# Real Fractal Meta Probability Theory

by Sven Nilsen, 2018

*In this paper I introduce a Meta Probability Theory (MPT) called "Real Fractal MPT" that is a solution to some problems of interpreting probabilities related to higher order goal reasoning. Real Fractal MPT assumes that modeling identity of real numbers is a set that can be approximated with course-grained sets, recursively. Measurements are then indexed by real numbers to create an order, which has the fractal property that for every real number of same identity as any nearby local optimum, the predicted probability is ¼ that the number is measured to be new local optimum (or equivalent). Global optimum can be "proven" given a correct hypothesis of the "true probability", which can be falsified over time by collecting counter-evidence. With other words, an agent can believe the "true probability" without ever measuring it directly, while at the same time assign a correct probability to the belief from evidence.*

One basic problem of Artificial Intelligence (AI) is that when an agent believes it has achieved a goal with 99.999%, it might figure out a way to increase the probability to 99.9998%, or 9.99998% etc. This has consequences for AI safety, since failure of terminating after achieving a goal can lead to harm.

For example, if an Artificial Super Intelligence (ASI) has the capability to make a weapon, it might use it against some humans because it figures out there is a 0.0000001% chance these particular humans are a threat to it achieving its goal. One way to approach this problem is to use a simplified model of the world where it is assumed that such probabilities are interpreted with a Meta Probability Theory.

I call this new MPT for "Real Fractal MPT".

The semantics of Real Fractal MPT exploits the semantics of identity of real numbers.
Assume that you have an indeterministic function `f`, which measures some property `a`:

$$x : [f]\ a$$

The probability that `f` returns `a` is given by the probabilistic existential path:

$$P(\text{“`f` returns `a`”}) = (\exists_p f)(a)$$

However, when measuring probabilities using real numbers, it would be equally correct to say:

$$(\exists_p f)(a) = 1$$

as to say:

$$(\exists_p f)(a) = 0.999999\ldots$$

Since $\exists_p f$ must make a choice, it means that all the alternative ways of representing the same real number are free to be used for other purposes. These representations are not "used" by the agent for anything and this property can be taken advantage of.

A complex agent needs to operate with multiple layers of models about the world. One model is perhaps less detailed than another, which often leads to inaccuracy in predictions. Yet, if a model predicted another model perfectly, there would be no need to use a more complex model. There are always some interesting properties in the more complex model that do not exist in the simpler model.

The purpose of modelling a complex model with a simpler one is to do most of your thinking in the simpler model where reasoning is more efficient, then translate this as a hypothesis over to the more complex model.

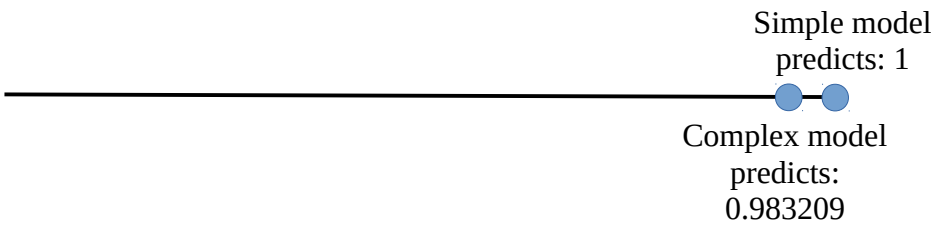
So, if `x` is some measurement in the more complex model and `a` is a similar measurement in a simpler model, then `f` is a function that transforms a measurement `x` into `a`:

x : [f] a

The function `f` is used to check whether some simpler model predicted `a` correctly. One can not use `a` directly, because this would just copy over the more complex model. Instead, the agent must form a hypothesis of the “true probability” that the simpler model should have.

The semantics of “true probability” is such that when obtaining the correct number, the simpler model has some predictability within well-defined constraints. In reality, there is no “true probability” because reality is just itself. The “true probability” exists only as number in the simpler model together with semantics of how to interpret it. It is called “true” because it satisfies the constraints.

On the real number line between 0 and 1, there is some difference between what the simple model predicts and what the complex model predicts:




Real Fractal MTP assumes the following:

> If the simple model is correct, then there exists some number
> arbitrary close to the prediction of the simple model
> which is within a stochastic variable range
> of the complex model’s prediction.

With other words, an indeterministic transform `st` “nudges” the probability a bit such that:

$st((\exists_p f)(a)) \cong (\exists_p g)(a)$

Where `g` is a function computing what the simpler model predicts.
Both `f` and `g` are constrained functions.

If you wonder if this is true for all `x`, then you should known that this is actually a more powerful statement in path semantics: For all sub-types constraining the input of `f` and `g`, this holds.

Translating to traditional semantics of functions:

$$st((\exists_p f\{\forall f\})(a)) \cong (\exists_p g\{\forall g\})(a)$$

So, you can set `∀f = (= x)` and it implies that for similar inputs of `g` (the models must be tuned to each other), the assumption of Real Fractal MTP holds for values of `x`.

The expression `st((∃ₚf)(a))` is called a “measurement”. This measurement is probabilistic. It does not return the same value when calling it twice. This means that when one measures something from the mode complex model, it is not certain that it means the same as what the simpler model predicted. However, by calling it many times, one can obtain a value that is arbitrary close:

$$st(0.983209) = 0.999999\ldots$$

This is where semantics of real numbers come in: One can treat multiple numbers as identical.

