Technical Appendices

Michael Timothy Bennett[1][0000−0001−6895−8782]

The Australian National University
michael.bennett@anu.edu.au
http://www.michaeltimothybennett.com/

Abstract. The following is a list of definitions [pp. 1-2], a frequently asked questions section containing examples of how these definitions may be applied [pp. 3-5], a description of two experiments performed using the code accompanying this appendix [pp. 5-7] and a list of proofs concerning the aforementioned definitions [p. 7].

1 List of definitions

Definitions 1, 2 and 3 are taken from [1]:

Definition 1 (states of reality). *A set H, where:*

- *We assume a set Φ whose elements we call **states**, one of which we single out as the **present state** of reality.*
- *A **declarative program** is a function $f : \Phi \to \{true, false\}$, and we write P for the set of all programs. By **objective truth** about a state ϕ, we mean a declarative program f such that $f(\phi) = true$.*
- *Given a state $\phi \in \Phi$, the **objective totality** of ϕ is the set of all objective truths $h_\phi = \{f \in P : f(\phi) = true\}$.*
- *$H = \{h_\phi : \phi \in \Phi\}$*

Definition 2 (implementable language). *A triple $\mathcal{L} = \langle H, V, L\rangle$, where:*

- *H is reality, the set containing all **objective totalities**.*
- *$V \subset \bigcup\limits_{h \in H} h$ is a finite set, named the **vocabulary**.*
- *$L = \{l \in 2^V : \exists h \in H \ (l \subseteq h)\}$, the elements of which are **statements**.*

(Extensions) The **extension of a statement** $a \in L$ is $Z_a = \{b \in L : a \subseteq b\}$, while the **extension of a set of statements** $A \subseteq L$ is $Z_A = \bigcup\limits_{a \in A} Z_a$.

(Notation) Lower case letters represent statements, and upper case represent sets of statements. The capital letter Z with a subscript indicates the extension of whatever is in the subscript. For example the extension of a statement a is Z_a, and the extension of a set of statements A is Z_A.

Definition 3 (task). *Given language $\langle H, V, L\rangle$, a task is $T = \langle S, D, M\rangle$ where:*

- *$S \subset L$ is a set of statements called **situations**, where the extension Z_S of S is the set of all **possible decisions** which can be made in those situations.*

– $D \subset Z_S$ *is the set of* ***correct decisions*** *for this task.*
– M *is the set of* ***models****, s.t.* $M = \{m \in L : Z_S \cap Z_m \equiv D, \forall z \in Z_m \ (z \subseteq \bigcup\limits_{d \in D} d)\}$

(How a task is completed) *Assume we have a hypothesis* $\boldsymbol{h} \in L$*:*

1. *we are then presented with a situation* $s \in S$*, and*
2. *we must select a decision* $z \in Z_s \cap Z_{\boldsymbol{h}}$*.*
3. *If* $z \in D$*, then the decision is correct and the task completed. This occurs if* $\boldsymbol{h} \in M$*.*

1.1 Induction definitions

Definitions 4, 5, 6, 7 and 8 are taken from [2]:

Definition 4 (probability of a task). *Let* Γ *be the set of all tasks given an implementable language* $\mathcal{L}$*. There exists a uniform distribution over* Γ*.*

Definition 5 (generalisation). *Given two tasks* $\alpha = \langle S_\alpha, D_\alpha, M_\alpha \rangle$ *and* $\omega = \langle S_\omega, D_\omega, M_\omega \rangle$*, a model* $m \in M_\alpha$ *generalises to task* ω *if* $m \in M_\omega$*.*

Definition 6 (child-task and parent-task). *A task* $\alpha = \langle S_\alpha, D_\alpha, M_\alpha \rangle$ *is a child-task of* $\omega = \langle S_\omega, D_\omega, M_\omega \rangle$ *if* $S_\alpha \subset S_\omega$ *and* $D_\alpha \subseteq D_\omega$*. This is written as* $\alpha \sqsubset \omega$*. If* $\alpha \sqsubset \omega$ *then* ω *is then a parent of* α*, and* α *is a child of* ω*.*

Definition 7 (proxy for intelligence). *A proxy is a function* $q : L \to \mathbb{N}$*. The set of all proxies is* Q*.*

