BiDirection Interperator for NeroDivegenct AntiHallicution Safety Critical - Copy

Okay, Namdi, this video is on PLP, Ontological Bayesian Intelligence System. So, I've got this video ready, but I think I lost it. So, what is PLP? PSP stands for Phenomenological Lensing Principle or Phenomenological Lensing Protocol.

It really means Ontological Bayesian Intelligence Infrastructure. Like, you know, my PhD is on Ontological Bayesian Infra-intelligence, but, you know, that means small stuff. So, the term Ontological, the nature of knowing yourself, you know, know yourself, to know who you are, you know, your BFN history.

Ontological, who, what, why, when, why, you know, how to build your stuff. So, Ontological means, like, role, like, you know, I know my goal, I know my purpose, I know what to do next. Now, the Bayesian means, like, probability of it, Bayesian inference, like, you know, half and half on the Bayesian graph.

The thing about Bayesian is, you know, you have to get the probability of range, you know, like, you know, something can be 0.5 to 0.9, but that's when the function is coherent and the function works and the thing works. That's Bayesian inference. Intelligence, AI intelligence, smart devices, things that don't need consciousness, model, you know, my lead architect and system architect.

Ontological or Bayesian reasoning, probability. Rhetorical or reasoning, like, you know, for AI systems, you know, you can use that to ensure an AI makes the right decision based in real time, you know, when something is an edge case. The rhetoric means, like, you know, my rhetorical reasoning, like, you know, it's just talk, but it means something, it has some grounding, but, you know, it's not practical.

Maybe practical, yes, it can be practical, definitely, but, you know, it's something like an edge case we haven't really encountered, you know, it has to absorb that, you know, by itself. That's what the rhetoric reasoning, rationale, like, you know, the rationale behind doing that. So I changed the rationale, you know.

The goal is to define a suppose of this ABC. Okay, now, the goal of this is to have a dual model, observer to consumer. Now, remember, guys, the observer model is just what, you know, it bubbles up errors.

It doesn't propagate any errors downstream. It bubbles up the source code error. And the event loop is witnessing all the time for the state change, based on local maximum and local minima.

So when a programme observes its state, it's really observing the local minima, the changes it does, and what it does, what's going to change, what's not going to change, you know, and then it changes what's going to change. It's like a, it's just witnesses, and that's called witnessing. So when it witnesses, the time-preserved portfolio time, that acts as a factor of four of the time.

Because the other thing, you know, the factor of four is just a standard testing, but the 4.5 is just the other one. So, this is like, this is what we interpret. The PhD on direct instruction, direct simulation systems, which are treated in the same thing, observer to consumer.

Observer to consumer in quantum systems, you know, you can't really observe, when you want to change in a quantum state, oh, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no. So basically there's an ad there, let me just pause it. So you don't want to have an interpreter state, you know, mangled or wrangled or all that stuff.

So basically, OBS interpreter, any Python or Python, any polyglot interpreter must cohere, like you know, it must be working with the Gussie manuscript, the Gussie language polyglot manuscript. Now, the manuscripts aren't really defined properly, but you know, we need an explanation for interpreter, because interpreter for Python, doesn't really differ from interpreter for Lua, etc. So the goal of this is to observe that the interpreter, a photocard system, is a song for the interpreter to Lua system, via a Gussie lang polyglot manuscript, .gsm files, .gs files for Gussie lang.

Now, the Gussie lang files are just polyglot files, it's just like a polyglot. So the goal of this is to have the .gs files, .gsm files, which is the interpreter layer. So let me give you an example, so Python can define something called bi-weekly, once in every two weeks, you know, once every two weeks, or once, you know, is it once of every two weeks, or once bi-weekly, so ambiguity in the term bi-weekly, or bi-daily, let's say it's bi-weekly, once in two weeks, you know, something happens in one week, every two weeks, one in, in, but then it's off, one off, one, every time, once off two weeks, you know, that's the thing.

So every once a week, you skip two weeks, you don't do it for two weeks. Every one in the two weeks, you know, the interpretation is on or off, you know, it's like an in or off loop, you know, that's the bi-weekly or bi-daily interpretation. So Python might interpret this one way, and then Luam has to go the other way, you know, based on whatever the standard specification is, because these are interpreted languages, not compiled, that execute as you run it, they're not compiled, they have a runtime to add byte-byte in the real world, and as I'm saying, the goal of this is just to have a system that understands this concept, and can infer from the new and better three AVL nodes, three AVL red-black animation, which red-black animation is just a standard algorithm I'm using to animate all the red redundant nodes from a system that needs black batteries, you know, and as I say, no, the red-black animation is just a standard for it.

