UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Formal appreciation of art

Author:
Maksym ZHURAVINSKYI

Supervisor:
Andryi GAZIN

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2022

Declaration of Authorship

I, Maksym ZHURAVINSKYI, declare that this thesis titled, "Formal appreciation of art" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Perfect is the enemy of good"

Voltaire, Questions sur l'Encyclopédie

"He gets all applause who has mingled the useful with the pleasant"

Horace, Ars Poetica

"My speech is imperfect. I speak in images. With nothing else can I express the words from the depth"

Jung, Red Book

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Formal appreciation of art

by Maksym ZHURAVINSKYI

Abstract

Our objective is to construct a simple computation model of perception in order to express a subjective comparison between two objects based on their extrinsic ability to be perceived in the eyes of an observer and their effect on the observer's world perception. The basis of the comparison we're interested in is often referred to as beauty, salience, interestingness, or aesthetic preference, yet we concede the completeness with which these notions are tasked to deal in favor of a core, conceptual formalism. We express perception's end goal to be making short descriptions of objects within some language and formalize this process with equality saturation. We examine mechanisms aiding the improvement of language to keep descriptions short and how it relates to perceived objects' relative worthiness given an observer's language and history of experience.

Acknowledgements

Many thanks to my advisor Andriy Gazin for his guidance and constant feedback, Mira Shulyak for fruitful discussions, and Oleg Farenyuk for miscellaneous support. I'm beyond obliged to my family, friends, and foes. Also, I'm very grateful to the Ukrainian Catholic University staff for putting their heart and soul into their work.

Contents

List of Figures

List of Tables

List of Symbols

λ_m	λ-term; description
λ	pair of λ-terms
Λ	set of pairs of λ-terms; language
$\lambda_1(\lambda_m)$	set of abstractions from one step of inverse β-reduction of λ_m
$\lambda(\lambda_m)$	complete set of abstractions from λ_k
x	object
x^Λ	set of descriptions of x in the language Λ
x_m^Λ	description of x in the language Λ
x_0^Λ	the primitive description of x in the language Λ
x_*^Λ	the shortest description of x in the language Λ

Dedicated to anyone who opposes evil both inwardly and externally

Chapter 1

Introduction

With the advent of ubiquitous deep generative models that have uncharted scaling limits (Gwern, 2020b) and which, in principle, can generate almost anything, a large subset of which attempts to create art specifically (visual (Ramesh et al., 2022), music (Dhariwal et al., 2020), fiction (Gwern, 2020a)), a captious question arises: artistically speaking is any of it good? Or rather: can the artistic value be explained formally, in the same or similar way as the artwork can be generated? Formal methods of judging artistic value are severely lagging behind those of generation, and unlike any other technology which redefined the art and lowered the skill threshold needed for its creation (camera, digital audio workstation), generative models, unless constrained, are destined for a complete informational flood with their artistry without requiring any human input whatsoever. There is already a surplus of the creative content of various quality, some of it already generated, and by the sheer amount, natural discoverability becomes pointless. Recommendation engines and algorithmic feeds only partially solve the problem (or maybe exacerbate, given the tendency of over-reliance on and even anthropomorphization of the algorithm itself (Freeman, Gibbs, and Nansen, 2022)), since they operate on tangential metrics such as engagement, gaming which has turned into an industry of its own.

But that's only a tiny sliver of the problem, or rather something taken from the zeitgeist, yet we're interested in far detached questions: what are things really, and what is their worth? How do we attribute the concept of beauty to some specific objects? How can the very same objects be seemingly inconspicuous or even debased in the eyes of a different observer? These questions were tried for centuries and given their monumentally, only with partial success. We will also take our chance at attempting to address those. However, our methodology will, by design, be too elementary to grapple with them in their entirety, nor will it surpass any of the existing philosophical treatment, and it will also lack the empirical evidence justifying itself. Despite that, we hope it will offer a straightforward and uncomplicated model formalizing the artistic value by relying upon as little as possible.

Concretely our objective is to construct a mapping from any object (either tangible or abstract) to a real value, which we call the subjective goodness that will serve as a proxy for a measure of beauty, aesthetic, or artistic value, but we leave the tightening of this connection, perhaps with real-world experiments, for the future work. There already has been convincing and also theoretical research in connecting beauty with simplicity framed in terms of algorithmic complexity (Section 2.1). However, we think that the generality in its formulation (it operates on opaque terms) and in its result (it's way too generous) is incomplete. Here, we want to sacrifice a level of generality to gain a closer look at possible mechanism of how an observer might judge the artistic value of objects based on how he makes descriptions of them. While the process of describing is often associated purely with communication, we equate perception as making descriptions for one-self, perhaps in

a language that has no requirement of being understandable by others or being utterable at all. We make descriptions in the form of programs using lambda-calculus (Section 3.1), which upon execution, reproduce an object losslessly. Then we give motivation as to what is a prerequisite for a system that makes such descriptions, which we call a language (Section 4.1), and how we can extend it in order to render it to be more efficient, which we call abstraction (Section 4.2). Then we describe a procedure for making and storing multiple descriptions with e-graphs (Section 4.3) and then explain what we think the object's worthiness is (Section 4.4). After we give preliminary examples both in a trivial domain (Section 5.1) and music domain (Section 5.2) of what we mean by descriptions and abstractions and give an example of our objective, a subjective goodness mapping, in action.