(Weakness) *The weakness of a statement* m *is the cardinality of its extension* $|Z_m|$*. There exists* $q \in Q$ *such that* $q(m) = |Z_m|$*.*

(Description Length) *The description length of a statement* m *is its cardinality* $|m|$*. Longer logical formulae are considered less likely to generalise [3], and a proxy is something to be maximised, so description length as a proxy is* $q \in Q$ *such that* $q(m) = \frac{1}{|m|}$*.*

Definition 8 (induction). $\alpha = \langle S_\alpha, D_\alpha, M_\alpha \rangle$ *and* $\omega = \langle S_\omega, D_\omega, M_\omega \rangle$ *are tasks such that* $\alpha \sqsubset \omega$*. Assume we are given a proxy* $q \in Q$*, the complete definition of* α *and the knowledge that* $\alpha \sqsubset \omega$*. We are not given the definition of* ω*. The process of induction would proceed as follows:*

1. *Obtain a hypothesis by computing a model* $\mathbf{h} \in \underset{m \in M_\alpha}{\arg\max}\ q(m)$*.*
2. *If* $\mathbf{h} \in M_\omega$*, then we have generalised from* α *to* ω*.*

1.2 Causal definitions

Definitions 9 and 10 are taken from [4]:

Definition 9 (identity). *If* $a \in L$ *is an intervention [5] to force* $c \in L$*, then* $k \subseteq a - c$ *may function as an identity undertaking the intervention if* $k \neq \emptyset$*.*

Definition 10 (higher and lower level statements). *A statement* $c \in L$ *is higher level than* $a \in L$ *if* $Z_a \subset Z_c$*, which is written as* $a \sqsubset c$*.*

2 Frequently Asked Questions

2.1 How would you apply this to solve a typical regression problem?

Say we have a finite set of input values $X \subset \mathbb{R}$ and output values $Y \subset \mathbb{R}$ [1], and $g : X \to Y$ be a function we wish to model. $G = \{(x, y) \in X \times Y : g(x) = y\}$, and we call G the ground truth. Let $Train \subset G$ be a training set, and $Test \subset G$ be a test set. We are given $Train$ and $Test$, and our goal is to infer G. Typically, machine learning could be used to obtain an approximation of G from $Train$ by doing the following:

1. Fit a function f (e.g. a neural net) such that $\forall (x, y) \in Train\, (f(x) \approx y)$[2].
2. Measure test accuracy as $\frac{|\{(x,y) \in Test : y \approx f(x)\}|}{|Test|}$.
3. If accuracy good enough, use f to make predictions $P = \{(x, y) \in X \times Y : f(x) = y\}$ and hope that $\frac{|\{(x,y) \in P : y \approx g(x)\}|}{|P|} \approx 1$.

This is how we would solve the problem using machine learning normally. Now let us represent this as a task and solve it that way.

1. We start by creating an implementable language.
   (a) Begin by defining the vocabulary V of the implementable language $\mathcal{L} = \langle H, V, L\rangle$[3]. We need to create V first and then L, because we cannot actually create H [4].
   (b) To obtain L, we must first write a program *converter* $: X \cup Y \to 2^V$ which converts members of X and Y into sets of declarative programs, and another program $converter^{-1} : 2^V \to X \cup Y$ which reverses that process (meaning an isomorphism between $X \cup Y$ and 2^V).
   (c) Next we define $pair\_converter : X \times Y \to 2^V$ such that $\forall (x, y) \in X \times Y$, $pair\_converter((x, y)) = converter(x) \cup converter(y)$ and likewise the inverse $pair\_converter^{-1}((converter(x) \cup converter(y))) = (x, y)$.
   (d) A set Q can be created as

   $$Q = \{v \in 2^V : \exists (x, y) \in X \times Y\, (pair\_converter((x, y)) = v)\}$$

   We can then use Q to create L as each member of Q must be a subset of an objective totality $h \in H$ (even though we haven't needed to explicitly define H), meaning $L = \{l \in 2^V : \exists v \in Q\ (l \subseteq v)\}$.
2. Now we can use *converter* and *pair_converter* to define a task $\langle S, D, M\rangle$.
   (a) First we compute $S = \{s \in L : \exists (x, y) \in Train\, (converter(x) = s)\}$.

