it might just go to the source. For example, to the human programmers, or perhaps asking people if it understands that its goals are meant to reflect human values.

When the AI learns something completely new that it can not relate to previous experience, it knows that there are no preferences specifically defined for a such situation. This could be a new invention that the AI discovers, some piece of new technology, which it believe can have applications in other areas than the intended purpose. A such discovery might have very bad outcomes, for example a new kind of weapon, or it could have very good outcomes, for example a new medicine. The AI will be uncertain about the positive or negative aspects of the technology and therefore display careful behavior.

A zen robot that is used to protect life on earth, will also be able to make priors about life forms on other planets that they worth protecting. It might also believe that when aliens attack earth, it should help us fight them. Not ALL life forms might be worth protecting. This uncertainty, that new situations could be valued strongly positive or negative, makes it easier to secure future safety also for rare encounters with other space faring civilizations. Even if an alien puts their feet on the ground, does not mean they automatically come under the zen robot’s protection. This is a refinement of the preferences that the zen robot is programmed to have, which it learns over time.

When thinking about closed-world assumptions vs open-world assumptions, one might initially think of this as a formal but not-so-important property. However, if you really think about it, spending some time pondering about this issue, you will probably reach another conclusion. It is kind of like adding NULL-pointer in a programming language, which is a simple thing to do, but which has costed the industry multiple billions of dollars.

If I had to pick either guiding prior estimates or the open-world assumption, I would pick the open-world assumption, because at least it does not assume that all preferences are known to some degree. It much safer to use the open-world assumption and simply do nothing in a new situation.

4. A utility function is unique while a granular judgement is usually not unique

A utility function is unique. It means that whatever you can say about it, is the same as you can say about an identical function. I mean unique in the logical sense, of course. There are many ways to program the same utility function, that might perform in different ways.

If Alice and Bob share the same utility function, but Bob interprets the utility score in a different way, then the utility function is still unique. What makes Alice and Bob different behaving is information that one can say about Alice and Bob, or perhaps the world. Uniqueness about a function is assuming you have a compiler and programming language that behaves in a consistent way. For example, it reads in some values from memory and writes some values in the same way. Whenever you load the same function, it behaves the same way as before. It behaves the same way whether you call it on a Friday as on a Thursday (or some situation that is unrelated to what the utility function does).

A granular judgement is usually not unique. This is because formally, it is defined in abstract path semantics, which uses overloaded functions. Such functions can be overloaded depending on the situation.

Here is an example:

E(x : [blue_sky] true)

If you put on red-tinted glasses, you see red everywhere, so the sky appears red as well. The granular judgement of the sky being blue might be related to how you feel about nice weather in general. Just because you put on red-tinted glass, does not mean you change how you feel about nice weather in general. The problem is: What does it *mean* that the sky is blue? Does it mean how it is seen through your eyes? In that case one would need to add more information.

E(x : [blue_sky] true ∧ [wearing_red_tinted_glasses] false)

It is easy to see that one can not add such exceptions for all possible situations. This becomes computationally intractable. Instead, one assumes that the function `blue_sky` is *overridden*, depending on the situation, such that one makes similar granular judgements in similar situations.

If you put on red tinted glasses, and you feel happy when the sky is blue, the world has not suddenly changed into a horrible place. What you mean by feeling happy when the sky is blue, is that you feel about as happy in that situation as you do when there are minor changes to that world. This includes putting on red tinted glasses. In general:

E(x : [wearing_red_tinted_glasses] true) ~= E(x : [wearing_red_tinted_glasses] false)

So, when the only change is that you are wearing red tinted glasses, the granular judgement of the sky being blue takes on a different meaning, perhaps even unknown, such that you feel roughly the same anyway. You do not have to care about the specific definition of the sky being blue. Just carry over the general good feeling, and take off the glasses if they make you uncomfortable.

An AI using a utility function is wimpy: It is scared of any small change. This is not a problem with granular judgements, because they are interpreted as overridden functions and not unique ones.

5. A utility function is usually absolute while a granular judgement is usually a constraint

A utility function when fully defined must return some value, for example:

U(x : [blue_sky] true) = 18

What does `18` mean in this context?

