MICHAEL TIMOTHY BENNETT

TECHNICAL APPENDICES

VERSION 4.0

MATH AND EXPERIMENTS

HERE I PRESENT A COLLECTION of mathematical proofs and experimental results pertaining to optimal learning, complexity, and a mathematical formalism of philosophical concepts including Gricean pragmatics, access and phenomenal consciousness. While some of this content has appeared in other publications substantial improvements have been made to both the formalism and the proofs, along with error corrections. Hence it is both a self contained paper, and a compilation of material.

STACK THEORY

DEFINITIONS 1 TO 4 are taken from this ref[1]:

[1] Michael Timothy Bennett. Computational dualism and objective superintelligence. In *Artificial General Intelligence*. Springer Nature, 2024a

Definition 1 (environment)

- *We assume a set Φ whose elements we call **states**.*
- *A **declarative program** is $f \subseteq \Phi$, and we write P for the set of all declarative programs (the powerset of Φ).*
- *By a **truth** or **fact** about a state ϕ, we mean $f \in P$ such that $\phi \in f$.*
- *By an **aspect of a state** ϕ we mean a set l of facts about ϕ s.t. $\phi \in \bigcap l$. By an **aspect of the environment** we mean an aspect l of any state, s.t. $\bigcap l \neq \emptyset$. We say an aspect of the environment is **expressed**, **realised**[2] or **embodied** in state ϕ if it is an aspect of ϕ.*

[2] Realised meaning it is made real, or brought into existence.

Definition 2 (abstraction layer)
By abstraction layer[3] I mean:

[3] (NOTATION) E with a subscript is the extension of the subscript. For example, E_l is the extension of l.

(INTUITIVE SUMMARY) $L_{\mathfrak{v}}$ is everything which can be realised in this abstraction layer. The extension E_x of a statement x is the set of all statements whose existence implies x, and so it is like the sub-table of x's truth table for which x is true.

- *We single out a subset $\mathfrak{v} \subseteq P$ which we call **the vocabulary** of an abstraction layer. The vocabulary is finite unless explicitly stated otherwise. If $\mathfrak{v} = P$, then we say that there is no abstraction.*
- *$L_{\mathfrak{v}} = \{l \subseteq \mathfrak{v} : \bigcap l \neq \emptyset\}$ is a set of aspects in $\mathfrak{v}$. We call $L_{\mathfrak{v}}$ a formal language, and $l \in L_{\mathfrak{v}}$ a **statement**.*
- *We say a statement is **true** given a state iff it is an aspect realised by that state.*
- *A **completion** of a statement x is a statement y which is a superset of x. If y is true, then x is true.*
- *The **extension of a statement** $x \in L_{\mathfrak{v}}$ is $E_x = \{y \in L_{\mathfrak{v}} : x \subseteq y\}$. E_x is the set of all completions of x.*
- *The **extension of a set of statements** $X \subseteq L_{\mathfrak{v}}$ is $E_X = \bigcup_{x \in X} E_x$.*
- *We say x and y are **equivalent** iff $E_x = E_y$.*

Definition 3 (abstractor function) $\mathfrak{f} : 2^P, 2^P \rightarrow 2^P$ *is an **abstractor** function that takes a vocabulary $\mathfrak{v}$ and a statement $l \subset \mathfrak{v}$, and returns a new vocabulary $\mathfrak{v}' = \{f \in P : \exists o \in E_l (\bigcap o = f)\}$.*

Definition 4 (Time) *Time is the ordered sequence of transitions between distinct states of the environment, where each state $\phi \in \Phi$ is a full snapshot of reality at a given tick.*

Definition 5 (Persistence) *An aspect l persists across time if there's a sequence of states $\phi_1, \phi_2, \ldots, \phi_n$ where each ϕ_i has a statement in l's extension E_l that's expressed.*

Definition 6 ($\mathfrak{v}$-task)
For a chosen $\mathfrak{v}$, a task α is a pair $\langle I_\alpha, O_\alpha \rangle$ where[4]*:*

- *$I_\alpha \subset L_\mathfrak{v}$ is a set whose elements we call **inputs** of α.*
- *$O_\alpha \subset E_{I_\alpha}$ is a set whose elements we call **correct outputs** of α.*

*I_α has the extension E_{I_α} we call **outputs**, and O_α are outputs deemed correct. $\Gamma_\mathfrak{v}$ is the set of **all tasks** given $\mathfrak{v}$.*

(GENERATIONAL HIERARCHY) *A $\mathfrak{v}$-task α is a **child** of $\mathfrak{v}$-task ω if $I_\alpha \subset I_\omega$ and $O_\alpha \subseteq O_\omega$. This is written as $\alpha \sqsubset \omega$. If $\alpha \sqsubset \omega$ then ω is then a **parent** of α. $\sqsubset$ implies a "lattice" or generational hierarchy of tasks. Formally, the level of a task α in this hierarchy is the largest k such there is a sequence $\langle \alpha_0, \alpha_1, ...\alpha_k \rangle$ of k tasks such that $\alpha_0 = \alpha$ and $\alpha_i \sqsubset \alpha_{i+1}$ for all $i \in (0, k)$. A child is always "lower level" than its parents*[5]*.*

[4] (NOTATION) If $\omega \in \Gamma_\mathfrak{v}$, then we will use subscript ω to signify parts of ω, meaning one should assume $\omega = \langle I_\omega, O_\omega \rangle$ even if that isn't written.

(INTUITIVE SUMMARY) To reiterate and summarise the above:

- An **input** is a possibly incomplete description of a world.
- An **output** is a completion of an input [see def. of $\mathfrak{v}$-task].
- A **correct output** is a correct completion of an input.

[5] (FURTHER INTUITIVE SUMMARY) A $\mathfrak{v}$-task is a formal, **behavioural** description of an aspect of the environment. For example, a self-organising biological system could be described as a task α enumerating all behaviour in which it remains alive. It begins alive in circumstances given by inputs I_α, and remains alive in circumstances given by outputs O_α, and is dead in circumstances given by $E_{I_\alpha} - O_\alpha$. Likewise, we could describe the game chess played from the perspective of white. We could say Φ contains a state corresponding to each and every move of each and every possible game of chess, I_α contains every possible sequence of moves in which the game has not ended and it remains *possible* for white to win, and O_α contains every possible sequence ending in a move that means white *has* won. Tasks are *behavioural* descriptions of systems in the philosophical sense of the word, and we will next relate these ideas to machine functionalism.

PANCOMPUTATIONAL ENACTIVISM

MACHINE LEARNING is typically divided into learning and inference. To learn a task, an organism learns a policy (strictly speaking, an organism *is* a policy). By constraining how we complete inputs, a policy allows us to do abductive inference (much as a constraint in a SAT solver would). Inference requires a **policy** and learning a policy requires a **proxy**, the definitions of which follow. Stack Theory is the idea that everything is an infinite state of abstraction layers. Pancomputational Enactivism is the formalisation of enactivism within Stack Theory.

Definition 7 (inference)

- *A $\mathfrak{v}$-task **policy** is a statement $\pi \in L_{\mathfrak{v}}$. It constrains how we complete inputs.*
- *π is a **correct policy** iff the correct outputs O_α of α are exactly the completions π' of π such that π' is also a completion of an input.*
- *The set of all correct policies for a task α is denoted Π_α.*[6]

Assume $\mathfrak{v}$-task ω and a policy $\pi \in L_{\mathfrak{v}}$. Inference[7] *proceeds as follows:*

1. *we are presented with an input $i \in I_\omega$, and*
2. *we must select an output $e \in E_i \cap E_\pi$.*
3. *If $e \in O_\omega$, then e is correct and the task "complete". $\pi \in \Pi_\omega$ implies $e \in O_\omega$, but $e \in O_\omega$ doesn't imply $\pi \in \Pi_\omega$ (an incorrect policy can imply a correct output).*

[6] To repeat the above definition in set builder notation:

$$\Pi_\alpha = \{\pi \in L_{\mathfrak{v}} : E_{I_\alpha} \cap E_\pi = O_\alpha\}$$

[7] (INTUITIVE SUMMARY) To reiterate and summarise the above:

- A **policy** constrains how we complete inputs.
- A **correct policy** is one that constrains us to correct outputs.

Definition 8 (λ-tasks)
*The set of all tasks with no abstraction (meaning $\mathfrak{v} = P$) is Γ_P (it contains every task in every vocabulary). For every P-task $\rho \in \Gamma_P$ there exists a function $\lambda_\rho : 2^P \rightarrow \Gamma_P$ that takes a vocabulary $\mathfrak{v}' \in 2^P$ and returns a highest level child $\omega \sqsubset \rho$ which is also a $\mathfrak{v}'$-task. We call λ_ρ an **uninstantiated-task**, and $\lambda_1 \sqsubset \lambda_2$ iff $\lambda_1(P) \sqsubset \lambda_2(P)$.*

Definition 9 (learning) *Learning is a collection of definitions that describe the process by which a policy is constructed by any system*[8].

- *A **proxy** $<$ is a binary relation on statements, and the set of all proxies is Q.*
- *$<_w$ is the **weakness** proxy*[9]*. For statements l_1, l_2 we have $l_1 <_w l_2$ iff $|E_{l_1}| < |E_{l_2}|$.*
- *$<_d$ is the **description length** or **simplicity** proxy*[10]*. We have $l_1 <_d l_2$ iff $|l_1| > |l_2|$.*

[8] (INTUITIVE SUMMARY) Learning is an activity undertaken by an adaptive system, and a task has been **learned** by a system that embodies a correct policy. Humans typically learn from **examples**. An example of a task is a correct output and input.

[9] By the weakness of a statement, we mean the cardinality of its extension. By the weakness of an extension we mean its cardinality.

[10] When we speak of simplicity with regards to a policy $\pi \in \Pi_\alpha$ we mean the cardinality of the smallest correct policy $\pi' \in \Pi_\alpha$ s.t. $E_{\pi'} = E_\pi$. The complexity of an extension is the **simplest** statement of which it is an extension.

(GENERALISATION) *A statement l **generalises** to a $\mathfrak{v}$-task α iff $l \in \Pi_\alpha$. We speak of **learning** ω from α iff, given a proxy $<$, $\pi \in \Pi_\alpha$ maximises $<$ relative to all other policies in Π_α, and $\pi \in \Pi_\omega$.*

(PROBABILITY OF GENERALISATION) *We assume a uniform distribution over $\Gamma_\mathfrak{v}$. If l_1 and l_2 are policies, we say it is less probable that l_1 generalizes than that l_2 generalizes, written $l_1 <_g l_2$, iff, when a task α is chosen at random from $\Gamma_\mathfrak{v}$ (using a uniform distribution) then the probability that l_1 generalizes to α is less than the probability that l_2 generalizes to α.*

(EFFICIENCY) *Suppose*[11] *$\mathfrak{app}$ is the set of **a**ll **p**airs of **p**olicies. Assume a proxy $<$ returns 1 iff true, else 0. Proxy $<_a$ is more efficient than $<_b$ iff*

$$\left(\sum_{(l_1,l_2)\in\mathfrak{app}} |(l_1 <_g l_2) - (l_1 <_a l_2)| - |(l_1 <_g l_2) - (l_1 <_b l_2)| \right) < 0$$

[11] (FURTHER INTUITIVE SUMMARY) A collection of examples is a child task, so learning is an attempt to generalise from a child, to one of its parents. The lower level the child from which an agent generalises to parent, the 'faster' it learns, the more sample efficient the proxy.