Now, for the quantum computers, you're going to understand that the observer-consumer model had another hook attached. So for example, if you observe, you know, the observer, you can't observe a state and change something in quantum computing, you only observe something in concepts in a quantum system, so you have to ensure that you can act before you can, you know, consume. So for example, if I'm observing, I can have a pre-observed model, pre-observed, you know, something like a pre-event, a bubble, an event loop, that witnesses.

So when an event pre-happens before another event, that's the duality that acts on that event, because the system needs the next state of that information. So you can create the information, you can get the information you need before you ever observe, or you can get the observation, you run the code, act, consume, before the function caller, callee, uses it. Now we've got, so we've got a matrix, 0, 0, 1, 2, 3, in, off, so you've got like an in, off, if you just remember, for in loop, for off loop, you know, these are just standard loops for the interpretation layer.

You can try to understand this more formally, you know, to ensure that the system understands the concept and can resolve itself based on the aviary rotation for that set structure, the bi-query key. We are a bi-DAG. Okay, now, very important, the bi-DAG stands for bi-directional DAG, it's like, it's a bit of the things that are interpreted by most of the context.

So everything that's interpreted has two states, you know, it can be either one state or the other state, so it's kind of weird to, you have to rotate to ensure that the system is flexible. So the thing about the bi-interpreter, or the bi-interpreter, is that they have a bi-directional, bi-directional interpretation, it can mean one thing to one system, and another thing to another system, but you know, that's what's quite humorous about it. It reminds me of AI host solutioning system.

You can stop an AI host system with this framework too. This one's important. So if you go to github.com slash o-g-h-e, drive, or like a drive, you know, I think that means like a web drive, yeah, so it's an Apple web drive, but you know, that web drive is for the AI system.

So you get like a very important item. So basically, how this interpretation stops the AI from working incoherently is this. So remember those things you did at school? 20 equals 8, 25 equals 10, and 6 equals 3. What is 5 equals 1? So this is just another implementation.

This is called my anti-AI hallucination system. So basically, this 10, this is a system of linear equations, but you know, the goal is for you to understand what's happening here. So you see 8 equals, 20 equals 8. This is 25.

This is just one system of linear equations, but you know, it's like a puzzle really. It's a puzzle. It's like one of those puzzles that gives you, so 20 equals 8 means one system's 20 is one system's 8, you know, another one's 8. So there's a layer between, when you have an 8 in a Python file, you know, or number 8, you know, what is 8 made of? That's 20 in like a Lua, or C. 20 in Lua, because Lua is an interpretive language.

So when you have 25 in Lua, you have 10 in Python. When you have 6 in Lua, you have 3. And then when you have 5 in Python, Lua, you have X. But what is X? We don't know X. The goal of this thing is to ensure that the AI or systems that are active never hallucinate or make up their own interpretation of that assisted. It must resolve the bi-directional, bi-directional directed side-click graph with open access derivational tracing system to trace all the sequences and the series of function calls.

Recursively, not imprecursively, but you know, it's resolving a circular dependency to ensure that every dependency has a telemetry ID, which is called a unique user ID, for that local scope, local maximum scope, local minimum scope, based on what is allowed to execute. Now, this puzzle can be geometric as well. It can be like a square.

This is like a cube, or it can be a mathematical geometry. This represents two acres, you know, two land, you know, two land. You know, this is like an example.

These are auxiliary constructions, you know. They geometrically are things you make to solve the problem, you know. And this puzzle is just either geometric or algebraic puzzles, which I call them.

They are geometric or algebraic. Because, you know, they can be in the form of a graph, a diagram like this, or it can be a form of anything else. Okay, Namdi, 3, 2, 1, pause.

Namdi, 3, 2, 1, take a picture. Okay, so this is a puzzle here. It can be a puzzle here.

But the thing about this number system is, you know, this number system is non-universal. We've never seen the number... What is universal concepts? You know, it's a concept we use. We've never seen an actual number 9, because the number 9 in French, we've never seen the number 9 in French.

A, B, C, D, F, G, that's it. But, you know, we've only seen the number 3 in real life, walking about, you know, Hi, I'm number 3. But, you know, I'm the only number 3. You haven't seen the number 1, 2, 3. You haven't seen 3. The number 3, you know, 3 in French, you know, the spelt in French way, you know, walking about, it's not an actual concept. That's what I'm saying.