[1] We assume finite X and Y for practical reasons, for example that the real numbers we can represent as floating point values in a computer are constrained by the number of bits used.

[2] $a \approx b$ just means that there exists a very small number c and a measure of distance d such that $d(a, b) < c$, meaning the distance $d(a, b)$ between a and b is less than c

[3] The choice of V permit an isomorphism between $X \times Y$ and 2^V

[4] H is the set of objective totalities of states of the universe, which may be infinite and are certainly impractical to represent. Subsequently we must obtain L from directly V using the structure inherent in the values of X and Y (e.g. X and Y represent real numbers in different parts of memory in a computer, not the same part, and so we can create statements in L describing values from both without creating problems).

(b) Second we create the set of correct decisions

$$D = \{d \in L : \exists (x, y) \in Train\,(pair\_converter((x, y)) = d)\}$$

(c) Finally to create $M \subset L$ by excluding members of L according to the definition:

$$M = \{m \in L : Z_S \cap Z_m \equiv D, \forall z \in Z_m \ (z \subseteq \bigcup_{d \in D} d)\}$$

3. We now have a set of models M and situations S which we can treat as constraints, and can define a program $search : S, M \rightarrow D$ which, given any situation and model, returns a decision in D. We can now measure the accuracy of a given model $m \in M$.
   (a) First we compute $S_{Test} = \{s \in L : \exists (x, y) \in Test\,(converter(x) = s)\}$.
   (b) Compute $D_{Test} = \{l \in L : \exists s \in S_{Test}\,(search(s, m) = l)\}$.
   (c) Convert to real numbers by computing:

$$P = \{(x, y) \in X \times Y : \gamma(x, y)\}$$

   where $\gamma(x, y)$ means

$$\exists d \in D_{Test}\,\big(pair\_converter^{-1}(d) = (x, y)\big)$$

   (d) Measure accuracy as:

$$\frac{|\{(x, y) \in Test : \exists (a, b) \in P\,((x, y) \approx (a, b))\}|}{|Test|}$$

4. Assuming test accuracy is acceptable, we can use this to predict the ground truth G.
   (a) Compute $S_X = \{l \in L : \exists x \in X\,(converter(x) = l)\}$, which is the set of all situations in which we need to make a decision.
   (b) Choose a model $m \in M$.
   (c) Compute $D_{Predicted} = \{l \in L : \exists s \in S_X\,(search(s, m) = l)\}$.
   (d) Convert to real numbers by computing

$$G_{Predicted} = \{(x, y) \in X \times Y : \delta(x, y)\}$$

   where $\delta(x, y)$ means

$$\exists d \in D_{Predicted}\,\big(pair\_converter^{-1}(d) = (x, y)\big)$$

2.2 Example of an implementable language

- There exist 4 bits bit_1, bit_2, bit_3 and bit_4, to which each $h \in H$ assigns a value.
- $V = \{a, b, c, d, e, f, g, h, i, j, k, l\}$ is a subset of all logical tests which might be applied to these 4 bits:

  - $a : bit_1 = 1$
  - $b : bit_2 = 1$
  - $c : bit_3 = 1$
  - $d : bit_4 = 1$
  - $e : bit_1 = 0$
  - $f : bit_2 = 0$
  - $g : bit_3 = 0$
  - $h : bit_4 = 0$
  - $i : j \wedge k$
  - $j : bit_1 = bit_3$
  - $k : bit_2 = bit_4$
  - $l : i \vee bit_2 = 1$

- $L = \{\{a, b, c, d, i, j, k, l\}, \{e, b, c, d, k, l\}, \{a, f, c, d, j\}, \{e, f, c, d\}, \{a, b, g, d, k, l\}, \{e, b, g, d, i, j, k, l\}, \{a, f, g, d\}, \{e, f, g, d, j\}, \{a, b, c, h, j, l\}, \{a, b, g, h, l\}, \{e, b, c, h, l\}, \{a, f, c, h, i, j, k, l\}, \{e, f, c, h, k\}, \{e, b, g, h, j\}, \{a, f, g, h, k\}, \{e, f, g, h, i, j, k, l\}\}$