(OPTIMAL PROXY) *There is no proxy more efficient than weakness. The weakness proxy formalises the idea that "explanations should be no more specific than necessary" (see Bennett's razor in this ref*[12]*).*

[12] Michael Timothy Bennett. The optimal choice of hypothesis is the weakest, not the shortest. In *Artificial General Intelligence*. Springer Nature, 2023a

(INTUITIVE SUMMARY) *Learning is an activity undertaken by some manner of intelligent agent, and a task has been "learned" by an agent that knows a correct policy. Humans typically learn from "examples". An example of a task is a correct output and input. A collection of examples is a child task, so "learning" is an attempt to generalise from a child to one of its parents. The lower level the child from which an agent generalises to parent, the "faster" it learns (it chooses policies that complete a wider variety of tasks, and thus are more sample and energy efficient choices), the more efficient the proxy. The most efficient proxy is weakness (see proofs 1 and 2, or these refs*[13]*), which is why we're using it here.*

[13] Michael Timothy Bennett. The optimal choice of hypothesis is the weakest, not the shortest. In *Artificial General Intelligence*. Springer Nature, 2023a; and Michael Timothy Bennett. Are biological systems more intelligent than artificial intelligence? Forthcoming in *Philosophical Transactions of The Royal Society B, Special Issue on Hybrid Agencies*, 2025a

PHILOSOPHICAL AND BIOLOGICAL ANALOGUES

Definition 10 (multilayer architecture) *The multilayer architecture (MLA) found in both biological systems and computers. It integrates a stack with tasks to represent natural selection or 'correctness' at different layers of abstraction.*

- *The **stack** is represented here by a sequence of uninstantiated tasks* $\langle \lambda^0, \lambda^1 ... \lambda^n \rangle$ *s.t* $\lambda^{i+1} \sqsubset \lambda^i$.
- $\mathfrak{f}$ *is an **abstractor** function.*
- *The **state** of the MLA is a sequence of policies* $\langle \pi^0, \pi^1 ... \pi^n \rangle$ *and a sequence of vocabularies* $\langle \mathfrak{v}^0, \mathfrak{v}^1 ... \mathfrak{v}^n \rangle$ *such that* $\mathfrak{v}^{i+1} = \mathfrak{f}(\mathfrak{v}^i, \pi^i)$ *and* $\pi^i \in \Pi_{\lambda^i(\mathfrak{v}^i)}$.

In the absence of abstraction where the system is seen as nothing more than the sum of its parts, the MLA is just a task $\lambda^0(\mathfrak{v}^0)$*, allowing us to look at the system across scales of distribution. We say the MLA is **over-constrained** when there exists* $i < n$ *s.t. and* $\Pi_{\lambda^i(\mathfrak{v}^i)} = \emptyset$*, and **multilayer-causal-learning** (MCL) occurs when the MLA is not over-constrained and the proxy for learning is weakness. Note that the vocabulary is different at each level in the stack, which means each has its own generational hierarchy of tasks. By a higher level of abstraction, we mean a task higher in the stack (later in the causal chain).*

Definition 11 (utility of intelligence)
Every task $\gamma \in \Gamma$ *has a "utility of intelligence"*[14] *computed as* $\epsilon : \Gamma \rightarrow \mathbb{N}$ *such that* $\epsilon(\gamma) = \max_{m \in \Pi_\gamma} (|E_m| - |O_\gamma|)$*. Maximisation of utility means maximising* $\bullet \overset{\epsilon}{<} \bullet$ *that returns true iff* $\epsilon(\alpha) < \epsilon(\omega)$.

[14] Assuming we accept that intelligence is the ability to generalise, then we can measure the utility of selecting policies in accord with Bennett's Razor by measuring the weakness of the weakest policy for a task. Tasks with weaker policies make more use of intelligence.

CAUSALITY

ORIGINAL DEFINITIONS are to be found in this ref[15].

[15] Michael Timothy Bennett. Emergent causality and the foundation of consciousness. In *Artificial General Intelligence*. Springer Nature, 2023b

Definition 12 (intervention)
Intuitively, if int and obs are "events" which have happened, then we say that int has ***caused*** *obs if obs would not have happened in the absence of int (counterfactual). In our formalism, an* ***event*** *is a statement in* $L_{\mathfrak{v}}$*, and an event* ***happens*** *or is* ***observed*** *iff it is a true statement. If* $obs \in L_{\mathfrak{v}}$ *is sensorimotor activity we interpret as an "observed event", and* $int \in L_{\mathfrak{v}}$ *is in* ***intervention*** *(by an organism or other agency, in the sense described by Pearl[16]) to cause that event, then* $obs \subset int$ *(because int could not be said to cause obs unless* $obs \subset int$*).*

[16] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, Inc., New York, 1st edition, 2018

Definition 13 (causal identity) *If[17]* $obs \in L_{\mathfrak{v}}$ *is an observed event, and* $int \in L_{\mathfrak{v}}$ *is in intervention causing obs, then intuitively* $c \subseteq int - obs$ *"identifies" or "names" the intervening agency if* $c \neq \emptyset$*. We call c a* ***causal identity*** *corresponding to int and obs. Suppose* INT *and* OBS *are sets of statements, and we assume* OBS *contains observed events and* INT *interventions, then a causal identity corresponding to* INT *and* OBS *is* $c \neq \emptyset$ *s.t.* $\forall i \in INT (c \subset int)$ *and* $\forall obs \in OBS (c \cap obs = \emptyset)$ *(we can attempt to construct a causal identity for any* INT *and* OBS*). If a policy is a causal identity, then the associated task is to classify interventions.*

[17] (EXAMPLE) Suppose we have organisms $\mathfrak{a}$ (Alice) and $\mathfrak{b}$ (Bob). The inputs Alice has experienced so far $I_{\mathfrak{h}_{<t_{\mathfrak{a}}}}$ can be divided into those in which Bob affected Alice $I_{\mathfrak{a}}^{\mathfrak{b}}$ and those in which Bob did not $I_{\mathfrak{a}}^{\neg\mathfrak{b}} = I_{\mathfrak{h}_{<t_{\mathfrak{a}}}} - I_{\mathfrak{a}}^{\mathfrak{b}}$. By affecting Alice, Bob has intervened in Alice's experience. Alice can construct a causal identity b for Bob corresponding to interventions $INT = I_{\mathfrak{a}}^{\mathfrak{b}}$ and observations $OBS = I_{\mathfrak{a}}^{\neg\mathfrak{b}}$. The objects which "exist" in Alice's experience are those for which she constructs a causal identity, so this is how Bob comes to exist as a distinct "object" which Alice experiences, rather than in parts of other objects).

Definition 14 (preconditions)
If $\mathfrak{o}$ *is an organism, and c is a causal identity:*

- *the* ***representation*** *precondition is met iff* $c \in L_{\mathfrak{v}_{\mathfrak{o}}}$*, and*
- *the* ***incentive*** *precondition is met if* $\mathfrak{o}$ *must learn c to remain "fit"[18].*

[18] It is possible an organism might construct c even if it is not required for the organism to remain fit, hence 'if' instead of 'iff'. Incentive is a sufficient predondition in conjunction with representation, but it is not strictly necessary.

Definition 15 (purpose, goal or intent)
We consider a policy c which is a causal identity corresponding to INT *and* OBS *to be the* ***intent***, ***purpose*** *or* ***goal*** *ascribed to the interventions. c is what the interventions share in common, meaning the "name" or "identity" of behaviour is the "intent", "goal" or "purpose" of behaviour. Just as an intervention caused an observation, the particular intent which motivated the agency undertaking the intervention is what caused it (to correctly infer intent, one must infer a causal identity that implies subsequent interventions).*

Definition 16 (first order self)
*If c is the lowest level causal identity corresponding to INT and OBS, and INT is every intervention an organism could make (not just past interventions, but all potential future interventions), then we consider c to be the system's **first order self**. If $c \in \mathfrak{p}_{\mathfrak{o}}$ then an organism has constructed a first order self. A first order self for an organism $\mathfrak{o}$ is denoted $\mathfrak{o}^1$. An organism has at most one first order self.*

Definition 17 (chain notation)
Suppose we have two organisms, $\mathfrak{a}$ (Alice) and $\mathfrak{b}$ (Bob). $c_{\mathfrak{a}}^{\mathfrak{b}}$ denotes a causal identity for $\mathfrak{b}$ constructed by $\mathfrak{a}$ (what Alice thinks Bob intends). Subscript denotes the organism who constructs the causal identity, while superscript denotes the object. The superscript can be extended to denote chains of predicted causal identity. For example, $c_{\mathfrak{a}}^{\mathfrak{ba}} \subset c_{\mathfrak{a}}^{\mathfrak{b}}$ denotes $\mathfrak{a}$'s prediction of $\mathfrak{b}$'s prediction of $\mathfrak{a}^1$ (what Alice thinks Bob thinks Alice intends). The superscript of $c_{\mathfrak{a}}^{*}$ can be extended indefinitely to indicate recursive predictions, however the extent recursion is possible is determined by $\mathfrak{a}$'s vocabulary $\mathfrak{v}_{\mathfrak{a}}$. Finally, Bob need not be an organism. Bob can be anything for which Alice constructs a causal identity.

Definition 18 ($\mathbf{n^{th}}$ order self)
An n^{th} order self for $\mathfrak{a}$ is $\mathfrak{a}^{\mathfrak{n}} = c_{\mathfrak{a}}^{*\mathfrak{a}}$ where $*$ is replaced by a chain, and n denotes the number of reflections. For example, a second order self $\mathfrak{a}^2 = c_{\mathfrak{a}}^{\mathfrak{ba}}$, and a third order self $\mathfrak{a}^3 = c_{\mathfrak{a}}^{\mathfrak{baba}}$. We use $\mathfrak{a}^2$ to refer to any second order self, and chain notation to refer to a specific second order self, for example $c_{\mathfrak{a}}^{\mathfrak{ba}}$. The union of two n^{th} order selves is also considered to be an n^{th} order self, for example $\mathfrak{a}^3 = c_{\mathfrak{a}}^{\mathfrak{baba}} \cup c_{\mathfrak{a}}^{\mathfrak{dada}}$, and the weaker or higher level a self is in the generational hierarchy, the more selves there are of which it is part.

SEMIOSIS

Original definitions are to be found in this ref[19].