This system must remain coherent by the way they're modelled. And this is why the PLP is the mother of the pathological Bayesian infrastructure. So, my goal is to ensure this works coherently.

I can try to say that, you know, this is just a puzzle, yes, but you know, this can mean a geometric alignment or a system, it's for anti-AI isolation. So I'm going to try to anti-AI, how do you say, so the AI cannot isolate and interpret via interpretation. Too much interpretation.

So, basically, the ten of others is to remove a vector bi-weekly, bi-daily, ensure it's a bi-DAG, bi-organised digital trading system bag. The sequence of instructions or sequence of series of steps, a sequence is just a non-alternate series, but you know, a series is what can alternate, can split between interpretation based on context. Which is context switching? But you know, context switching, not code switching.

Because, you know, switching context is based on the user's needs, you know, inputs. Now, this is part of the Functor Frameworks. Now, Functor Frameworks is very important for this, because those are the mappings for the system.

If everything is a functor in an infrastructure, then you can evolve it easier and seamlessly. There's no need to slap a new component in it, a new component. Phenomenology-Phenomenal Protocol, consumer to observer, you know, it can be resolved like that, it can go from observer to consumer, consumer to observer, observer to consumer, in any way, shape you want, but the goal is to resolve the state of mind.

Functor... So, PLP, Objective of PLP Framework. Now, PLP stands for, just like example here, you can see this, PLP. So basically, the PLP stands for Phenomenology Learning Principle.

HOLE, now, HOLE stands for Homogeneous, or HOME. I call it HOM, Homogeneous. Now, you see this data 0, 0 here, this one is uniform.

It goes like, 0, 0, so this is an input to the matrix, 0, 0. Now if you get another matrix, you get 0, 0, you get the same matrix, 0, 0, you get 1, 1, you get 1, 1, same, you know, same. And this is a PLP, here, here's just data, but you know, it can be, it can be anything, but it's just a standard form of it. Now, anything that's input to this matrix is the input.

It can be in a vector form, sparse, it can be in the same form, but you know, depending on how you want to use it, you know, how the data is structured, how you want to best transverse the graph of the data, based on the open access deregistration code, but you know, the goal of this vector is to align with a uniform structure. If it's heterogeneous data inputs, you know, that is a uniform and can be modelled quickly. This can be defined as sparse.

So the sparse concept is like, you know, if I want to model some information in a uniform manner, you know, I'll define that uniform thing, input, as the model, in the model, because you know, if I have like a data structure for a person's student ID, I don't mix it with student credit card or bank card ID, because you know, that's not identified by all the students, but if I have a student number, student name, student name is me, first name is Namdi, Namdi, second name is Appala, and then DOB is my domain. This is not a uniform data, because you know, this is a uniform stream of information, just the first name is what, a string, and then the second name is a string, and the DOB is my domain. Now, this applies to all homogenous data, but you know, if I then have a, if I want to store a unique key, which is called a key, a foreign key, the foreign key, a recent ID, a foreign key, and I put that there, and that foreign key, you can use to go on the next student ID, but you know, if it's not coherent, if it's not, if it's like, if the information is not coherent, I can get like a non-uniform model.

Oh, sorry, sorry, this is a non-uniform model, sorry. So this one is non-uniform, because you know, it has a, it represents a student, enough about a student, a student name and number, kind of a boring, a student name is like Namdi, and the other name is string, but you know, so that's just one side of it that's uniform, there's two strings there, now, the, what is it called, it's uniform, because it has two strings there, that's the first and last, but it makes it non-uniform, because the date of birth is a, oh, sorry, it's a string, so it's uniform, so it's, you know, you put in the 19th of November, if I put it in 2001, sorry, 19th of May 2001, that's uniform, because they're all strings. Now, if I ever put my age there, my age string, four years old, and then age is 24 years old, that becomes a non-uniform data, because you know, there's something there, the data is non-uniform, you know, you can put in another data structure, or we can handle it another way, but you know, that data can be something else, it can, it's structured, the model is still structured, but the sparse locality construct, because you know, I don't need it, if I want to define a function now, I have to either, and this model doesn't need the age, it doesn't have to need add the age, you know, you don't need the age at all, so you just need to discard the age, you know, it doesn't need it, you can just make it private, you can define it between a functor, you can define a structure that handles the names, the string datatype, and the names and the numbers, you know, this is called a schema, this is what the OpenAccess schema is about.