Chapter 2

Related work

2.1 Formal Theory of Creativity

Formal Theory of Creativity (Schmidhuber, 2008; Schmidhuber, 2010) describes beauty in the context of reinforcement learning (Kaelbling, Littman, and Moore, 1996) and recurrent neural networks (Hochreiter and Schmidhuber, 1997). It asserts that a reinforcement learning agent's (observer's) main objective, subserving the objective put up by the environment he is in, is to compress its history of observations or more formally is to maximize an intrinsic reward signal r_{int} as a function[1] measuring the rate of change in the efficiency of the observer's compression algorithm applied on agent's observation history $h(\leq t+1)$.

$$r_{\text{int}}(t+1) = f(C(p(t), h(\leq t+1)), C(p(t+1), h(\leq t+1)))$$

Using the observer's computationally limited encoding model $p(t)$, $C(p(t), H(\leq t+1))$ is the number of bits needed to encode his history bounded from below by Kolmogorov complexity $K(h(\leq t+1))$ (Kolmogorov, 1965; Solomonoff, 1964; Chaitin, 1966). Beauty $B(D \mid O, t)$ of a piece of data D for an observer O at time t is defined as a negative number of bits required to encode D given the observer's state, implying that the subjectively most beautiful is the one with the shortest description length using the observer's particular method of compression. Additionally, subjective interestingness, surprise, or aesthetic value $I(DO, t)$ is defined as the first derivative of subjective beauty:

$$B(D \mid O, t) = -C(p(t), D)$$
$$I(D \mid O, t) = \frac{\partial B(D \mid O, t)}{\partial t}$$

As a result, an observer wants to seek novel observations but still orderly enough to be compressible on time, riddled with undiscovered patterns or symmetries, yielding an improvement to the observer's model of the world in the fastest way possible. Given its simplicity, the theory knowingly gives up on the notion that an object might be beautiful because it reminds of or references something, but rather it judges its form only. Nevertheless, there is also an inconsistency: in the example given in (Schmidhuber, 2008; Schmidhuber, 2012), a pitch-black room viewed by a vision-based agent has to be boring and not interesting, yet according to the theory, it remains beautiful due to its almost non-existent complexity. And if we compare a dimmed Louvre with the same empty dark room, the latter being far more compressible no matter how much any observer can become acquainted with either, implying that it is also far more beautiful is surely inadmissible. Instead, to confine beauty

[1] This definition is copied ad verbatim for consistency, however by f, $f(a, b) = a - b$ is implied

with more precision, we might have to investigate further intermediaries of the encoding process itself. Our approach described here will not deviate greatly, at least ideologically, from this theory, in so far as we delegate all our philosophical line of reasoning to it. And while it may be possible to frame it in the spirit of this work instead, not devoid of the said spirit, we will try to delineate beauty using methods that give concrete descriptions of objects using lambda calculus and equivalence graphs.

Chapter 3

Background

3.1 Lambda calculus

Lambda calculus (hereafter referred to as λ-calculus) is a simple formal system and a notational tool discovered by Alonzo Church during the 1930s (Hindley and Seldin, 2008), which is capable of representing every computable function using an intuitively simple syntax: keywords "λ" and "." together with parenthesis "(", ")" and an infinite alphabet to disambiguate the naming of variables are sufficient. An expression (also referred to as a λ-term) from λ-calculus is either a:

1. Variable a, where the symbol a is drawn from some infinite alphabet.
2. Application (ab), if a and b are λ-terms
3. Abstraction $(\lambda a.b)$, if a is a variable and b is a λ-term

Abstraction is the central idea in λ-calculus, it formalizes a function of one variable, for example $(\lambda a.b)$ is akin to a function of variable a which if contained within function's body b would be bound to the argument supplied upon an application. The function of multiple variables can be expressed as the chaining of abstractions $(\lambda a.\lambda b.\lambda c.d)$ which is often shorthanded notationally to $(\lambda abc.d)$. To compute in lambda calculus means to simplify the application of abstraction $((\lambda a.b)c)$, which is called a redex, and for that, we need a single rule called β-reduction:

$$((\lambda a.b)c) \longrightarrow_\beta [a \to c]b$$

Where $[a \to c]$ is an operation which replaces every occurrence of bounded variable a with c. A term without redexes or equivalently a term in which β-reduction is no longer applicable is said to be in the β-normal form and treated as a final result of a computation. Not every term has a β-normal form, but if the term has a β-normal form, it has only a single one (up to renaming of binding variables, which is called α-reduction). Because of λ-calculus pureness, to compute something useful, we need to overlay special interpretations to certain terms, a process called the Church encoding. For example, to compute something with numbers, we first need to define what numbers are (Figure 3.1).

This particular encoding of numbers is not the only one possible similarly, as there are numerous ways to interpret a collection of bits, few equivalent extensionally, there are plenty of ways of defining numbers together with operations on them in λ-calculus. In most cases, it's practical to extend λ-calculus to permit constants without explicit encoding, assuming there always is one, leaving its meaning to be interpreted during the reduction. To show an example of reduction, we define a successor function S which takes a single number as an argument and increases it by 1 (Figure 3.2).

$$
\begin{aligned}
0 &\longleftrightarrow (\lambda sz.z) \\
1 &\longleftrightarrow (\lambda sz.(sz)) \\
2 &\longleftrightarrow (\lambda sz.(s(sz))) \\
&\vdots \\
n &\longleftrightarrow (\lambda sz.(s^n z))
\end{aligned}
$$

FIGURE 3.1: Encoding known as Church numerals, arrow represents encoding

$$
\begin{aligned}
S &\longleftrightarrow (\lambda nsz.(s((ns)z))) \\
(S\ 0) &\longleftrightarrow ((\lambda nsz.(s((ns)z)))(\lambda sz.z)) \\
&\longrightarrow_\beta (\lambda sz.(s(((\lambda sz.z)s)z))) \\
&\longrightarrow_\beta (\lambda sz.(s((\lambda z.z)z))) \\
&\longrightarrow_\beta (\lambda sz.(sz)) \\
&\longleftrightarrow 1
\end{aligned}
$$

FIGURE 3.2: Successor function and an example of β-reduction

There is also exists an opposite operation to β-reduction — an inverse β-reduction, also referred to as β-expansion or abstraction as a verb. The idea is to abstract away from the specifics of the term one step at a time:

$$
\begin{aligned}
(ab) &\longrightarrow_{I\beta} ((\lambda x.(xb))a) \\
(ab) &\longrightarrow_{I\beta} ((\lambda x.(ax))b)
\end{aligned}
$$