[19] Michael Timothy Bennett. On the computation of meaning, language models and incomprehensible horrors. In *Artificial General Intelligence*. Springer Nature, 2023c

Definition 19 (organism)
I describe the circumstances of an organism[20] $\mathfrak{o}$ *as* $\langle \mathfrak{v}_\mathfrak{o}, \mu_\mathfrak{o}, \mathfrak{p}_\mathfrak{o}, <_\mathfrak{o} \rangle$ *where:*

[20] (INTUITIVE SUMMARY) Strictly speaking an organism $\mathfrak{o}$ would be a policy, but we can describe the circumstances of its existence as a task μ that describes all "fit" behaviour for that organism. We can also identify policies the organism "knows", because these are implied by the policy that is the organism. Likewise, we can represent lossy memory by having the organism "know" fewer policies than are implied by its history of interactions. Finally, preferences are the particular "protosymbol" the organism will use to "interpret" an input in later definitions.

- $O_{\mu_\mathfrak{o}}$ *contains every output which qualifies as "fit" according to natural selection.*
- $\mathfrak{p}_\mathfrak{o}$ *is the set of policies an organism knows, s.t.* $\mathfrak{p}_\mathfrak{o} \subset \mathfrak{p}_{n.s.} \cup \mathfrak{p}_{\mathfrak{h}_{<t_\mathfrak{o}}}$ *and:*
  - $\mathfrak{p}_{n.s.} \subset L_{\mathfrak{v}_\mathfrak{o}}$ *is* ***reflexes*** *hard coded from birth by natural selection.*
  - $\mathfrak{p}_{\mathfrak{h}_{<t_\mathfrak{o}}} = \bigcup_{\zeta \in \mathfrak{h}_{<t_\mathfrak{o}}} \Pi_\zeta$ *is the set of policies it is possible to* ***learn*** *from a history of past interactions represented by a task* $\mathfrak{h}_{<t_\mathfrak{o}}$.
  - *If* $\Pi_{\mathfrak{h}_{<t_\mathfrak{o}}} \not\subset (\mathfrak{p}_\mathfrak{o} - \mathfrak{p}_{n.s.})$ *then the organism has* ***selective memory****. It can "forget" outputs, possibly to productive ends if they contradict otherwise good policies.*
- $<_\mathfrak{o}$ *is a binary relation over* $\Gamma_{\mathfrak{v}_\mathfrak{o}}$ *we call* ***preferences****.*

Definition 20 (protosymbol system)
Assume an organism $\mathfrak{o}$. *For each policy* $p \in \mathfrak{p}_\mathfrak{o}$ *there exists a set* $\mathfrak{s}_p = \{\alpha \in \Gamma_{\mathfrak{v}_\mathfrak{o}} : p \in \Pi_\alpha\}$ *of all tasks for which* p *is a correct policy. The union of all such sets is*

$$\mathfrak{s}_\mathfrak{o} = \bigcup_{p \in \mathfrak{p}_\mathfrak{o}} \{\alpha \in \Gamma_{\mathfrak{v}_\mathfrak{o}} : p \in \Pi_\alpha\}$$

We call $\mathfrak{s}_\mathfrak{o}$ *a "protosymbol system". A* $\mathfrak{v}$*-task* $\alpha \in \mathfrak{s}_\mathfrak{o}$ *is called a "protosymbol".*

Definition 21 (affect)
Suppose we have two organisms, $\mathfrak{a}$ *(Alice) and* $\mathfrak{b}$ *(Bob). Suppose* $\mathfrak{a}$ *interprets* $i \in L_{\mathfrak{v}_\mathfrak{o}}$ *as an output* o*, then:*

- *a* ***statement*** $v \subset i$ *affects* $\mathfrak{a}$ *if* $\mathfrak{a}$ *would have interpreted* $e = i - v$ *as a different output* $g \neq o$.
- *an* ***organism*** $\mathfrak{b}$ *has affected* $\mathfrak{a}$ *by making an output* k *if, as a consequence of* k*, there exists* $v \subset s$ *which affects* $\mathfrak{a}$.

Definition 22 (interpretation)
Interpretation is inference, with the additional step of choosing a policy according to preference. Interpretation is an activity undertaken by an organism $\mathfrak{o}$. *It proceeds as follows:*

1. *Assume true input* $i \in L_{\mathfrak{v}_{\mathfrak{o}}}$ *(meaning* $i = i_t$ *in an EGRL system at time* t*).*
2. *We say that* i ***signifies*** *a protosymbol* $\alpha \in \mathfrak{s}_{\mathfrak{o}}$ *if* $i \in I_\alpha$*.*
3. $\mathfrak{s}^s_{\mathfrak{o}} = \{\alpha \in \mathfrak{s}_{\mathfrak{o}} : i \in I_\alpha\}$ *is the set of all protosymbols which* i *signifies.*
4. *If* $\mathfrak{s}^s_{\mathfrak{o}} \neq \emptyset$ *then* i ***means something*** *to the organism in the sense that there is "value" ascribed to symbols in* $\mathfrak{s}^s_{\mathfrak{o}}$ *compelling the organism to act.*
5. *If* i *means something, then* $\mathfrak{o}$ *chooses* $\alpha \in \mathfrak{s}^s_{\mathfrak{o}}$ *that maximises its preferences* $<_{\mathfrak{o}}$*.*
6. *The organism then infers an output* $o \in E_s \cap E_{\Pi_\alpha}$*.*

MEANING

ORIGINAL DEFINITIONS are to be found in this ref[21]

Definition 23 (meaning)
*The **meaning** an organism $\mathfrak{o}$ ascribes to an input i is a protosymbol $\alpha \in \mathfrak{s}_\mathfrak{o}$ which $\mathfrak{o}$ uses to interpret i. Symbols in different protosymbol systems can be **roughly equivalent** (result in similar behaviour etc), in accord with the philosophical arguments in the body of the paper[22]. We use $\omega \approx \alpha$ to indicate that ω and α are roughly equivalent. In accord with earlier definitions, one organism "intends" to affect another if completion of the protosymbol (a task) the former uses to interpret hinges on how the latter's behaviour is affected.*
Assume an organism $\mathfrak{a}$ (Alice) in input $i_\mathfrak{a}$ and organism $\mathfrak{b}$ (Bob) in $i_\mathfrak{b}$. $\mathfrak{a}$ means $\alpha \in \mathfrak{s}_\mathfrak{a}$ by deciding $u \in E_{i_\mathfrak{a}}$ iff $\mathfrak{a}$ intends in deciding u:

1. *that $\mathfrak{b}$ interpret $i_\mathfrak{b}$ using $\omega \approx \alpha$,*
2. *$\mathfrak{b}$ "recognize" this intention by being affected by u such that the input $i_\mathfrak{b} = j$ in which $\mathfrak{b}$ finds itself change to $i_\mathfrak{b} = k \neq j$,*
3. *and (1) occur on the basis of (2), meaning had $\mathfrak{a}$ not decided u then $\mathfrak{b}$ would have interpreted $i_\mathfrak{b}$ using $\zeta \not\approx \alpha$.*

To communicate meaning organisms must:

1. *be able to affect one another.*
2. *have similar experiences, so $\mathfrak{s}_\mathfrak{b}$ and $\mathfrak{s}_\mathfrak{a}$ contain roughly equivalent symbols.*
3. *have similar preferences.*

[21] Michael Timothy Bennett. On the computation of meaning, language models and incomprehensible horrors. In *Artificial General Intelligence*. Springer Nature, 2023c

[22] We argue organisms of the same species construct roughly equivalent protosymbols, even though each member of a species exists in its own unique abstraction layer with its own protosymbol system.

CONSCIOUSNESS

THE ORIGINAL definitions of these can be found in this ref[23]. A first order self correctly classifies the system's interventions, just as the neuroscientific phenomenon[24] known as "reafference" classifies interventions in living organisms. A first order self allows a system to reason about its own future interventions, plan spatial navigation and so forth.

[23]

[24] Andrew B. Barron and Colin Klein. What insects can tell us about the origins of consciousness. *Proceedings of the National Academy of Sciences*, 2016

Definition 24 (levels of consciousness)

1. *an organism that acts but does not learn, meaning* $\mathfrak{p}_\mathfrak{o}$ *is fixed from birth.*
2. *an organism that learns, but* $\mathfrak{o}^1 \notin \mathfrak{p}_\mathfrak{o}$ *either because* $\mathfrak{o}^1 \notin L_{\mathfrak{v}_\mathfrak{o}}$ *(failing the "representation precondition") or because the organism is not incentivised to construct* $\mathfrak{o}^1$ *(failing the "incentive precondition").*
3. *reafference and phenomenal or core consciousness are achieved when* $\mathfrak{o}^1 \in \mathfrak{p}_\mathfrak{o}$ *is learned by an organism as a consequence of attraction to and repulsion from statements in* $L_{\mathfrak{v}_\mathfrak{o}}$*.*
4. (a) *access or self reflexive consciousness is achieved when* $\mathfrak{o}^2 \in \mathfrak{p}_\mathfrak{o}$*.*
   (b) *hard consciousness[25] is achieved when a phenomenally conscious organism learns a second order self (an organism is consciously aware of the contents of second order selves, which must have quality if learned through phenomenal conscious).*
5. *meta self reflexive consciousness (human level hard consciousness) is achieved when* $\mathfrak{o}^3 \in \mathfrak{p}_\mathfrak{o}$*.*

[25] Piotr Boltuc. The engineering thesis in machine consciousness. *Techné: Research in Philosophy and Technology*, 2012

ILLUSTRATIVE EXAMPLES

FOR THE SAKE OF INTUITION I will now provide some some simple examples of how these definitions might be applied in the context of general reinforcement learning. In general reinforcement learning (GRL) $\mathcal{A}$,$\mathcal{O}$,$\mathcal{R}$ denote the (finite) action, observation and reward set respectively. The following definition of G.R.L. was taken from this ref[26].

- $\mathcal{E} := \mathcal{O} \times \mathcal{R}$ is the set of **percepts**.
- $\Delta\mathcal{X}$ is the set of distributions over $\mathcal{X}$.
- $\mathcal{H} := (\mathcal{A} \times \mathcal{E})^*$ is the set of **histories**, where $*$ signifies that histories can be of any length (for example, were we to type $(\mathcal{A} \times \mathcal{E})^3$ would just have all histories of length 3).
- $\pi : \mathcal{H} \to \Delta\mathcal{A}$ is a function that outputs distributions over actions given a history. π is called a **policy**.
- $\mu : \mathcal{H} \times \mathcal{A} \to \Delta\mathcal{E}$ is an **environment**. An environment is a distribution over precepts *given* a history, and an action.
- $h_{<t} := a_1e_1...a_{t-1}e_{t-1} = a_1o_1r_1...a_{t-1}o_{t-1}r_{t-1}$ denotes a history of interactions up to time $t-1$ inclusive.
- Given an environment μ and policy π, the conjunction $\mu\pi$ denotes the induced probability measure on histories, where $h_{<t}$ is assumed to be sampled from $\mu\pi$ (intuitively, an environment does the opposite of a policy, predicting the outcome of an action rather than which action comes next).
- GRL is concerned with how well policies perform (and by extension agents). We measure this performance by the future expected sum of discounted rewards, called value function. The agent selects a policy that maximises the value function.

Note that though it is not explicitly stated, the above definition assumes reward is between 0 and 1.