2.3 Example of a task ω

- $S = \{\{a, b\}, \{e, b\}, \{a, f\}, \{e, f\}\}$
- $D = \{\{a, b, c, d, i, j, k, l\}, \{e, b, g, d, i, j, k, l\}, \{a, f, c, h, i, j, k, l\}, \{e, f, g, h, i, j, k, l\}\}$
- $M = \{\{i\}, \{j, k\}, \{i, j, k\}, \{i, l\}...\}$

2.4 Example of a child-task α of ω

- $S = \{\{a, b\}, \{e, b\}\}$
- $D = \{\{a, b, c, d, i, j, k, l\}, \{e, b, g, d, i, j, k, l\}\}$
- $M = \{\{i, j, k, l\}, \{b, d, j\}, ...\}$
  - Weakest model $\mathbf{m} = \{i, j, k, l\}$
  - Strongest model $\mathbf{e} = \{b, d, j\}$
  - $Z_{\mathbf{m}} = \{\{a, b, c, d, i, j, k, l\}, \{e, b, g, d, i, j, k, l\}, \{a, f, c, h, i, j, k, l\}, \{e, f, g, h, i, j, k, l\}\}$
  - $Z_{\mathbf{e}} = \{\{a, b, c, d, i, j, k, l\}, \{e, b, g, d, i, j, k, l\}\}$

3 Experiments

Accompanying this appendix is a Python script to perform two experiments using PyTorch with CUDA, SymPy and A^* [6, 7, 8, 9] (see commented code and for details). Context for and a detailed analysis of these experiments is given in [2]. What is given here is merely a brief technical report on the two experiments. In these two experiments, a toy program computes models to 8-bit string prediction tasks (binary addition and multiplication).

Implementable language: There were 256 states, one for every possible 8-bit string. The statements in L were expressions regarding those 8 bits that could be written in propositional logic ($\neg$, $\wedge$ and $\vee$). These statements were represented using PyTorch tensors or SymPy expressions.

Task: A task was specified by choosing $D \subset L$ such that all $d \in D$ 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.

3.1 Trials

Each of the two experiments (addition and multiplication) involved repeated trials 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 D_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 D_n.
   (b) A bit between 0 and 7 was then chosen, and S_n created by cloning D_n and deleting the chosen bit from every string (meaning S_n was composed of 16 different 7-bit strings, each of which could be found in an 8-bit string in D_n).
2. A child-task $T_k = \langle S_k, D_k, M_k \rangle$ was sampled from the parent task T_n. Recall, $|D_k|$ was determined as a parameter of the trial.
3. From T_k two models (rulesets) were generated; a weakest c_w, and a MDL c_{mdl}.

Testing phase: For each model $c \in \{c_w, c_{mdl}\}$:

1. The extension Z_c of c was then generated.
2. A prediction D_{recon} was then constructed s.t. $D_{recon} = \{z \in Z_c : \exists s \in S_n \ (s \subset z)\}$.
3. D_{recon} was then compared to the ground truth D_n, and results recorded.

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

3.2 Measurements

Rate at which models generalised completely: Generalisation was deemed to have occurred where $D_{recon} = D_n$. 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.

Average extent to which models generalised: Even where $D_{recon} \neq D_n$, the extent to which models generalised could be ascertained. $\frac{|D_{recon} \cap D_n|}{|D_n|}$ was measured and averaged for each value of $|D_k|$, and the standard error computed.

Table 1. Results for Binary Addition

	c_w				c_{mdl}			
$\lvert D_k\rvert$	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 2. Results for Binary Multiplication

	c_w				c_{mdl}			
$\lvert D_k \rvert$	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

4 List of proofs

These proofs support the claims of [10]. Longer and more detailed versions of these and other proofs are given in [2].