Also, we want to extract the set of all possible abstractions from the exact λ-term, using inverse β-reduction but while discarding the final application:

$$\lambda_1(a) = \{(\lambda x.b) \mid \exists c\ ((\lambda x.b)c) \longrightarrow_\beta a\}$$

For example the expression 1 + 2, and the corresponding λ-term ((+ 1) 2), would abstract in one step to an "add-1" abstraction $f(x) = 1 + x$, "add-2" abstraction $f(x) = x + 2$ and the "binary-apply" abstraction $f(g) = g(1, 2)$. These abstractions would form a set $\lambda_1(1 + 2)$, and to extract more abstract abstraction we repeat the same process until the fixed-point is reached, that is $\lambda_2(a) = \lambda_1(\lambda_1(a))$ and $\lambda(a) = \lambda_\infty(a)$. Lastly, and also most importantly, we need the length of λ-terms which we define as:

$$
\begin{aligned}
|a| &= 1 \text{ if } a \text{ is a variable or constant} \\
|(ab)| &= |a| + |b| \\
|(\lambda a.b)| &= |b|
\end{aligned}
$$

Chapter 4

Approach

4.1 Motivation

Before we can speak of comparing things, we must have a way of representing them first. Since most of the art we consume nowadays comes primarily through digital screens, we assume any art we're interested in can be digitized and therefore can be shaped into a representation that corresponds to universal computers, be it combinatory logic, λ-calculus, type system etc. (Morales, 2018). While it might be more proper to focus on how human minds represent things, there is no consensus on how exactly it is, and be it either in a language of thought (Fodor, 1975) or through pure connectionism (Bechtel, 1991), we assume that at least same simplicity and laconisity which will be essential for our purposes must also be present in whichever it is, since human mental capacity is limited. Yet our goal here is an accurate model of human aesthetics, but something far more elementary, which is why the question of exact representation for us is not crucial. Therefore by perceiving or representing an object, we mean to compress its description from a crude sensory input to some sparse coding. Despite an inevitable loss of information, those two descriptions can remain equivalent in most contexts, with the benefit of one being much shorter. Such distillation of information is done by compression algorithms, creation of which is a challenging task in the most optimal sense since there exists an uncomputable limit, approximating which is also a long-standing problem in a practical sense (Hutter, 2006), so challenging that the compression has been closely associated with both comprehension (Chaitin, 2004) and intelligence (Hutter, 2009) itself. While there is ongoing research as to through what exact schema compression manifests inside our brains (Kwak and Curtis, 2022), we can at least see the impact of it in our daily language: we use abbreviations, we invent new words to refer to specific events, in the end, we imply things by using silence, all to not speak a single syllable more than needed.[1] We use this analogy to refer to a stateful part of the observer's compression algorithm as an observer's language.

4.2 Language

Here we consider languages in which there is always at least a procedure to describe objects in terms of bare primitives, albeit rather redundantly, and starting from this primitive description we use the capacity of language to further compress it. More precisely we define a language Λ to be a set of pairs of descriptions that can be used interchangeably:

$$\Lambda = \{\lambda\} = \{\lambda_\ell \longleftrightarrow \lambda_r\}$$

[1] Natural languages are still redundant character-wise, yet most of misinterpretations and inconsistencies come primarily from being overly succinct

With each side of such pair there is a corresponding length measure, here in place of descriptions we use λ-terms, so the length of the description is the length of the corresponding λ-term. Pairs of descriptions (which we call abstractions or equivalences) don't have to be non-overlapping or exclusive, which implies that there can be multiple ways to describe the same thing. For an object x we notate x^Λ to be the set of all descriptions in the language Λ equivalent to the one made primitively which is notated by x_0^Λ, while all other individual descriptions are notated by x_k^Λ and the shortest one is x_*^Λ. Equivalences can be brought up either by a process of renaming ("the Sun" $\longleftrightarrow$ "the star from the Solar System" or length-of-lists(ℓ) $\longleftrightarrow$ (map len ℓ)) like in Dreamcoder (Ellis et al., 2021) where language is referred to as a library, or by enumerating expressions finding ones semantically equivalent ($|\,a\,|^2 \longleftrightarrow a^2$) as in Ruler (Nandi et al., 2021) where language is referred to as rewrite rules. In both of these systems, every new abstraction brings an exponential cost associated with it, so the introduction of new ones has to be considered carefully. Moreover, there exists a danger of unintentionally equating one too many things, rendering the whole language to be totally inconsistent[2]. Here, similarly to a "crude" Minimal Description Length objective (Grunwald, 2004) we aim to make short descriptions while not complicating the language itself too much. Thus we define the helpfulness or goodness of each abstraction so far as it helps in compressing a description of an object, in the number of symbols saved when employing it versus when not, plus the additional cost of the definition:

$$\log \kappa(\lambda \mid x_*^\Lambda, \Lambda) = |x_*^{\Lambda-\lambda}| - |x_*^{\Lambda+\lambda}| + |\lambda| \tag{4.1}$$

Where $\Lambda - \lambda$ and $\Lambda + \lambda$ are languages with or without an abstraction λ, and the length of the definition of λ is the length of both its sides $|\lambda_\ell| + |\lambda_r|$. Hence an observer must refine his language stripping it from archaic words and adding new apt ones that use abstractions which are easy to define in terms of his current language and help describe many things in the clearest way possible.