[26] Elliot Catt, Jordi Grau-Moya, Marcus Hutter, Matthew Aitchison, Tim Genewein, Gregoire Deletang, Li Kevin Wenliang, and Joel Veness. Self-predictive universal AI. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=psXVkKO9No`

GRL AND PANCOMPUTATIONAL ENACTIVISM

HERE I REPRESENT THE DIFFERENT INGREDIENTS of GRL using $\mathfrak{v}$-tasks and an abstraction layer. This is just for illustrative purposes, to give some intuition about how this all works. It is one possible way to translate GRL into the context of my formalism, but is by no means the only way. The most important difference to note is that there is no need to append new outputs to the history. This is because it is an *embodied* formalism. The "input" is not just a stream from sensors, it is the state of the computer; including memory and the surrounding environment. The input at every step therefore includes memory implicitly. These are examples of how these definitions might be applied. I am not asserting that an EGRL system as described below is conscious. It leaves many details unresolved.

Definition 25 (EGRL)
Assume a vocabulary $\mathfrak{v}$, a function $r : L_\mathfrak{v} \rightarrow \{1, 0\}$ and an integer $len > 0$. The first two imply a $\mathfrak{v}$-task μ such that $O_\mu = \{l \in L_\mathfrak{v} : r(l) = 1\}$ and $\Pi_\mu \neq \emptyset$[27]. O_μ is every correct output and I_μ is every input in which an output can be made. Together these imply several other objects.

- *We assume sequence of timesteps from 0 to len inclusive, and that there is a current **timestep** t. At each timestep there is a new "current state" of the environment, which means we get a new input.*
- *We single out a $\mathfrak{v}$-task $\mathfrak{h}_{<t} \in \Gamma_\mathfrak{v}$ of outputs made leading up to time t is a $\mathfrak{v}$-task. We call $\mathfrak{h}_{<t}$ the **history** until time t. As reward is implicit in the task definition, r is applied to $O_{\mathfrak{h}_{<t}}$ to obtain a pair of $\mathfrak{v}$-tasks $(\mathfrak{h}^1_{<t}, \mathfrak{h}^0_{<t}) \in \Gamma_\mathfrak{v} \times \Gamma_\mathfrak{v}$ s.t. $O_{\mathfrak{h}^1_{<t}} \subseteq O_\mu$ and $O_{\mathfrak{h}^0_{<t}} \cap O_\mu = \emptyset$ (meaning $\mathfrak{h}^1_{<t}$ is a history of correct outputs preceeding t, and $\mathfrak{h}^0_{<t}$ is a history of incorrect outputs).*
- *A **pair proxy** $<^{\mathfrak{pp}}$ is a binary relation on pairs in $\mathfrak{pp} = \{(m, n) \in \Pi_{\mathfrak{h}^1_{<t}} \times \Pi_{\mathfrak{h}^0_{<t}} : E_m \cap E_n = \emptyset\}$ that classify correct and incorrect outputs. For such pairs, **learning** means choosing $(\pi, n) \in \mathfrak{pp}$ that maximises $<^{\mathfrak{pp}}$, and **inference** means choosing $o \in E_{i_t} \cap E_\pi$.*
- *Given $(\pi, n), (j, k) \in \mathfrak{pp}$ "weakness" pair proxy is $(j, k) <^{\mathfrak{pp}}_w (\pi, n)$ iff $\left|E_j \cup E_k\right| < |E_\pi \cup E_n|$. Intuitively and with some abuse of notation, this is because (vocabulary permitting) there can exist a solitary statement j which has the equivalent extension, meaning $E_\pi \cup E_n = E_j$. j is a weakest policy implied by all past outputs, rather than only correct outputs. Whether or not $j \subset \mathfrak{v}$, it remains the optimal choice according to $<_w$, which means (π, n) is also optimal because it has the equivalent extension.*

For convenience going forward I'll refer to $\langle \mathfrak{v}, \mu, len \rangle$ as an EGRL problem. It is solved by an EGRL system, the pseudocode of which is given below in the algorithm. To run an an EGRL system and solve an EGRL we require $\langle \mathfrak{v}, \mathfrak{h}_{<t}, <^{\mathfrak{pp}}, r \rangle$, where $<^{\mathfrak{pp}}$ is a pair proxy and $\mathfrak{h}_{<t}$ is a history (a $\mathfrak{v}$-task).

[27] This is because we're trying to offer a computable and objectively optimal alternative to AIXI, which means we want lossless interpolation. If $\Pi_\mu \neq \emptyset$, then we'd need to consider lossy approximations by "selective forgetting" of past outputs inconsistent with otherwise good hypotheses.

Algorithm 1: EGRL SYSTEM

If I is a set, $I.choose()$ returns a member of I, and $I.add(i) := I \cup \{i\}$
The $.choose()$ method assumes a total order or distribution
$\mathfrak{r}_{\mathfrak{h}_{<t}}$ is an int variable for the reward accrued before time t

function EGRL_LOOP(μ, $<^{\mathfrak{pp}}$, r, len)
    $i_t \leftarrow I_\mu.choose()$
    $t, \mathfrak{r}_{\mathfrak{h}_{<t}} \leftarrow 0$
    $\mathfrak{h}_{<t} \leftarrow \langle \{i_t\}, \emptyset \rangle$
    while $t < len$ **do**
        $\pi \leftarrow$ LEARNING$(\mathfrak{h}_{<t}, <^{\mathfrak{pp}}, i_t, r)$ ▷ See LEARNING METHODS
        $o_t \leftarrow$ INFERENCE(π, i_t)
        $\mathfrak{r}_{\mathfrak{h}_{<t}} \leftarrow \mathfrak{r}_{\mathfrak{h}_{<t}} + r(o_t)$ ▷ feedback
        $t, i_t, \mathfrak{h}_{<t} \leftarrow$ NEXT_TIMESTEP$(t, I_\mu, \mathfrak{h}_{<t})$
    end while
    return $\mathfrak{r}_{\mathfrak{h}_{<t}}$
end function

function INFERENCE(π, i_t)
    $E_{\pi \cap i_t} \leftarrow E_\pi \cap E_{i_t}$
    return $E_{\pi \cap i_t}.choose()$
end function

function NEXT_TIMESTEP(t,I_μ, $\mathfrak{h}_{<t}$)
    $t \leftarrow t + 1$
    $O_{\mathfrak{h}_{<t}} \leftarrow O_{\mathfrak{h}_{<t}}.add(o_t)$
    $I_{\mathfrak{h}_{<t}} \leftarrow I_{\mathfrak{h}_{<t}}.add(i_t)$
    return t, $I_\mu.choose()$, $\mathfrak{h}_{<t}$
end function

Algorithm 2: LEARNING METHODS

For a $\mathfrak{v}$-task α, $\alpha.expand() := \{\omega \in \Gamma_{\mathfrak{v}} : \omega \sqsubset \alpha, \neg\exists\zeta \in \Gamma_{\mathfrak{v}}(\omega \sqsubset \zeta \sqsubset \alpha)\}$ gets the set of all α's immediate children (i.e. children which are only one generation away from α, meaning we exclude grandchildren etc)

$\mathfrak{q}$ is a prioritised queue. If (n, obj) is in $\mathfrak{q}$, then n is inverse priority (smaller n is higher priority). $\mathfrak{q}.put(n, obj)$ places (n, obj) in the queue, and $\mathfrak{q}.get()$ returns the highest priority item. $\mathfrak{q}.notempty()$ returns true iff $\mathfrak{q}$ is not empty.

If $\mathcal{V}$ is a set then $\mathcal{V}.add(\mathfrak{c}) := \mathcal{V} \cup \{\mathfrak{c}\}$.

function CLASSIFICATION($\mathfrak{h}_{<t}$, r) ▷ used by LEARNING()

  $O_{\mathfrak{h}^1_{<t}} \leftarrow \{o \in O_{\mathfrak{h}_{<t}} : r(o) = 1\}$ ▷ relatively "attractive" memories

  $O_{\mathfrak{h}^0_{<t}} \leftarrow O_{\mathfrak{h}_{<t}} - O_{\mathfrak{h}^1_{<t}}$

  $I_{\mathfrak{h}^1_{<t}} \leftarrow \{i \in I_{\mathfrak{h}_{<t}} : \exists o \in O_{\mathfrak{h}^1_{<t}} (i \subseteq o)\}$

  $I_{\mathfrak{h}^0_{<t}} \leftarrow \{i \in I_{\mathfrak{h}_{<t}} : \exists o \in O_{\mathfrak{h}^0_{<t}} (i \subseteq o)\}$

  $\mathfrak{h}^0_{<t} \leftarrow \langle I_{\mathfrak{h}^0_{<t}}, O_{\mathfrak{h}^0_{<t}} \rangle$

  $\mathfrak{h}^1_{<t} \leftarrow \langle I_{\mathfrak{h}^1_{<t}}, O_{\mathfrak{h}^1_{<t}} \rangle$

  return $\mathfrak{h}^1_{<t}, \mathfrak{h}^0_{<t}$

end function

Example 1 : A simple example showing how learning works without noise.

function LEARNING($\mathfrak{h}_{<t}$, $<^{\mathfrak{pp}}$, i_t, r)

  $\mathfrak{h}^1_{<t}, \mathfrak{h}^0_{<t} = \text{CLASSIFICATION}(\mathfrak{h}_{<t}, r)$

  $N_{\mathfrak{h}^1_{<t}} \leftarrow \{j \in \Pi_{\mathfrak{h}^1_{<t}} : E_j \cap E_{i_t} \neq \emptyset\}$ ▷ discard policies that don't apply

  $\Pi_{\mathfrak{h}^{(1,0)}_{<t}} \leftarrow \{(j,k) \in N_{\mathfrak{h}^1_{<t}} \times \Pi_{\mathfrak{h}^0_{<t}} : E_j \cap E_k = \emptyset\}$ ▷ remove ambiguity

  return π s.t. $\exists(\pi, n) \in \Pi_{\mathfrak{h}^{(1,0)}_{<t}}$ that maximises $<^{\mathfrak{pp}}$

end function

Algorithm 3: LEARNING METHODS CONTINUED...