Now, the schema for OpenAccess is services dot operation dot openaccess dot division dot department dot county dot org, so basically, that schema is a by DAG resolution case for the KNN cluster of the search engine, so basically, if you search for plasma services plasma systems dot operation services brain computing interfaces dot operation plasmoid systems plasmoid systems like, you know, plasma technology I'm making, dot openaccess dot department, department is sales department and dot division computing dot county county or ha county, the public housing dot o-h-a and each i-j-a means like order count dot org, and that's what I'm saying. So, this function would ensure that the isomorphic layer between the data is met with the interpretation of the OpenAccess to resolve a system using the the puzzle framework, you know, the kind of puzzle you're going to do as a child, like one number is another number, two sticks equals one big stick, you know, and ensure it works, you know. Now, that's what PSP's sparse data is, this is sparse enough, because, you know, the sparse data is uniform, and uniform data is the same type and everything, you know, everything is consistent in the, you know, going from input, you know, there's no different type.

Now, you get a PLP on non-uniform, which is you know, non-uniformity in the data, but the same, basically, the sparse they do the same thing, but the they have the same model of that data structure, but they're different, because you know, they are Hamiltonian resolved and you know, basically, the Hamiltonian, basically, the one or zero, one to zero, one, you know, is different, because, you know, it means the same thing, but it's just a different way of doing it, representing that same information or problem, you know, it's like, you know, it's a bad direction. So, basically, I'm trying to explore you guys to explore, I want you guys to explore this one, try to implement a student ID system with a telemetry tracker that ensures the schema of the system is null, nil, and nil to null to data, but you know, if the data is nil, there's no memory, no type, no value, and then that's nothing to be used, and then that will be reset to nil if there's nothing to be collected, because the goal of this is to have the bits fields aligned. For example, if you have a RIFT system, RIFT ecosystem, I'll just write it down here.

So, basically, I'm saying that the system for RIFT is, like, RIFT stands for RIFT is a flexible translator, and RIFT must ensure that the system is nil when you have, like, a schema, a schema with, like, a nil template, you know, nothing is being stored in the system, and how RIFT handles schema is null, nil, nil will be the default placeholder for no value, no memory, no type, but, you know, the goal is for every data structure to have something I call a bits field, like, you know, the bits field is a memory allocator, you know, say the string a string, it's 8 bytes, no, a character is 8 bytes, how much is a string? I don't know, but, you know, the size of the string is, like, you know, a void pointer is a nil, a void pointer is nil, a pointer to nothing, nothing, before the memory is allocated, you know, you point to nothing, that's a void pointer, but the size of a data structure, size of a void pointer, a void type, so, the null type is nothing, the null type is the size of the void pointer, and that's what the null type is, and the null type is nothing being allocated and point to user and the data itself, so I'm saying to you, you can define a programme and a student ID that has a schema to resolve students in a boarding school, so you've got the children, I was in Queen of England boarding school, you've got the children, and then you've got the young people, and you've got the adults, the young people, and then you've got the 18 years old, 20 years old, 22, it doesn't matter, so the struggle would be to have the data set again, and then the young people have different kind of data, you know, there are still students yet, but they have different requirements, so basically, the age of the child is 1 to 12, 13 to 18, and 18 to 25, that's when you leave the boarding school, that's when you're done with education anyway, so that's what I'm saying to you, so the thing about this, you're going to have a system that expresses null as null, so basically, if I want a child, there has to be some kind of coherence between the service offering and the null template for the schema, based on spots, so I can say children, uniform children, children type, age, date of birth, but you know, everything about child type in a computer, so I can say the child is male, boy or girl, that's a Boolean, now this Boolean is a Boolean data, a structured data, now, I get the structure, the end is a bool, bool, neurodivergent, neurodivergent, that's a bool, and then you have something else, that's just one output, and then look here, I can put the date of birth, the DLB, and the year, DLB is name, name, name, first name, last name, and that would be strings, strings, strings, and that would be it, and then you can put this as a date time, but the string is possible, so date time, so you can think about things like that, if the system can represent the data, because you know, basically, you can put a calendar system here, but you know, in the database, it's not a calendar system anymore, it's just a string, so I can just put the calendar format here, and ISO format, and then we'll have no, no, no, and then that would be it, things like that, you know, to ensure the system can maintain coherence over the life cycle of that system, you know, using the derivation system to create derivatives of the change, you know, if a system needs something in a bidirectional way, handling interpretations for people, especially as a first principle of that system, you know, what could a first principle be, the first thing you think of when you use a system, you know, it just works the way it is, and to mitigate fear, uncertainty, and doubt, that's the framework I've developed, so I hope you guys take this video with the utmost dignity, because this is what I'm saying, the PRP framework is really about ensuring that the consumer can act and they have coherence, it can work, it knows what it does, why it does it, and why it's acting the way it is, it doesn't hallucinate, because, you know, you can't hallucinate, because, you know, the system resolves it kind of a polygon manner, because, you know, if you have a system that represents one thing one way, and the system has an interpretation of that which represents it, it has to be coherent between the two systems, and that's where the puzzle, so remember, the puzzle for the thing is like 20 equals 25 or something like that, no, 1 equals 6 when you get one unit or something you get 6 of it on the other side, you know that's what I'm saying, because, you know, these are definitions of like a Python tuple, and a C struct, and, you know, a Python dictionary, a Python dictionary, different between a Python dictionary and a C structure, you know, they represent objects, but they're different models of objects, you know, the Python dictionary has different operations it can perform, and that's what the same thing, they're not as much in a standardised relation for the lilies of C, C structure is not equivalent to a Python dictionary, and that's what I'm basically talking about, the data models, how you model that. One more thing, the verb-noun counter-relation, you know, as a standard specification, you guys should remember this, you must remember this, it's a policy, it's a policy map, no one should mess it up, you know, enforces as a specification, you know, I want you guys to think about this properly. So if I have an error, something called a verb-noun error, 1 to 5 is my error threshold, now, that 1 to 5 is a low to high warning, 1, low, 2, low, medium, 3, medium, 4, medium, high, and 5, high.