4.3 Confluent descriptions

To create and compactly represent each description, we use an e-graph (equivalence graph) data structure (Nelson, 1980), which groups together equivalent descriptions into disjointed sets called e-classes, enabling resharing common parts of them without redundancy. Starting from the primitive description, to create new ones we use equivalences from the language and e-matching algorithm from Simplify theorem prover (Detlefs, Nelson, and Saxe, 2005), which scans an e-graph for the already present one side of equivalence and instantiates and equates with it its other side, given proper substitutions if any, while also making sure that any two descriptions which include each of two sides of the newly found equivalence are also made equivalent, in other words maintaining congruence relation $a \equiv b \implies f(a) \equiv f(b)$. Emerging equivalences enable finding more equivalences, so e-matching proceeds indefinitely until a fixed point, when any subsequent appliance of e-matching doesn't change the e-graph in any way, at which point the e-graph is saturated. After that, we may want to select a single representative description with the minimal length or some other optimal description according to some cost function, using the e-class analysis technique described in the egg library (e-graph good)(Willsey et al.,

[2] As does equating *true* with *false*, or $\lambda xy.x$ with $\lambda xy.y$ equivalently

2021). Importantly, during the process of saturation, the information only accumulates, allowing for us, in the end, to operate over both the initial description, the most optimal one, and any descriptions in-between at the same time. Thus we can extend the notion of the goodness of abstraction (Equation 4.1) over all possible descriptions of an object discounted by their importance:

$$\kappa(\lambda \mid x, \Lambda) = \sum_{m}^{|x^\Lambda|} 2^{|x_*^\Lambda| - |x_m^\Lambda|} \kappa(\lambda \mid x_m^\Lambda, \Lambda) \;\geq\; \kappa(\lambda \mid x_*^\Lambda, \Lambda) \tag{4.2}$$

We judge short descriptions to be more important than longer ones, whereas the role of degeneratively long descriptions is nullified. It is done in a similar vein as with algorithmic probability (Solomonoff, 1964), with the difference being that we care only about the relative length in comparison with the shortest description. This way, we don't overlook inherently long descriptions by themselves, but the input of each alternative description diminishes with its length, and if there is more than one equally short description, each reusing the same abstraction, their contribution is, on the contrary, compound. By different descriptions, we mean everything which can be said about the object and just as objects or events with multiple interpretations, yet each being drastically different or rooted in many contexts are not uncommon, as they are foundational for making jokes, for example, here we're treating them as efficient storage of abstractions.

The reason we use e-graphs for the production of descriptions, instead of more complete methods like enumeration search over the language, is that the former is far more tractable, and we think it reflects how descriptions are made by us in our languages since every description produced in this way will relate to our data unlike in the case of enumeration. Nevertheless, to create even shorter descriptions we would need to resort to some form of enumeration in order to extend our language with useful abstractions, and this is how we transition from measuring lone abstractions' goodness to the goodness of objects themselves.

4.4 Goodness

"A book or a speech for example is said to have a great deal in it, to be full of content in proportion to the greater number of thoughts and general results to be found in it" (Hegel, 1874). So far, we've examined the goodness of individual abstractions (thoughts, results, propositions), but where from should they come? While making descriptions of objects in immediacy doesn't involve any modification of the language, if an observer isn't satisfied with the description he arrived at, he can spend additional time looking for a better one while extending his language. In a way, to describe an object optimally observer inadvertently must arrive at certain abstractions, even if he had not possessed them before, and we say that these abstractions are "in" an object. Also, we have to describe how difficult it is to arrive at them and how good they are for each observer. Starting from the latter, as in (Section 2.1) we also assume that observer's main intent is to make the shortest description of his history so that the usefulness of each abstraction is measured as far as it aids this objective. As for derivability, we state that an object implies abstractions unequally in proportion to their goodness (Equation 4.2). And since we only examine one facet of extracting abstractions in a uniform order, from (Section 3.1), we also discount the goodness of an object by their total amount present. Taken together, the goodness

of a thing for an observer with a language Λ and a history H is proportional to the availability of good abstractions in it discounted by their derivability:

$$\kappa(x \mid H, \Lambda) = \frac{1}{|\lambda(x^\Lambda)|} \sum_{\lambda \in \lambda(x^\Lambda)} \kappa(\lambda \mid x, \Lambda)\, \kappa(\lambda \mid H, \Lambda) \tag{4.3}$$

Objects which yield a few but strong abstractions are in higher regard than those with plenty but shallow ones. The more complex or "lengthy" is an object, the more abstractions it would have by inevitability, hence we prefer things simple as possible but not too simple. We also don't regard any already known abstractions by an observer condescendingly so that any cliches or banalities, unless they have some utility, are treated as nothing more than dead weight. Also, we don't care about the origin or the creator of an object we consider, except when it's due to constructing a corresponding description using this information. In summary, to state anything about an object's goodness, we only need to examine how good are abstractions present in their descriptions.

Chapter 5

Proof of concept

5.1 Number-list language

As a preliminary example, consider a trivial language describing lists of natural numbers. Numbers are encoded using Church numerals and lists, by a function that takes an element and a list (starting with an empty one) and prepends an element to it.

TABLE 5.1: Primitives from the number-list language

name	description	λ-term
0	zero	$(\lambda sz.z)$
S	successor function	$(\lambda nsz.(s((ns)z)))$
ø	empty list	$(\lambda x.x)$
.	list constructor	$(\lambda htf.((fh)t))$

Here λ-terms are shown for demonstrative purposes, in the implementation primitives are assigned constants. Using this language, so far, there's only one way to describe lists which is the primitive one as shown in (Table 5.1). Given this particular list x, using (Equation 4.2), we can measure the goodness of abstractions taken from $\lambda(x^{\Lambda})$, a handful of which are shown in (Figure 5.2), where brackets denote free variables with number and list types. Despite that, it's evident that defining shorthands for numbers is first-most expedient, we ignore this suggestion and, for illustrative purposes, select two different abstractions (Figure 5.3), give them appropriate names, and include them in our language. As a result, it's now possible to describe x in a few more ways, one of which is the shortest (Figure 5.4).