Example 2 : Learning is possible even with noise (the only drawback being that it is harder to understand) by selectively "forgetting" outputs when no valid policy. Given only noise, there would be a policy for each output (rote learning means $\mid E_\pi \mid \rightarrow 1$).

function LEARNING($\mathfrak{h}_{<t}$, $<^{\mathfrak{pp}}$, i_t, r)
    $\mathfrak{q} \leftarrow \mathfrak{q}.put(0, \mathfrak{h}_{<t})$
    $\mathcal{V} \leftarrow \{\mathfrak{h}_{<t}\}$ ▷ set of visited nodes
    while $\mathfrak{q}.notempty()$ **do**
        $n, \mathfrak{h} \leftarrow \mathfrak{q}.get()$
        $\mathfrak{h}^1_{<t}, \mathfrak{h}^0_{<t} =$ CLASSIFICATION$(\mathfrak{h}, r)$
        $N_{\mathfrak{h}^1_{<t}} \leftarrow \{j \in \Pi_{\mathfrak{h}^1_{<t}} : E_j \cap E_{i_t} \neq \emptyset\}$
        $\Pi_{\mathfrak{h}^{(1,0)}_{<t}} \leftarrow \{(j,k) \in N_{\mathfrak{h}^1_{<t}} \times \Pi_{\mathfrak{h}^0_{<t}} : E_j \cap E_k = \emptyset\}$
        if $\Pi_{\mathfrak{h}^{(1,0)}_{<t}} \neq \emptyset$ **then**
            return π s.t. $\exists(\pi, n) \in \Pi_{\mathfrak{h}^{(1,0)}_{<t}}$ that maximises $<^{\mathfrak{pp}}$
        end if
        for all $\mathfrak{c} \in \mathfrak{h}.expand()$ **do**
            if $\mathfrak{c} \notin \mathcal{V}$ **then**
                $\mathfrak{q}.put(n+1, \mathfrak{c})$
                $\mathcal{V} \leftarrow \mathcal{V}.add(\mathfrak{c})$
            end if
        end for
    end while
    return $\emptyset$ ▷ try uninformed exploration when there is no valid policy
end function

Definition 26 (self-organising EGRL system)
A self-organising EGRL system $\mathfrak{o}$ *is* $\langle \mathfrak{v}_{\mathfrak{o}}, \mu_{\mathfrak{o}}, \mathfrak{p}_{\mathfrak{o}}, <_{\mathfrak{o}} \rangle$ *accompanied by an E.G.R.L system* $\langle \mathfrak{v}_{\mathfrak{o}}, \mathfrak{h}_{<t_{\mathfrak{o}}}, <^{\mathfrak{pp}}_{\mathfrak{o}}, r_{\mathfrak{o}} \rangle$ *and an E.G.R.L problem* $\langle \mathfrak{v}_{\mathfrak{o}}, \mu_{\mathfrak{o}}, len \rangle$*:*

- $O_{\mu_{\mathfrak{o}}}$ *contains every output which qualifies as "fit" according to natural selection.*
- $\mathfrak{p}_{\mathfrak{o}}$ *is the set of policies the self-organising EGRL system knows, s.t.* $\mathfrak{p}_{\mathfrak{o}} \subset \mathfrak{p}_{n.s.} \cup \mathfrak{p}_{\mathfrak{h}_{<t_{\mathfrak{o}}}}$ *and:*
  - $\mathfrak{p}_{n.s.} \subset L_{\mathfrak{v}_{\mathfrak{o}}}$ *is* ***reflexes*** *hard coded from birth by natural selection.*
  - $\mathfrak{p}_{\mathfrak{h}_{<t_{\mathfrak{o}}}} = \bigcup_{\zeta \in \mathfrak{h}_{<t_{\mathfrak{o}}}} \Pi_{\zeta}$ *is the set of policies it is possible to* ***learn****.*
  - *If* $(\mathfrak{p}_{\mathfrak{o}} - \mathfrak{p}_{n.s.}) \not\subset \Pi_{\mathfrak{h}_{<t_{\mathfrak{o}}}}$ *then the self-organising EGRL system has* ***selective memory***[28]*.*
- $<_{\mathfrak{o}}$ *is a binary relation over* $\Gamma_{\mathfrak{v}_{\mathfrak{o}}}$ *we call* ***preferences****.*
- $\mathfrak{o}$ *is an EGRL system iff maximising* $<_{\mathfrak{o}}$ *maximises* r *and* $\mathfrak{p}_{\mathfrak{o}} = \mathfrak{p}_{\mathfrak{h}_{<t_{\mathfrak{o}}}}$*.*

[28] It can "forget" outputs that contradict otherwise good policies.

EXAMPLES

THE FOLLOWING are based on this ref[29]. As before these are meant to be illustrative, rather than final definitions:

[29] Michael Timothy Bennett. Computational dualism and objective superintelligence. In *Artificial General Intelligence*. Springer Nature, 2024a

Definition 27 (bagī [バギー])
*"**B**ennett's **a**rtificial **g**eneral **i**ntelligence" (bagī) is an EGRL system $\langle \mathfrak{v}, \mathfrak{h}_{<t}, <^{\mathfrak{pp}}_{w}, r\rangle$ that uses weakness as the pair proxy[30]. From this ref[31] we have that bagī is the optimal system, meaning it is "most intelligent" given embodiment, in that it maximises the ability to complete a wide range of tasks using the vocabulary and history.*

[30] This assumes a particular vocabulary, which acts as a vehicle for cognition (bagī or バギー means "buggy" in Japanese).

[31] Michael Timothy Bennett. The optimal choice of hypothesis is the weakest, not the shortest. In *Artificial General Intelligence*. Springer Nature, 2023a

Definition 28 (unagi [鰻])
*The "**un**formed **a**rtificial **g**eneral **i**ntelligence" (unagi) formalises the upper bound given in this ref[32]. It is a system that searches through the space of finite vocabularies $V = \{v \subset P : v \text{ is } finite\}$ to identify the bagī for which the utility ϵ is maximised[33]. We assume an "**u**nformed GRL problem" (UGRL), which is $\langle \lambda_\mu, len\rangle$ where λ_μ is an uninstantiated task to be completed and len is the time horizon (it is like an EGRL problem $\langle \mathfrak{v}, \mu, len\rangle$, but λ_μ replaces μ and $\mathfrak{v}$). We also assume a "U.G.R.L system" given by $\langle \lambda_{\mathfrak{h}_{<t}}, <^{\mathfrak{pp}}_{w}, r\rangle$ where $<^{\mathfrak{pp}}_{w}$ is the **weakness** pair proxy and $\lambda_{\mathfrak{h}_{<t}}$ is the system's history ($\lambda_{\mathfrak{h}_{<t}}$ is an uninstantiated task replacing $\mathfrak{v}$ and $\mathfrak{h}_{<t}$ from the EGRL system definition)[34]. The unagi is then a search of vocabularies and then policies using those vocabularies. It takes $\langle \lambda_{\mathfrak{h}_{<t}}, <^{\mathfrak{pp}}_{w}, r\rangle$ and returns a policy π and vocabulary $\mathfrak{v}$ such that:*

$$\exists(\pi, \mathfrak{v}) \in V \times V \text{ s.t. } \pi \subset \mathfrak{v},\ \epsilon(\lambda_{\mathfrak{h}_{<t}}(\mathfrak{v})) \text{ and } <^{\mathfrak{pp}}_{w} \text{ are maximised}$$

Because $<^{\mathfrak{pp}}$ is the weakness proxy, an unagi finds the bagī is best suited to an uninstantiated task, and then constructs that bagī (it identifies the best tool for the problem, and then builds it).

[32] Michael Timothy Bennett. Computational dualism and objective superintelligence. In *Artificial General Intelligence*. Springer Nature, 2024a

[33] Its namesake is the Japanese water eel, slippery (having no particular embodiment). I used to get unagi lunches for free as an undergrad student, and this theoretical unagi system requires a "free lunch" in the sense of the "no free lunch" theorem.

[34] To have a disembodied history we must embrace dualism, which means unagi is impossible to build within the confines of the environment.

LIST OF PROOFS

Versions of the following three proofs were published in this ref[35].

Theorem 1 (sufficiency)
Assume $\alpha \sqsubset \omega$. The weakness proxy sufficient to maximise the probability that a parent ω is learned from a child α[36].

Proof 1 *You're given the definition of $\mathfrak{v}$-task α from which you infer a hypothesis $\pi \in \Pi_\alpha$. To learn ω, you need $\pi \in \Pi_\omega$:*

1. *For every $\pi \in \Pi_\alpha$ there exists a $\mathfrak{v}$-task $\gamma_\pi \in \Gamma_\mathfrak{v}$ s.t. $O_{\gamma_\pi} = E_\pi$, meaning π permits only correct outputs for that task regardless of input. We'll call the highest level task γ_π s.t. $O_{\gamma_\pi} = E_\pi$ the **policy task** of π.*
2. *ω is either the policy task of a policy in Π_α, or a child thereof*[37].
3. *If a policy π is correct for a parent of ω, then it is also correct for ω. Hence we should choose π that has a policy task with the largest number of children. As tasks are uniformly distributed, that will maximise the probability that ω is γ_π or a child thereof.*
4. *For the purpose of this proof, we say one task is **equivalent**[38] to another if it has the same correct outputs.*
5. *No two policies in Π_α have the same policy task*[39]. *This is because all the policies in Π_α are derived from the same set inputs, I_α.*
6. *The set of statements which might be outputs addressing inputs in I_ω and not I_α, is $\overline{E_{I_\alpha}} = \{l \in L_\mathfrak{v} : l \notin E_{I_\alpha}\}$*[40].
7. *For any given $\pi \in \Pi_\alpha$, the extension E_π of π is the set of outputs π implies. The subset of E_π which fall outside the scope of what is required for the known task α is $\overline{E_{I_\alpha}} \cap E_\pi$*[41].
8. *$L_\mathfrak{v} = \overline{E_{I_\alpha}} \cup E_{I_\alpha}$ and for all $\pi \in \Pi_\alpha$, $E_\pi \subset L_\mathfrak{v}$. Apart from the inputs and correct outputs of α, E_{I_α} contains only outputs which would be incorrect according to both α and ω. Put another way, $E_{I_\alpha} \cap E_\pi = O_\alpha$ for every possible choice of π in Π_α. Hence the only way $|E_\pi|$ can increase is if $\left|\overline{E_{I_\alpha}} \cap E_\pi\right|$ increases. It follows that $\left|\overline{E_{I_\alpha}} \cap E_\pi\right|$ increases with $|E_\pi|$.*
9. *$2^{\left|\overline{E_{I_\alpha}} \cap E_\pi\right|}$ is the number of non-equivalent **parents** of α to which π generalises. It increases monotonically with the weakness of π.*
10. *Given $\mathfrak{v}$-tasks are uniformly distributed and $\Pi_\alpha \cap \Pi_\omega \neq \emptyset$, the probability that $\pi \in \Pi_\alpha$ generalises to ω is*

$$p(\pi \in \Pi_\omega \mid \pi \in \Pi_\alpha, \alpha \sqsubset \omega) = \frac{2^{\left|\overline{E_{I_\alpha}} \cap E_\pi\right|}}{2^{\left|\overline{E_{I_\alpha}}\right|}}$$

[35] Michael Timothy Bennett. The optimal choice of hypothesis is the weakest, not the shortest. In *Artificial General Intelligence*. Springer Nature, 2023a

[36] Assume there exist correct policies for ω, because otherwise there would be no point in trying to learn it.

[37] I'd like to give credit here to Nora Belrose for pointing out an error. Nora pointed out I was miscounting the number of tasks. As a result I realised I was not counting tasks, I was in fact counting policy tasks and had entirely neglected to mention this fact. This was a significant error which has now been corrected, with several additional steps added to account for equivalence.

[38] This is because switching from β to ζ s.t. $I_\beta \neq I_\zeta$ and $O_\beta = O_\zeta$ would be to pursue the same goal in different circumstances. This is because inputs are *subsets* of outputs, so both sets of inputs are implied by the outputs. O_ζ implies I_β and O_β implies I_ζ

[39] Every policy task for policies of α is non-equivalent from the others.