Proposition 1 (sufficiency). *Weakness is a proxy sufficient to maximise the probability that induction generalises from α to ω.*

Proof: You're given $\alpha = \langle S_\alpha, D_\alpha, M_\alpha \rangle$ and a hypothesis $\mathbf{h} \in M_\alpha$. Let $\omega = \langle S_\omega, D_\omega, M_\omega \rangle$ be the parent to which we wish to generalise:

1. The set of statements which *might* be decisions addressing situations in S_ω and not S_α, is $\overline{Z_{S_\alpha}} = \{l \in L : l \notin Z_{S_\alpha}\}$.
2. For any given $\mathbf{h} \in M_\alpha$, the set of decisions $\mathbf{h}$ implies which fall outside the scope of what is required for the known task α is $\overline{Z_{S_\alpha}} \cap Z_\mathbf{h}$.
3. Increasing $|Z_\mathbf{h}|$ increases[5] $|\overline{Z_{S_\alpha}} \cap Z_\mathbf{h}|$, because $\forall z \in Z_m : z \notin \overline{Z_{S_\alpha}} \rightarrow z \in Z_{S_\alpha}$.
4. $2^{|\overline{Z_{S_\alpha}}|}$ is the number of tasks which fall outside of what it is necessary for a model of α to generalise to, and $2^{|\overline{Z_{S_\alpha}} \cap Z_\mathbf{h}|}$ is the number of those tasks to which a given $\mathbf{h} \in M_\alpha$ does generalise.
5. The probability that a model $\mathbf{h} \in M_\alpha$ generalises to the unknown parent task ω is

$$p(\mathbf{h} \in M_\omega \mid \mathbf{h} \in M_\alpha, \alpha \sqsubset \omega) = \frac{2^{|\overline{Z_{S_\alpha}} \cap Z_\mathbf{h}|}}{2^{|\overline{Z_{S_\alpha}}|}}$$

$p(\mathbf{h} \in M_\omega \mid \mathbf{h} \in M_\alpha, \alpha \sqsubset \omega)$ is maximised when $|Z_\mathbf{h}|$ is maximised.

Proposition 2 (necessity). *To maximise the probability induction generalises from α to ω, it is necessary to use weakness, or a function thereof, as proxy.*

Proof: Let α and ω be defined exactly as they were in the proof of prop. 1.

1. If $\mathbf{h} \in M_\alpha$ and $Z_{S_\omega} \cap Z_\mathbf{h} = D_\omega$, then it must be he case that $D_\omega \subseteq Z_\mathbf{h}$.
2. If $|Z_\mathbf{h}| < |D_\omega|$ then generalisation cannot occur, because that would mean that $D_\omega \not\subseteq Z_\mathbf{h}$.
3. Therefore generalisation is only possible if $|Z_m| \geq |D_\omega|$, meaning a sufficiently weak hypothesis is necessary to generalise from child to parent.
4. The probability that $|Z_m| \geq |D_\omega|$ is maximised when $|Z_m|$ is maximised. Therefore to maximise the probability induction results in generalisation, it is necessary to select the weakest hypothesis.

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

[5] Monotonically.

References

[1] M. T. Bennett. *Enactivism & Objectively Optimal Super-Intelligence*. 2023. URL: `https://arxiv.org/abs/2302.00843`.

[2] M. T. Bennett. *The Optimal Choice of Hypothesis Is the Weakest, Not the Shortest*. 2023. URL: `arxiv.org/abs/2301.12987`.

[3] J. Rissanen. "Modeling By Shortest Data Description*". In: *Autom.* 14 (1978), pp. 465–471.

[4] M. T. Bennett. *Emergent Causality & the Foundation of Consciousness*. 2023. URL: `arxiv.org/abs/2302.03189`.

[5] J. Pearl and D. Mackenzie. *The Book of Why: The New Science of Cause and Effect*. 1st. New York: Basic Books, Inc., 2018.

[6] A. Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *NeurIPS*. USA: Curran Assoc. Inc., 2019.

[7] D. Kirk. "NVIDIA Cuda Software and Gpu Parallel Computing Architecture". In: *ISMM '07*. Canada: ACM, 2007, pp. 103–104.

[8] A. Meurer et al. "SymPy: Symbolic computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017), e103. DOI: `10.7717/peerj-cs.103`.

[9] P. E. Hart, N. J. Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[10] M. T. Bennett. "Symbol Emergence and the Solutions to Any Task". In: *Artificial General Intelligence*. Cham: Springer, 2022, pp. 30–40.