$$x = [1, 2, 3]$$
$$x_0^{\Lambda} = (.\ (S\ 0)\ (.\ (S\ (S\ 0)\ (.\ (S\ (S\ (S\ 0)))\ ø)))$$

FIGURE 5.1: The primitive description from the number-list language

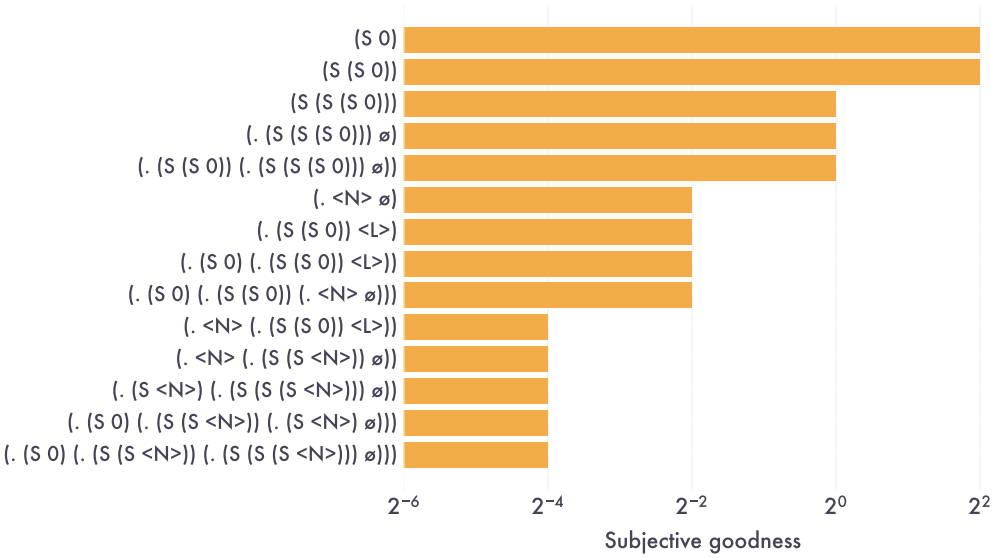


FIGURE 5.2: Relative goodness of abstractions

$$\text{starts-with-1-and-2} \longleftrightarrow (\lambda x.(.\ (S\ 0)\ (.\ (S\ (S\ 0)\ x))))$$
$$\text{ends-with-3} \longleftrightarrow (.\ (S\ (S\ (S\ 0))\ \text{ø})$$

FIGURE 5.3: Additional abstractions for the number-list language

$$x_1^{\Lambda} = (\text{starts-with-1-and-2}\ (.\ (S\ (S\ (S\ 0))\ \text{ø}))$$
$$x_2^{\Lambda} = (.\ (S\ 0)\ (.\ (S\ (S\ 0))\ \text{ends-with-3})$$
$$x_*^{\Lambda} = x_3^{\Lambda} = (\text{starts-with-1-and-2}\ \text{ends-with-3})$$

FIGURE 5.4: Descriptions using abstractions from the number-list language

5.2 Scales language

As for a more involved example, we will make descriptions of jazz licks, which are short but informationally very dense pieces of monophonic music. We choose this domain due to the simplicity and the compressibility of musical phrases, which are not unindicative of their inherent expressivity. What follows is a little introduction to music theory upon which we build our language for descriptions.

5.2.1 Brief music theory introduction

For simplicity, we'll be only considering the frequency domain, in so far as we're interested only in the frequencies of each musical note. The most fundamental principle in music is the concept of octave equivalence or octave circularity, where an octave is a difference between two frequencies, one of which is twice the frequency of the other. The human ear perceives two such pitches (where the pitch is a perceived frequency) as nearly the same, and this effect also occurs in other mammals (Wright et al., 2000). With this in mind, musicians use tuning systems that quantize a range within an octave into some number of pitches, treating the rest of the frequency space as a continuation of the same pattern. Equal temperament, the most widespread tuning system in the Western world, divides an octave into 12 pitches, each equally spaced within the range by a multiple of $\sqrt[12]{2}$ of the frequency of their predecessors. With this quantization, we only need to think about twelve unique pitches, and to each, we can assign a natural number. It's common to group pitches to form a scale, which is a list of strictly ascending (modulo 12) unique numbers. The most primitive scale is called chromatic, which has all 12 pitches within the octave, while other popular ones like major scale or pentatonic scale have 7 or 5 total. There are twelve different variations of each scale, each equivalent up to a shift of all its pitches by another pitch, called a key. We use scales to construct a more narrow view of the notes, and for that, we need a concept of indexing a scale which is the same thing as indexing an array. Often with pop music, the whole song is written in one scale and one the key, yet the same doesn't hold for jazz, where both are subject to frequent changes.

$$
\begin{aligned}
\text{chromatic} &= [\text{C}, \text{D}\flat, \text{D}, \text{E}\flat, \text{E}, \text{F}, \text{G}\flat, \text{G}, \text{A}\flat, \text{A}, \text{B}\flat, \text{B}] \\
&= [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] \\
\text{minor} &= [0, 2, 3, 5, 7, 8, 10] \\
\text{C minor} &= 0 + \text{minor} = [0, 2, 3, 5, 7, 8, 10] \\
\text{D}\flat\ \text{minor} &= 1 + \text{minor} = [1, 3, 4, 6, 8, 9, 11] \\
\text{G}\flat\ \text{minor} &= 6 + \text{minor} = [6, 8, 9, 11, 1, 2, 4] \\
\text{G}\flat\ \text{minor}[1, 3, 5] &= [6, 9, 1] = [\text{G}\flat, \text{A}, \text{D}\flat]
\end{aligned}
$$

FIGURE 5.5: Pitch names, scales and scale indexing

5.2.2 Language

We reuse primitives from the number-list language to construct a new language, where pitches correspond to numbers and scales and licks to doted pairs. Extra

primitives (Table 5.2.2) and abstractions (Figure 5.6) come from music theory. Abstraction 1 defines scales as equivalences between different scale degrees. Abstraction 2 defines multi-indexing (indexing with multiple indices). Abstraction 3 defines relative indexing with special primitives $\uparrow, \downarrow$, which indexes the next or the previous scale degree from the indexed before. Abstractions 4 and 5 define the looping of relative indices. In the implementation, we don't enumerate rules for each special case, but instead, we use wildcard variables and store and look up context information in e-class analyses. So far, this language enables us to talk about scales and runs through the scales, but surely it is not exhaustive: there are no concepts of approaches, enclosures, or even relative progression of chords, but for our purposes of making slightly more high-level descriptions it is sufficient enough.