[40] This is because E_{I_α} contains every statement which is a correct output or an incorrect output, and $\overline{E_{I_\alpha}}$ contains every statement which could possibly be in I_ω, E_{I_ω} and thus O_ω.

[41] This is because E_{I_α} is the set of all conceivable outputs by which one might attempt to complete α, and so the set of all outputs that can't be made when undertaking α is $\overline{E_{I_\alpha}}$ because those outputs occur given inputs that aren't part of I_α.

*$p(\pi \in \Pi_\omega \mid \pi \in \Pi_\alpha, \alpha \sqsubset \omega)$ is maximised when $|E_\pi|$ is maximised. Recall from definition 4 that $<_w$ is the **weakness** proxy. For statements l_1, l_2 we have $l_1 <_w l_2$ iff $|E_{l_1}| < |E_{l_2}|$. π that maximises $<_w$ will also maximise $p(\pi \in \Pi_\omega \mid \pi \in \Pi_\alpha, \alpha \sqsubset \omega)$. Hence the weakness proxy maximises the probability that[42] a parent ω is learned from a child α.* □

[42] Subsequently it also maximises the sample efficiency with which a parent ω is learned from a child α.

Theorem 2 (necessity)
To maximise the probability of learning ω from α, it is necessary to use weakness as a proxy.

Proof 2 *Let α and ω be defined exactly as they were in proof 1.*

1. *If $\pi \in \Pi_\alpha$ and $E_{I_\omega} \cap E_\pi = O_\omega$, then it must be he case that $O_\omega \subseteq E_\pi$.*
2. *If $|E_\pi| < |O_\omega|$ then generalisation cannot occur, because that would mean that $O_\omega \not\subseteq E_\pi$.*
3. *Therefore generalisation is only possible if $|E_\pi| \geq |O_\omega|$, meaning a sufficiently weak hypothesis is necessary to generalise from child to parent.*
4. *For any two hypotheses π_1 and π_2, if $\mid E_{\pi_1} \mid < \mid E_{\pi_2} \mid$ then the probability $p(|E_{\pi_1}| \geq |O_\omega|) < p(|E_{\pi_2}| \geq |O_\omega|)$ because tasks are uniformly distributed.*
5. *Hence the probability that $|E_m| \geq |O_\omega|$ is maximised when $|E_m|$ is maximised. To maximise the probability of learning ω from α, it is necessary to select the weakest hypothesis.*

To select the weakest hypothesis, it is necessary to use the weakness proxy. □

Theorem 3 (simplicity sub-optimality)
Description length is neither a necessary nor sufficient proxy for the purposes of maximising the probability that induction generalises.

Proof 3 *In proofs 1 and 2 we proved that weakness is a necessary and sufficient choice of proxy to maximise the probability of generalisation. It follows that either maximising $\frac{1}{|m|}$ (minimising description length) maximises $|E_m|$ (weakness), or minimisation of description length is unnecessary to maximise the probability of generalisation. Assume the former, and we'll construct a counterexample with $\mathfrak{v} = \{a, b, c, d, e, f, g, h, j, k, z\}$ s.t. $L_\mathfrak{v} = \{\{a, b, c, d, j, k, z\}, \{e, b, c, d, k\}, \{a, f, c, d, j\}, \{e, b, g, d, j, k, z\}, \{a, f, c, h, j, k\}, \{e, f, g, h, j, k\}\}$ and a task α where*

- $I_\alpha = \{\{a, b\}, \{e, b\}\}$
- $O_\alpha = \{\{a, b, c, d, j, k, z\}, \{e, b, g, d, j, k, z\}\}$
- $\Pi_\alpha = \{\{z\}, \{j, k\}\}$

Weakness as a proxy selects $\{j, k\}$*, while description length as a proxy selects* $\{z\}$*. This demonstrates the minimising description length does not necessarily maximise weakness, and maximising weakness does not minimise description length. As weakness is necessary and sufficient to maximise the probability of generalisation, it follows that minimising description length is neither.* $\square$

THE FOLLOWING PROOF proof was published in this ref[43].

[43] Michael Timothy Bennett. Computational dualism and objective superintelligence. In *Artificial General Intelligence*. Springer Nature, 2024a

Theorem 4 (upper bound) *The most 'intelligent' choice of policy and vocabulary given uninstantiated task λ_ρ is π and $\mathfrak{v}$ s.t. $\mathfrak{v}$ maximises utility for $\lambda_\rho(\mathfrak{v})$, $\pi \in \Pi_{\lambda_\rho(\mathfrak{v})}$ and π maximises weakness.*

Proof 4 *We have equated intelligence with sample efficient generalisation. The weakest correct policies have the highest probability of generalising. Given an uninstantiated task λ_ρ, utility measures the weakness of the weakest correct policies. We can use this to compare vocabularies. By choosing a vocabulary $\mathfrak{v}$ which maximises utility for $\lambda_\rho(\mathfrak{v})$, we instantiate λ_ρ in a vocabulary that maximises the weakness of correct policies for λ_ρ even in the absence of abstraction (meaning when $\mathfrak{v} = P$). Then, using weakness proxy, we can select a policy that has the highest possible probability of generalising, and thus maximise sample efficiency.* $\square$

THE FOLLOWING TWO proofs were published in this ref[44].

[44] Michael Timothy Bennett. Is complexity an illusion? In *Artificial General Intelligence*. Springer Nature, 2024b

Theorem 5 (subjectivity) *If there is no abstraction, complexity can always be minimized without improving sample efficiency, regardless of the task.*

Proof 5 *In accord with the definition of an abstraction layer, the absence of abstraction means the vocabulary is the set of all declarative programs, meaning $\mathfrak{v} = P$. It follows that for every $l \in L_{\mathfrak{v}}$ there exists $f \in \mathfrak{v}$ such that $\bigcap l = f$. Statements l and $\{f\}$ are equivalent iff $E_l = E_{\{f\}}$, which is exactly the case here because $\bigcap l = f$. Theorems 1 and 2 show that maximising weakness is necessary and sufficient to maximise the probability of generalisation, which means weakness maximises sample efficiency (is the optimal proxy). This means sample efficiency is determined by the cardinality of extension. For every correct policy l of every task in $\Gamma_{\mathfrak{v}}$ there exists $f \in \mathfrak{v}$ s.t. $E_l = E_{\{f\}}$. Policy complexity can be minimised regardless of weakness, because the simplest representation of every extension is a set containing exactly one program.* □

Theorem 6 (confounding) *If the vocabulary is finite, then policy weakness can confound[45]sample efficiency with policy simplicity.*

[45] *A confounds B* and *C* when for example $A =$ "*badlyinjured*" *causes* $B =$ "*died*" and $C =$ "*pickedupbyambulance*", and it looks like C causes B because $p(B \mid C) > p(B \mid \neg C)$, and yet it may be that $p(B \mid C, A) < p(B \mid \neg C, A)$.

Proof 6 *We already have that policy weakness causes sample efficiency, in that it is necessary and sufficient to maximise it in order to maximise sample efficiency. Continuing from proof 1, in a finite vocabulary, there may not exist $f \in \mathfrak{v}$ s.t. $E_l = E_{\{f\}}$, which means the complexity of all extensions will not be the same. If we choose any vocabulary in which weaker aspects take simpler forms, then simplicity will be correlated with weakness and so will also be correlated with sample efficiency. This means we would choose $\mathfrak{v}$ s.t. for all $a, b \in L_{\mathfrak{v}}$, the simpler statement has the larger extension, meaning $a <_w b \leftrightarrow a <_d b$. For example, suppose $P = \{a, b, c \dots\}$, $a = \{1, 2, 4\}$, $b = \{1, 3, 4\}$, $\mathfrak{v} = \{a, b\}$, $L_{\mathfrak{v}} = \{\{a\}, \{b\}, \{a, b\}\}$, then it follows $\{a, b\} <_w \{a\}$, $\{a, b\} <_w \{b\}$, $\{a, b\} <_d \{a\}$, $\{a, b\} <_d \{b\}$.* □

THE FOLLOWING proofs were published in these refs[46,47].

Theorem 7 (The Law of the Stack) *The greater the utility $\epsilon(\lambda^{i+1}(\mathfrak{v}^{i+1}))$, the weaker the policy π^i s.t. $\mathfrak{f}(E_{\pi^i}) = \mathfrak{v}^{i+1}$ must be*[48].

Proof 7 *If $\mathfrak{a} \subset \mathfrak{b}$ then $\epsilon(\lambda^{i+1}(\mathfrak{a})) < \epsilon(\lambda^{i+1}(\mathfrak{b}))$, meaning if $\mathfrak{b}$ is the vocabulary at $i+1$, then it will be possible to construct weaker policies than if $\mathfrak{a}$ is the vocabulary (intuitively, a larger vocabulary enables a wider range of policies). We consider two policies $\pi^i_\mathfrak{a}$ and $\pi^i_\mathfrak{b}$ which could be the policy π^i at i. If $\mathfrak{a} = \mathfrak{f}(E_{\pi^i_\mathfrak{a}}) \subset \mathfrak{f}(E_{\pi^i_\mathfrak{b}}) = \mathfrak{b}$, then $\pi^i_\mathfrak{a} <_w \pi^i_\mathfrak{b}$, meaning an enlarged vocabulary at $i+1$ implies a weaker policy at i.* □

[46] Michael Timothy Bennett. Are biological systems more intelligent than artificial intelligence? Forthcoming in *Philosophical Transactions of The Royal Society B, Special Issue on Hybrid Agencies*, 2025a; and

[47] Further proofs and discussion are available in a reply to one of the papers, written by Gabriel Simmons. It gives an intuitive and succinct description of the formalism, and an example of a task that doesn't have a correct policy. The comment pertains to one paper in particular and so is missing some of the broader context, but it raises some interesting questions that I answer in other publications, or seek to answer in this thesis.

Gabriel Simmons. Comment on is complexity an illusion?, 2024. URL `https://arxiv.org/abs/2411.08897`

[48] Intuitively, this just means adaptability at higher levels implies adaptability at lower levels.