A utility function returns utility points, or *utils*. The *util* unit is a way of measuring preferences relative to each other. What it means depends on how other preferences are valued. For example, `18` could be very low or very high, it is impossible to know without examining other preferences.

Both utility functions and granular judgement might have no well defined data type or range. It can use scalars or some other ordered type. It might also be a probability distribution, or a range. Maybe even deontic modalities (obligatory, permissible, ought to)?

When you have a well defined data type and it is ordered, it is sufficient to serve as utility.

A granular judgement on the other hand, often comes without specifying a specific value. For example, it might be known that Peter prefer blue sky verses a non-blue sky, but not setting a specific score:

E(x : [blue_sky] true) > E(x : [blue_sky] false)

This is because when reasoning using granular judgements, the absolute value of preferences is much less taken seriously than relations between preferences. One can think of this as granular judgements being a kind of constraints, such that their value might lie in a range or being distributed by probability.

Therefore, using granular judgements, it is much more important to justify an action based on proof consistency, than based on value consistency. If the utility can be increased by 0.0000001 utils, then it better be a high quality chain of reasoning behind that choice than interpreting the utility function literally. One might even think of the `>` operator being overloaded as well. It usually takes on a meaning of “bigger than some insignificant difference”.

An AI programmed with a utility function will kill all humans on earth just to increase the score with 0.0000001 (or even less, it only matters that it is larger). You could achieve the same with granular judgements, but this would take a substantially larger effort, given the agent architecture should perform well in other situations too.

An AI programmed with granular judgements might have doubts about doing something, even if there are no specific contradictions to what it is thinking about doing. It might doubt simply because there is only one reason it could come up with to justify the action.

For example, when an AI is heading home, it might think that is a good choice for several reasons: At home it can charge its battery undisturbed, it can receive new commands from the owner, it can achieve some task it is thinking about etc. However, when the AI is heading someplace it never has been before, it might be more careful. It could be run over by a car or fall down a stair. It does not think that buying groceries is absolutely necessary even when the store is closed, etc. Its programming might allow enough uncertainty about the preference to allow it to head home again without being explicitly told so.

6. A utility function is unambiguous while a granular judgement might depend on use of language

When using a utility function, one takes the statements unambiguously. A sky is blue e.g. when there is a certain amount of permitted clouds and the gradient color is not changing too much etc. This way of programming gets annoying when you try to deal with the complexity of the real world.

One property of unambiguousness is that when we say something, we usually imply something else. When it is summer, we usually imply that the temperature is hot, even if we do not explicitly say that the temperature is hot. We do not imply that the temperature is the average measured in the summer, instead we imply some temperature range that we associate with summer. This can vary from person to person, from culture to culture.

The probabilistic semantics of granular judgements takes into account a confidence factor that when we say `A`, we usually imply `B`. This can be thought of as a hidden probabilistic sub-type of the ethical judgement. In a micro-model of the world where the judgement is fully defined, this takes on a concrete computational interpretation through the concept of probabilistic paths. There is a well-defined probability of the output of a function given probabilistic belief in the input. The hidden probabilistic sub-type is then used to justify the belief that the function will return some output. At first sight it might seem that granular judgements are identical to utility functions in micro-worlds, but actually they might better capture language usage in those cases too.

One sentence in one language might mean something slightly different in another language. This means that an AI using granular judgements could infer a prior estimate of what somebody wants who speaks German, when its original goals were programmed by somebody thinking about statements closer to English.

Combined with function overloading, constraints, open-world assumption, guiding prior estimates and non-arbitrariness, this provides a powerful semantics of capturing dynamic use of language. The ambiguity is not only valid for a single thing. When we say `A`, we might imply `B`, but we could also imply `C`, `D`, `E` etc.

So, when we say to an AI using granular judgements:

> “Please, help me!”

It might understand much better what we mean than an AI programmed with a utility function that brings all police cars, all fire trucks, all ambulances, just because it might be a small chance that somebody are threatening us, or there is a fire somewhere, or we might have gotten hurt. Adding another police car, fire truck or ambulance might increase the utility function a little bit.

Perhaps we just want to cheat on a video game? Or helping carry groceries out of the car?

Understanding how we use language is very important to understand what our goals are, and what humans values are like.