TABLE 5.2: Primitives from the scales language

$\mathbb{N}$	natural numbers (Church numerals)
minor, diminished, etc.	scales
scale[$\mathbb{N}$]	indexing of the scale
$\mathbb{N}$ + scale[$\mathbb{N}$]	shift key of the scale (e.g. D♭ minor)
$\uparrow, \downarrow$	next, previous index (+1, -1)
loop $\mathbb{N}$ $\uparrow, \downarrow$	repeat $\uparrow, \downarrow$ $\mathbb{N}$ times

1. D♭ chromatic[1] $\longleftrightarrow$ D♭ minor[1] $\longleftrightarrow$ C locrian[2] $\longleftrightarrow$ F minor[6] etc.
2. D♭ minor[1], D♭ minor[2] $\longleftrightarrow$ D♭ minor[1, 2]
3. D♭ minor[1, 2] $\longleftrightarrow$ D♭ minor[1, $\uparrow$]
4. D♭ minor[$\uparrow$] $\longleftrightarrow$ D♭ minor[loop 1 $\uparrow$]
5. D♭ minor[loop $\mathbb{N}$ $\uparrow$, $\uparrow$] $\longleftrightarrow$ D♭ minor[loop $\mathbb{N} + 1$ $\uparrow$]

FIGURE 5.6: Abstractions for the scales language

5.2.3 Descriptions

A jazz lick, being simply a collection of some pitches, can be primitively described as a list of numbers, the same way as in the number-list language. Starting from this primitive description embedded in e-graph we use previously defined language to find more compressed representations, finishing with the one of the shortest length, one which we visualize together with the sheet music representation as in (Figure 5.7).[1] We use the example of the scales language to show that it's possible to arrive at a representation that will loosely resemble something coherent, or something which can be seen as a result of some theory analysis, by principles we've examined in the preceding sections. Next, we employ those descriptions to compare jazz licks among themselves.

[1]The rest of jazz licks together with their descriptions, and their sources can be found in the appendix

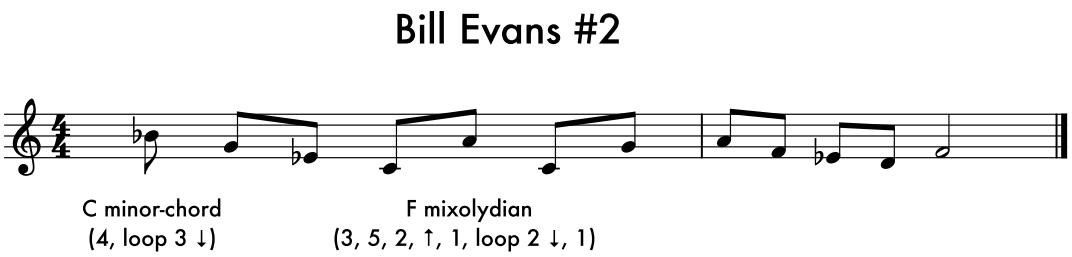


FIGURE 5.7: Description of a jazz lick in the scales language

5.2.4 Comparison

To measure subjective goodness, we need some language and some history of observations, both of which are strenuous to apprehend empirically. Nevertheless, we make some concessions: we set the observer's language to be the language we've described so far and his history of experience to a seamless concatenation of jazz licks in a randomized order. The subjective goodness measured in this way must be skewed towards long licks since they constitute a larger portion of the observer's auditory experience and, by themselves, explain most of it. Due to that, we also plot lengths of each lick (number of notes wise) and lengths of their minimal descriptions in our scales language (Figure 5.9). Implementation-wise, to reduce the computation time needed to measure the goodness of individual licks, we select abstractions from each lick by using only one step of inverse β-reduction, plus we don't integrate all of the descriptions (given how our language is constructed there is a countless amount of which), instead, we take the shortest description plus one thousand randomly sampled ones, for both the history and licks.

As for the final comparison (Figure 5.8), while we personally more or less agree on the ranking given, we might do so for different reasons since we speak a bit different language from the one shown here and our experiences are way unalike.