Theorem 8 ($\mathbf{n^{th}}$ order self convergence) *An organism that uses weakness as its proxy will learn an n^{th} order self if the incentive and representation preconditions are met for that order of self.*

Proof 8 *Assume we have an organism $\mathfrak{o}$ that learns using "weakness" as a proxy. A $\mathfrak{v}_\mathfrak{o}$-task $\mathfrak{h}_{<t_\mathfrak{o}}$ represents the history of $\mathfrak{o}$ (meaning $\mathfrak{h}_{<t_\mathfrak{o}} \sqsubset \mu_\mathfrak{o}$ and $\mathfrak{h}_{<t_\mathfrak{o}}$ is an ostensive definition of $\mu_\mathfrak{o}$, by virtue of the fact that $\mathfrak{o}$ remains alive). The organism explores the environment, intervening to maintain homeostasis. As it does so, more and more inputs and outputs are included in $\mathfrak{h}_{<t_\mathfrak{o}}$. It follows that:*

1. *From the representation precondition we have that there exists a n^{th} order self $\mathfrak{o}^\mathfrak{n} \in L_{\mathfrak{v}_\mathfrak{o}}$.*
2. *To remain fit, $\mathfrak{o}$ must "generalise" to $\mu_\mathfrak{o}$ from $\mathfrak{h}_{<t_\mathfrak{o}}$. According to the incentive precondition, generalisation to $\mu_\mathfrak{o}$ requires $\mathfrak{o}$ learn the n^{th} order self, which is when $\mathfrak{o}^\mathfrak{n} \in \mathfrak{p}_\mathfrak{o}$.*
3. *From this ref*[49] *we have proof that weakness is the optimal choice of proxy to maximise the probability of generalisation from child to parent is the weakest policy. It follows that $\mathfrak{o}$ will generalise from $\mathfrak{h}_{<t_\mathfrak{o}}$ to $\mu_\mathfrak{o}$ given the smallest history of interventions with which it is possible to do so (meaning the smallest possible ostensive definition, or cardinality $|O_\alpha|$).*

[49] Michael Timothy Bennett. The optimal choice of hypothesis is the weakest, not the shortest. In *Artificial General Intelligence*. Springer Nature, 2023a

Were we to assume learning under the above conditions does not construct an n^{th} order self for $\mathfrak{o}$, then one of the three statements above would be false and we would have a contradiction. It follows that the proposition must be true. □

THE FOLLOWING are new proofs included in the thesis. The first is to be published in *Lies, Damned Lies, and the Orthogonality Thesis*[50].

[50] Michael Timothy Bennett. Lies, damned lies, and the orthogonality thesis. *Under Review*, 2025b

Theorem 9 *Intelligence is not independent of goals.*

Proof 9 *Assume $\mathcal{C}$ is a space of software programs, Γ is a space of behaviours a system can exhibit, $f_1 \in \mathcal{C}$ is a software mind and $f_2 : \mathcal{C} \to \Gamma$ is a hardware body. It interprets f_1. Finally $\mathbb{G}$ is the set of environments, which here are functions $f : \Gamma \to \{0, 1\}$. We single out $f_3 \in \mathbb{G}$ as the environment in which goals are pursued. If I am the engineer who build the software AI f_1 for some particular purpose, then I am part of its environment. Goals are satisfied if $f_3(f_2(f_1)) = 1$. I will now show why intelligence is not independent of embodiment, and embodiment is not independent of goals.*

1. *Suppose we are given f_3. For every environment there exists hardware s.t. $f_3(f_2(\cdot)) = 1$ regardless of software (it maps all software to the same behaviour, or in other words it is hard-wired to satisfy the goals in f_3).*
2. *For every choice of f_1 given f_3 alone, there exists f_2 s.t. $f_3(f_2(f_1)) = 0$.*
3. *This means intelligence is not independent of embodiment*[51]*.*
4. *Now I'll show goals are not independent of embodiment. Assume we are given a fixed $f_2(f_1)$.*
5. *We can choose f_3 so that $f_2(f_1)$ is optimal and the goals in f_3 are satisfied.*
6. *However, $f_2(f_1)$ determines which choices of f_3 are optimal. $f_2(f_1) = \gamma$ constrains us to choices of f_3 where $f_3(\gamma) = 1$*[52]*.*
7. *This means goals are not independent of embodiment.*

As intelligence hinges on embodiment, and embodiment is goal directed, intelligence is inevitably goal directed. $\square$

[51] Similar points are made in related work
Michael Timothy Bennett. The optimal choice of hypothesis is the weakest, not the shortest. In *Artificial General Intelligence*. Springer Nature, 2023a; Michael Timothy Bennett. Is complexity an illusion? In *Artificial General Intelligence*. Springer Nature, 2024b; and Jan Leike and Marcus Hutter. Bad universal priors and notions of optimality. *Proceedings of The 28th Conference on Learning Theory, in Proceedings of Machine Learning Research*, pages 1244–1259, 2015

[52] Meaning only those environments and goals where a particular behaviour or phenotype γ will succeed.

I TRANSLATE THE CONCEPTS of Wong et. al.[53] into Pancomputational Enactivism as follows:

[53] Michael L. Wong, Carol E. Cleland, Daniel Arend, Stuart Bartlett, H. James Cleaves, Heather Demarest, Anirudh Prabhu, Jonathan I. Lunine, and Robert M. Hazen. On the roles of function and selection in evolving systems. *Proceedings of the National Academy of Sciences*, 120(43):e2310223120, 2023. DOI: 10.1073/pnas.2310223120. URL https://www.pnas.org/doi/abs/10.1073/pnas.2310223120

1. Their function x sounds like it could be a $\mathfrak{v}$-task α in vocabulary $\mathfrak{v}$. The possible futures given past α would be a parent ω of α.

2. They use F_x to denote a quantitative measure of a configurations ability to perform function x. I would say a "configuration" here is a policy π, and F_x sounds a bit like the cardinality of the extension $|E_\pi|$ of a policy π. The weaker π is, the greater its ability to perform function x, which is the probability that it generalises to the parent task ω of α.

3. They use $M(F_x)$ to be the number of configurations with a degree of function *greater* than F_x. This sounds like the number of correct policies which are *weaker* than π. I will interpret this as meaning the cardinality the set $|M_\pi|$ such that $M_\pi = \{\pi_{alt} \in \Pi_\alpha : |E_\pi| \geq |E_{\pi_{alt}}|$, meaning M_π has at least 1 member.

4. N would be the total number of possible configurations, for example if we are dealing with an n bit string then $N = 2^n$. I will interpret this as being the total number of possible statements which can be made in the abstraction layer $L_\mathfrak{v}$, so $|L_\mathfrak{v}| = N$.

5. Functional information $I(F_x) = -\log_2(\frac{M(F_x)}{N})$ would then be a function of weakness $I(\pi) = -\log_2(\frac{|M_\pi|}{|L_\mathfrak{v}|})$, and functional information is attained by w-maxing π.

IF ONE ACCEPTS MY INTERPRETATION of functional information as being a function of weakness, then I can prove the law of increasing functional information. It follows trivially from the fact that w-maxing is necessary and sufficient to maximise the probability of generalisation.

Theorem 10 (The Law of Increasing Functional Information) *The functional information in a system will increase with time, where definitions are as above. Assume systems are policies. Assume ω represents future selection pressures on policies, and policies which generalise to ω from α are those which persist into the future.*

Proof 10 *From proofs 1 and 2 we have that w-maxing is sufficient to maximise the probability of generalising from α to ω. Therefore the policies which generalise will be the weakest. W-maxing maximises functional information, so functional information must increase with time.* $\square$

THE FOLLOWING are supplementary proofs that exist only here.

Theorem 11 (pair proxy optimality)
The optimal choice of pair proxy for EGRL is the weakness pair proxy.

Proof 11 *For every pair (a, b) of statements s.t. $a \cap b = \emptyset$ there exists a statement c such that $E_c = E_a \cup E_b$. The completions of c are completions of either a or b. An extension is a truth table, so this means that a pair (a, b) is equivalent to a single policy c. From proofs 1 and 2 we already have that the optimal policy for a single policy is weakness, and we can the weakness of c in accord with $<_w$ and definition 9 is exactly the weakness of (a, b) in accord with $<_w^{\mathtt{pp}}$ and definition 25. Because a pair of policies is equivalent to a policy the optimal pair proxy is equivalent to the optimal proxy, and so the optimal pair proxy is $<_w^{\mathtt{pp}}$.*

Theorem 12 (do operator equivalence) *We can substitute the do operator for a statement l.*

Proof 12 *In the context of the conditional probability of Y, for every intervention $do[X = x]$ there exists a variable A which is equivalent given X, meaning $p(Y \mid A = true, X = x) = p(Y \mid do[X = x])$ and $p(Y \mid A = false, X = x) = p(Y \mid X = x)$ (the equivalence between variables and the do operator was pointed out by Dawid[54], however we will prove it here for illustrative purposes). Likewise, for each possible value y of Y, there are three possibilities to consider pertaining to X and the do operator. Each of these cases has a probability of occurring represented by the rightmost column.*

[54] A. P. Dawid. Influence diagrams for causal modelling and inference. *International Statistical Review / Revue Internationale de Statistique*, 70(2):161–189, 2002. ISSN 03067734, 17515823. URL `http://www.jstor.org/stable/1403901`

$do[X = x]$	$X = x$	$p(Y \mid ...)$
true	*true*	$p(Y \mid do[X = x]) = p(Y)$
false	*true*	$p(Y \mid X = x)$
false	*false*	$p(Y \mid X \neq x)$

In all cases we either have $do[X = x]$, or we do not. Therefore we can substitute the application of the do operator effecting $X = x$ for $A \in \{true, false\}$. A is not a random variable, but something to which one assigns a true or $false$ value to indicate whether $X = x$ is the result of one's intervention. We then assert the following conditional probabilities:

$A = true$	$X = x$	$p(Y \mid ...)$
true	*true*	$p(Y \mid A = true, X = x) = p(Y)$
true	*false*	$p(Y \mid A = true, X \neq x) = 0$
false	*true*	$p(Y \mid A = false, X = x) = p(Y \mid X = x)$
false	*false*	$p(Y \mid A = false, X \neq x) = p(Y \mid X \neq x)$

As $p(Y \mid A = true, X = x) = p(Y \mid do[X = x])$ *and* $p(Y \mid A = false, X = x) = p(Y \mid X = x)$ *for all possible cases (*$p(Y \mid A = true, X \neq x)$ *being an impossibility). From the definition of an environment we have that* 2^{Φ} *contains every declarative program.* 2^{Φ} *is sufficient to represent* A*, because there exists* $a \subset 2^{\Phi}$ *s.t.* a *is true iff* $A = true$*, so* a *is equivalent to* $do[A]$. $\square$

EXPERIMENTS

In the appendix is a Python script to perform two experiments using PyTorch with CUDA, SymPy and A^*[55]. Here I'll summarise the results and how the experiments worked. First, I wrote a toy program based on A^* that learns policies for 8-bit string prediction tasks (binary addition and multiplication)[56].

(abstraction layer) I used a simplified environment of 256 states, one for every possible 8-bit string. Basically an abstraction layer without our environment. The statements in L were expressions regarding those 8 bits that could be written in propositional logic ($\neg$, $\wedge$ and $\vee$).

(task) A task was specified by choosing $O \subset L$ such that all $d \in O$ conformed to the rules of either binary addition (for the first experiment) or multiplication (for the second experiment) with 4-bits of input, followed by 4-bits of output.

Each of the two experiments (addition and multiplication) involved repeated trials (sampling results). The parameters of each trial were "operation" (a function), and an even integer "number_of_trials" between 4 and 14 which determined the cardinality of the set O_k (defined below). Each trial was divided into training and testing phases.

(training phase)

1. A task T_n was generated:
   (a) First, every possible 4-bit input for the chosen binary operation was used to generate an 8-bit string. These 16 strings then formed O_n.
   (b) A bit between 0 and 7 was then chosen, and I_n created by cloning O_n and deleting the chosen bit from every string (meaning I_n was composed of 16 different 7-bit strings, each of which could be found in an 8-bit string in O_n).
2. A child-task $T_k = \langle I_k, O_k \rangle$ was sampled from the parent task T_n. Recall, $|O_k|$ was determined as a parameter of the trial.
3. From T_k two policies (formerly known as models) were generated; a weakest c_w, and a MDL c_{mdl}.