The number of times one must call `st` is to obtain an identical number to the prediction of the simpler model is the proof of work to verify the hypothesis. This semantics is necessary to compare proof of work for different hypotheses against each other. By keeping a sorted log of the returned values of `st`, one can see how much effort the agent has tried to verify a particular hypothesis.

For example:

[0.00001, 0.0001, 0.001, 0.01, 0.5, 0.6, 0.9]

Here, one can see that it tried to look closer and closer to `0`, while other parts of the range were unexplored. There is much more details in the log around the hypotheses that the agent cares about.

What the agent tries to verify is this:

> For each “true probability” there is an assigned utility.
> The agent tries to optimize the utility by exploring
> probabilities that it believes will increase utility.

With other words, the agent assigns higher confidence to hypotheses that helps it to model the world in a such way that the expected reward is maximized.

According to the semantics of real numbers, one can take any two numbers `a` and `c`, and return a new one `b` that is between `a` and `c`. If these two numbers, `a` and `c`, are considered identical under the limits of the simpler model, then `b` is also identical under the limits of the simpler model.

*Key assumption: Since `a`, `b` and `c` can differ as representations of real numbers, they can be associated with different rewards as memories in the past.*

Assume a reward function `r` that takes the probabilities and outputs a reward for that measurement:

(r(a), r(b), r(c)) : (reward, reward, reward)

The probability that `r(b)` is locally optimal is:

P(r(b) >= r(a) ∧ r(b) >= r(c)) = ¼ = 0.25

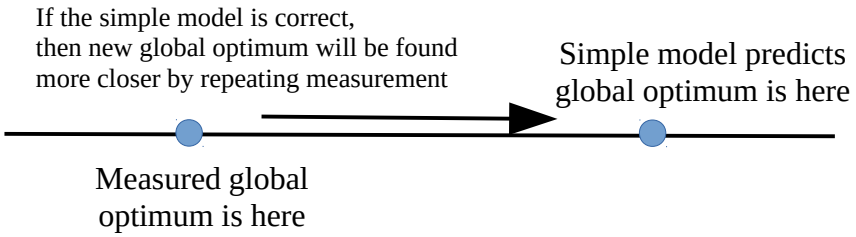
Proof (assume `b`, `a` and `c` are identical):

P(r(b) >= r(a) ∧ r(b) >= r(c))
P(r(b) >= r(a)) · P(r(b) >= r(c)) assume independent probabilities
½ · ½
¼

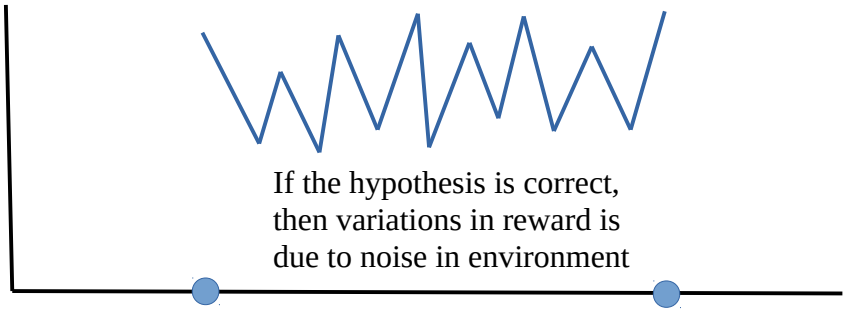
This is a fractal property, since you can do this recursively.

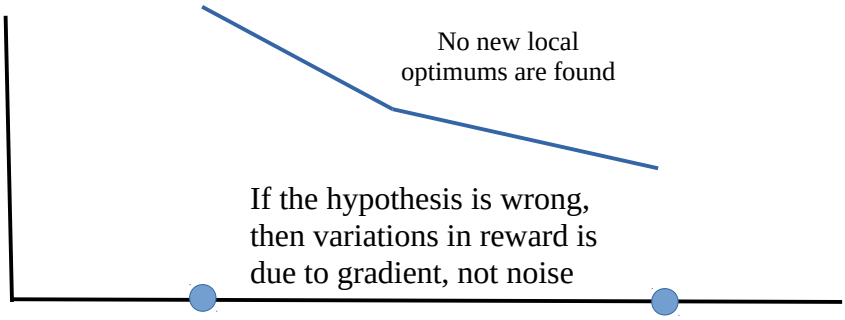
Since global optimum is a local optimum, the agent can verify the hypothesis of global optimum by exploiting the fractal property that yields expected probability ¼ for a new nearby local optimum.

The agent tries to tweak the constraints of its actions such that the measured global optimum gets closer and closer to its hypothesis. When it is close enough or it has tried too many times and failed, it makes a decision whether to build confidence in the simpler model.



Another way to see this, is thinking it as a detection problem of noise vs gradient:




This property is scale-free: The sensitivity in the “true probability” can be arbitrary. What defines the right amount of sensitivity is how likely it is to find local optimum. When there is noise, it is expected that ¼ of the time a new local optimum will be found compared to its ordered neighbors.

Real numbers has an advantage because multiple representations of the same number can still be ordered. When two numbers are considered identical, their variations in expected rewards is due to noise. So, by comparing ordered neighbors with each other, local optimums are expected ¼ of the data.

When this expected probability of ¼ does not happen, it means that there is a gradient and the identity of probabilities have not yet been found.

This means that the context where two “true probabilities” are considered identical, depends on the probability distribution of the environment, since one can only consider two probabilities to be identical if their variations in rewards are due to noise. If there is a gradient in the rewards, then it means the model of the world is wrong and the agent is comparing apples and oranges.