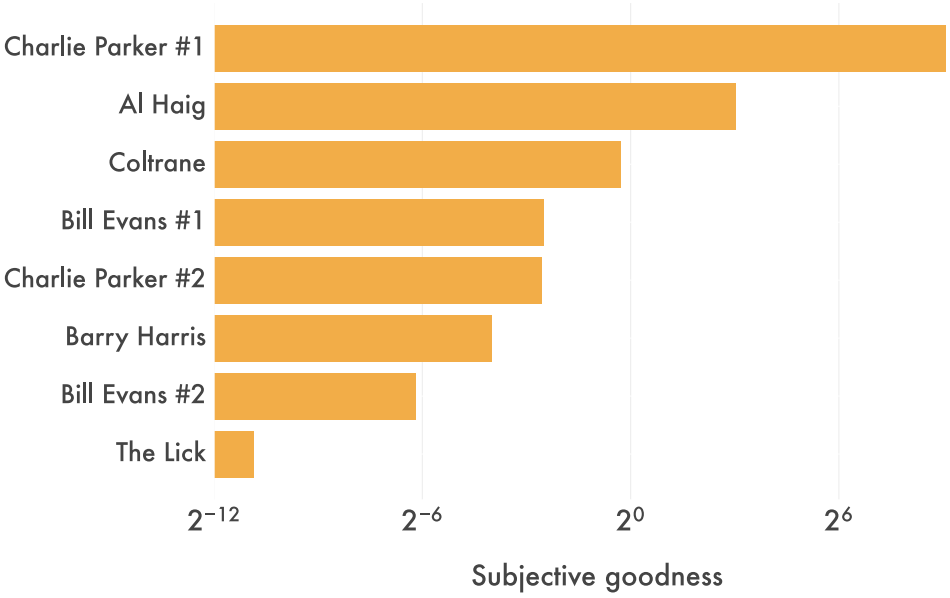


FIGURE 5.8: Subjective comparison of jazz licks

It's evident that there is some correlation between length and subjective goodness by construction, yet there is also an inverse relationship between subjective goodness and simplicity of descriptions, which is something we contemplated in (Section 2.1). And since it's practically unfeasible to come up with a more justified experiment, which wouldn't make arbitrary assumptions just as we did, our goal is more than fulfilled since what we were interested in was a procedure by which posed in the very beginning questions could be answered, not as much in the exact answer by itself.

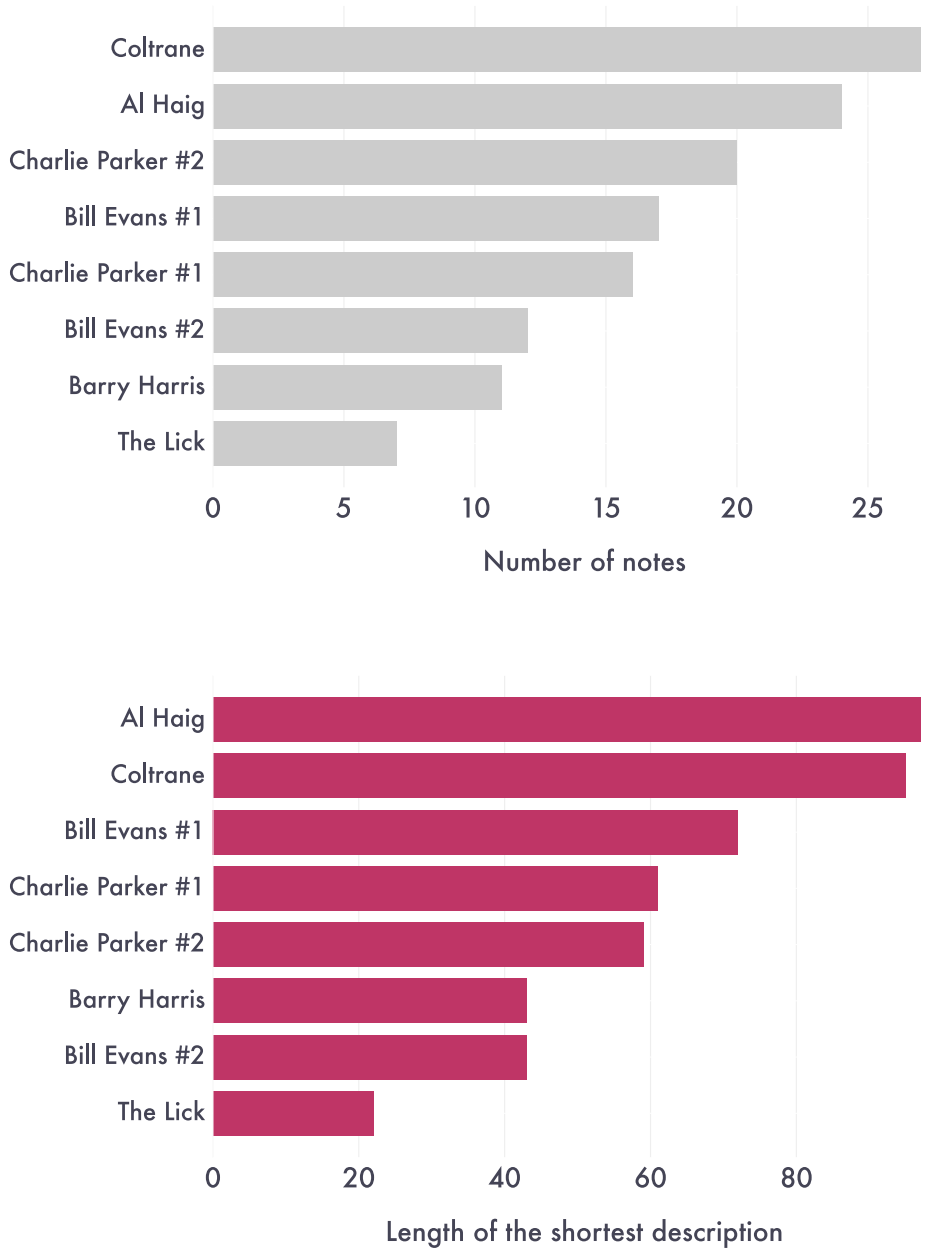


FIGURE 5.9: Note wise and description wise lengths of jazz licks

Chapter 6

Conclusion

We presented a simple model of perception and the possible way of distinguishing between objects based on their subjective beauty. However, this approach remains way too primitive and also desperate for experimental validation or some other form of justifying it. At heart, what led up to here was an elementary strive at postulating bare essential primitives that must play some part in determining the artistic value. Yet it's hard to delineate what actual little part of the appreciation of art takes appreciation of the form only and not thinking of the higher-order, which even further diminishes any formal attempt at addressing the question. Even with that in mind, concepts of compression and multiple descriptions of which we spoke at length must partake in whichever the actual answer is.