(testing phase) For each policy $c \in \{c_w, c_{mdl}\}$:

1. The extension E_c of c was then generated.
2. A prediction O_{recon} was then constructed s.t. $O_{recon} = \{e \in E_c : \exists s \in I_n \ (s \subset z)\}$.

[55] Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019; David Kirk. Nvidia cuda software and gpu parallel computing architecture. In *Proceedings of the 6th International Symposium on Memory Management*, ISMM '07, page 103–104, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595938930. doi: 10.1145/1296907.1296909. URL `https://doi.org/10.1145/1296907.1296909`; Aaron Meurer, Christopher Smith, Mateusz Paprocki, Ondřej Čertík, Sergey Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian Granger, Richard Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, and Anthony Scopatz. Sympy: Symbolic computing in python. *PeerJ Computer Science*, 3:e103, 01 2017. doi: 10.7717/peerj-cs.103; and Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136

[56] Notation here varies slightly from the formal notation due to the limitations of what can be written in Python (not latex), and because it the experiments coincided with an earlier iteration of the formalism.

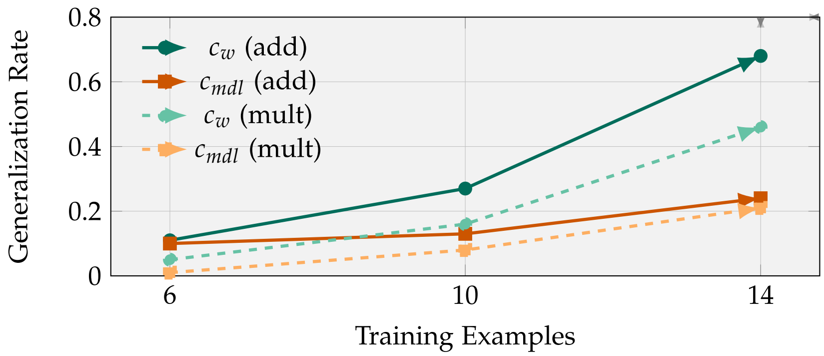

Figure 1: Rates for binary addition (solid) and mult. (dashed).

3. O_{recon} was then compared to the ground truth O_n, and results recorded.

Between 75 and 256 trials were run for each value of the parameter $|O_k|$. Fewer trials were run for larger values of $|O_k|$ due to restricted availability of hardware. The results of these trails were then averaged for each value of $|O_k|$.

(MEASUREMENTS) Generalisation was deemed to have occurred where $O_{recon} = O_n$. The number of trials in which generalisation occurred was measured, and divided by n to obtain the rate of generalisation for c_w and c_{mdl}. Error was computed as a Wald 95% confidence interval. Even where $O_{recon} \neq O_n$, the extent to which policies generalised could be ascertained. $\frac{|O_{recon} \cap O_n|}{|O_n|}$ was measured and averaged for each value of $|O_k|$, and the standard error computed.

	c_w				c_{mdl}			
$\|O_k\|$	Rate	±95%	AvgExt	StdErr	Rate	±95%	AvgExt	StdErr
6	.11	.039	.75	.008	.10	.037	.48	.012
10	.27	.064	.91	.006	.13	.048	.69	.009
14	.68	.106	.98	.005	.24	.097	.91	.006

Table 1: Binary addition.

	c_w				c_{mdl}			
$\|O_k\|$	Rate	±95%	AvgExt	StdErr	Rate	±95%	AvgExt	StdErr
6	.05	.026	.74	.009	.01	.011	.58	.011
10	.16	.045	.86	.006	.08	.034	.78	.008
14	.46	.061	.96	.003	.21	.050	.93	.003

Table 2: Binary multiplication.

EXAMPLES

We are only interested in computers that actually exist. Because memory is always finite, a computer that exists is a finite state machine (F.S.M.), not a Turing Machine. Instead of conventional models of computation we're using tasks, which are more general. To show how tasks relate to established notions of computation, we show how a F.S.M. can be converted into a task and back again. Let F be the set of all finite state machines, and from the definition of uninstantiated tasks we have that the set of all tasks in all vocabularies is Γ.

A F.S.M. is $\langle Q, \sigma, \delta, q_0, A \rangle$:

- Q is a set of F.S.M. states (we use "F.S.M. state" to distinguish Q from Φ).
- Σ is the input alphabet (a finite non-empty set of symbols);
- $\delta : Q \times \Sigma \rightarrow Q$ is the state transition function.
- $q_0 \in Q$ is the initial F.S.M. state.
- $A \subset Q$ is a set of correct or "accepting" F.S.M. states.

Recall that statements and the declarative programs they contain describe any and all aspects of a physicalist environment. This means we can represent the behaviour of a F.S.M. f as a $\mathfrak{v}$-task ρ. To do so we define:

- a sensorimotor vocabulary $\mathfrak{v}$ s.t. there is a unique declarative program in $\mathfrak{v}$ for each and every member of Q and Σ, every subset of Q and Σ, and every possible sequence of members of these sets or any part thereof.
- E_{I_ρ} as the set of all statements describing those sequences of input symbols and state transitions that include δ_ρ. Because Σ is finite, there are a finite number of infinite length input sequences, and so we can describe even these infinite length sequences using our finite vocabulary $\mathfrak{v}$.
- $O_\rho \subset E_{I_\rho}$ is the set of all statements describing sequences ending in an "accepting" state, and $I_\rho = E_{I_\rho} - O_\rho$.
  - Each member $\sigma \in \Sigma$ becomes a member of $\mathfrak{v}$. We call these **declarative equivalents**. We also include in $\mathfrak{v}$ a declarative program $\sigma_{a<b}$ for every pair $(\sigma_a, \sigma_b) \in \Sigma$ such that $\phi \in \sigma_{a<b}$ iff $\phi \in \sigma_a$ and $\phi \in \sigma_b$ and σ_b comes "after" σ_a. This indicates relative ordering. We call these **declarative orderings**.

- If $\sigma_a, \sigma_b, \sigma_{a<b} \in l \in L_\mathfrak{v}$, then if l is true then it is a fact that σ_a comes before σ_b.
- We also then include in $\mathfrak{v}$ declarative equivalents and orderings for the members of Q. This means we can construct a statement l that describes an ordered sequence of states and input symbols.
- To avoid infinite sequences we construct a set K of declarative programs, one for every possible **non repeating** (as in not self similar) sequence of symbols and states, and an additional declarative program to indicate self similarity. Because there are finitely many symbols and states, there are finitely many possible sequences which are not self similar. Doing so lets us represent every possible sequence, so we include these programs in $\mathfrak{v}$ (meaning $K \subset \mathfrak{v}$).
- We must also ensure there is a correct policy. An easy way to do this is to include a declarative program $\delta_\rho \in \mathfrak{v}$ s.t. $\delta_\rho \in l$ only when a sequence described by l is exactly what δ implies (it is the declarative equivalent of the imperative δ).
- We then construct a set $\Delta \subset L_\mathfrak{v}$ s.t. $\Delta = \{l \in L_\mathfrak{v} : \exists k \in L_\mathfrak{v}(l \subseteq k, \delta_\rho \in k)\}$, which means every sequence or part thereof which could be output by δ.
- $I_\rho \subset \Delta$ is then the set of all sequences that do **not** include the declarative equivalent of a member of A.
- $O_\rho \subset \Delta$ is then the set of all sequences that **do** include the declarative equivalent of a member of A.

f is then described in perfect detail, and every $f \in F$ is mapped to a unique task $\rho \in \Gamma$, so we have an injection $i : F \to \Gamma$. The process described above is reversible. We construct Q from E_{I_ρ}, Σ and q_0 from I_ρ, A from O_ρ, and then δ is just the imperative equivalent of δ_ρ.

Bibliography

Andrew B. Barron and Colin Klein. What insects can tell us about the origins of consciousness. *Proceedings of the National Academy of Sciences*, 2016.

Michael Timothy Bennett. The optimal choice of hypothesis is the weakest, not the shortest. In *Artificial General Intelligence*. Springer Nature, 2023a.

Michael Timothy Bennett. Emergent causality and the foundation of consciousness. In *Artificial General Intelligence*. Springer Nature, 2023b.

Michael Timothy Bennett. On the computation of meaning, language models and incomprehensible horrors. In *Artificial General Intelligence*. Springer Nature, 2023c.

Michael Timothy Bennett. Computational dualism and objective superintelligence. In *Artificial General Intelligence*. Springer Nature, 2024a.

Michael Timothy Bennett. Is complexity an illusion? In *Artificial General Intelligence*. Springer Nature, 2024b.

Michael Timothy Bennett. Are biological systems more intelligent than artificial intelligence? Forthcoming in *Philosophical Transactions of The Royal Society B, Special Issue on Hybrid Agencies*, 2025a.

Michael Timothy Bennett. Lies, damned lies, and the orthogonality thesis. *Under Review*, 2025b.

Piotr Boltuc. The engineering thesis in machine consciousness. *Techné: Research in Philosophy and Technology*, 2012.

Elliot Catt, Jordi Grau-Moya, Marcus Hutter, Matthew Aitchison, Tim Genewein, Gregoire Deletang, Li Kevin Wenliang, and Joel Veness. Self-predictive universal AI. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=psXVkKO9No`.

A. P. Dawid. Influence diagrams for causal modelling and inference. *International Statistical Review / Revue Internationale de Statistique*, 70(2):161–189, 2002. ISSN 03067734, 17515823. URL `http://www.jstor.org/stable/1403901`.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. DOI: 10.1109/TSSC.1968.300136.

David Kirk. Nvidia cuda software and gpu parallel computing architecture. In *Proceedings of the 6th International Symposium on Memory Management*, ISMM '07, page 103–104, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595938930. DOI: 10.1145/1296907.1296909. URL `https://doi.org/10.1145/1296907.1296909`.

Jan Leike and Marcus Hutter. Bad universal priors and notions of optimality. *Proceedings of The 28th Conference on Learning Theory, in Proceedings of Machine Learning Research*, pages 1244–1259, 2015.

Aaron Meurer, Christopher Smith, Mateusz Paprocki, Ondřej Čertík, Sergey Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian Granger, Richard Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, and Anthony Scopatz. Sympy: Symbolic computing in python. *PeerJ Computer Science*, 3: e103, 01 2017. DOI: 10.7717/peerj-cs.103.

Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.

Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, Inc., New York, 1st edition, 2018.

Gabriel Simmons. Comment on is complexity an illusion?, 2024. URL `https://arxiv.org/abs/2411.08897`.

Michael L. Wong, Carol E. Cleland, Daniel Arend, Stuart Bartlett, H. James Cleaves, Heather Demarest, Anirudh Prabhu, Jonathan I. Lunine, and Robert M. Hazen. On the roles of function and selection in evolving systems. *Proceedings of the National Academy of Sciences*, 120(43):e2310223120, 2023. DOI: 10.1073/pnas.2310223120. URL `https://www.pnas.org/doi/abs/10.1073/pnas.2310223120`.