That warning system, so after the short coherence, so basically, if I have a class, class low, low, low what? Low warning. Low warning. Low warning error.

Low warning. But this low warning, class low warning, this is like a low warning, so the verb is what? Low. Low warning.

Medium warning. Low-medium warning. Medium warning.

Medium warning. Medium-high warning. Medium-high warning and high warning.

So this is 1 to 5, just like a system. So basically, this is like a verb-noun pair, so the low is the verb, it's low, it's low, low, verb, noun, and the noun is a warning, you know, it's warning you to, that you know, and that's like a Python class, you can call it a class called low warning. It's a policy now, guys, enforce it everywhere you write a code.

So if you write a code in OBS, you've got to ensure that it's not something you bubble to, and you bubble to a low warning, a low warning. This is now represented in enum, you know, the enum transition between the low warning states, and then you go to the 6 to 11, which is part 5, and you get the exceptions model. Now, the exceptions is dual, so basically, as you can go up, you can probably get to, can be interpreted in two ways, you know, because you know, I want to show you guys the standard, so the interpretation of the exception is bidirectional, you know, so I can have a warning in one system and not have a warning in the other.

So again, what's my warning? You know, what's my information? It's not coming through. And then you go, oh, there's an exception. So the exception could be a low, low warning, a low warning exception, low exception warning, low exception warning, but you know, low warning exception, low warning error, low error warning, low error warning, low warning, basically, this ensures that the schema is flexible for you to remember what it is.

Low exception warning to a high exception warning. The goal of this is to ensure that everything you write is, you know, can be tested, can be audited. Just by looking at it, you know, auditing just means you can fix it, you can try to fix the little things, the nuances.

Auditing is just like inspecting, it's really introspection, inspecting that this code is what it is you're trying to model. You know, auditing is like, am I getting this code right? Can I see this function when I need to see it? Can I access it when I need to access it? Can I, you know, that's what I'm saying, the audit is very important for the tracing of the function calls via the functions, which is markers of the functions in that state's mind. So what I'm saying to you, the goal of this is to ensure that everything is standardised and conventional, this is a verb noun model, which is like a verb noun DAG, like a fast car, a speeding train, no, no, wrong again, okay, a shaking dog, a shivering dog, a shivering dog, that's a model, this is a shivering dog, it can be a story or something, but a shivering dog is a shivering cat, something that shivers, something that's quiver, a fiat, a large mouse, the large mouse is just a verb noun, so you've got to ensure that the system is coherent for your own concept, this is a concept for the model, so it's like the verb is what? The action to do, to act, to do, to use, to walk, to talk, to fart, to laugh, that's the verb, the verb is a doing word, and you can add a verb if you want, an adverb to do, like doing now, the adverb is just doing now, just make sure it's the verb, because the adverb is just doing something in the past, present or future, there's constraints on it, you can use it to log errors,