Most computer users nowadays are not aware of how rudimental compression algorithms are in the inner workings of computers nor that they are constantly using a large pool of them. We draw an analogy further: we're likewise unaware of how much art influences our thinking and how important it is in shaping our vision of the world. We view art as the storage or sharing of compressed sensations or compression algorithms. Further, we view any artwork as a puzzle piece, solving (making sense of) which acquires a procedure that is helpful in solving similar puzzles. Solving an unseen puzzle is hard while solving a "seen" one is trivial. Solving a puzzle set all at once (a process sure strange enough to be even considered) is much more diffucult than solving each at a time and even more difficult than using solutions already made by someone, which is why it's easier to view one's life in terms of existing art and not be an artist oneself.

As far as the comparison between objects goes, we conclude that for a thing having a short description is obviously good, but enabling an observer to enrich his language with powerful abstractions, such that a lot of other things will become far easier to comprehend, is so much better. The thesis of this work is that to define goodness one only needs the notion of representation, since by virtue of making sense of a good thing, that is by searching for short representations of it, one also acquires abstractions that are helpful at representing things in general.

Appendix A

Code is available at ogoremeni/revel written using Python 3.10. Implementation of e-graphs is inspired by (Zucker, 2021) [Julia], by Metatheory.jl (Cheli, 2021) [Julia] and by the egg library (Willsey et al., 2021) [Rust, pseudocode]. Implementation of λ-calculus with de-Bruijn index is inspired by (Tsú-thuàn, 2020) [Racket]. Sheet music visualization was done with the help of Musescore software[1] and music21 library (Cuthbert and Ariza, 2010).

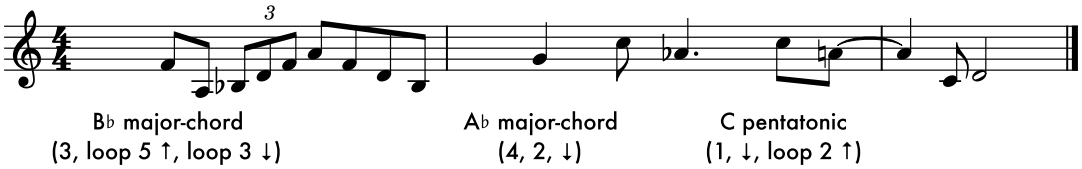
[1] https://musescore.org/en

Appendix B

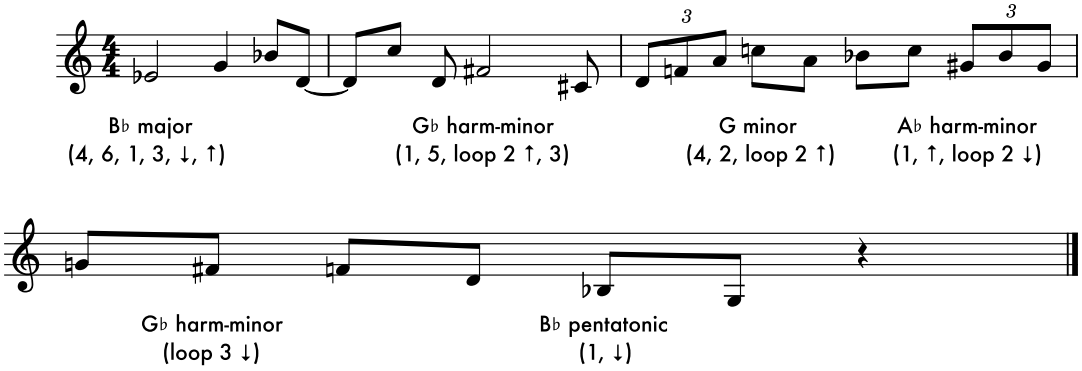
We describe eight jazz licks: each referred to after a jazz pianist who came up with it, except for "The Lick", which is its own kind. Here they are listed in the order as in the comparison from (Figure 5.8). Transcriptions used are from various sources.[1, 2, 3, 4]

[1] https://youtu.be/i5kR8Sn09CI
[2] https://youtu.be/JfnExW0iV_o
[3] https://musescore.com/user/34106726/scores/6515974
[4] https://musescore.com/user/3142241/scores/2298111

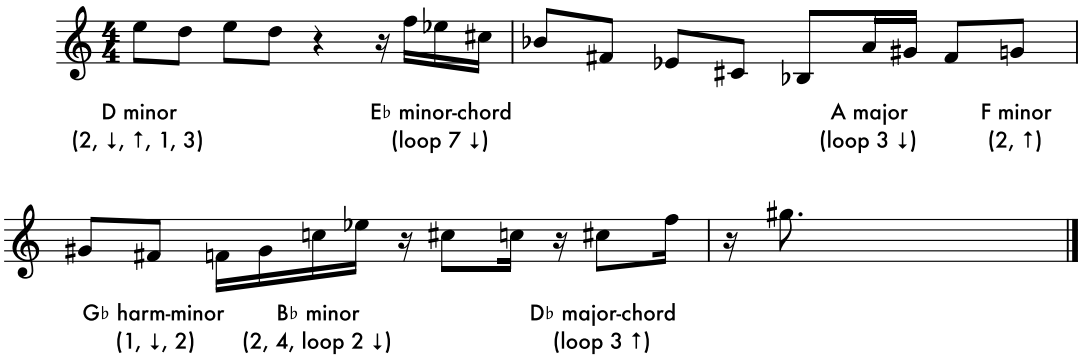
Charlie Parker #1



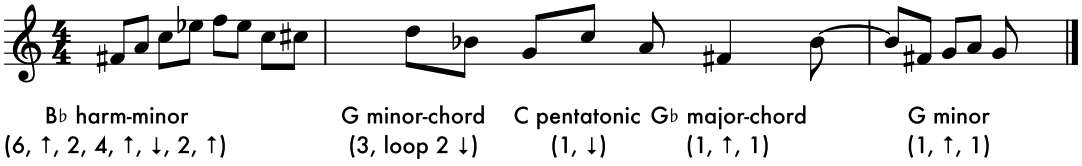
Al Haig



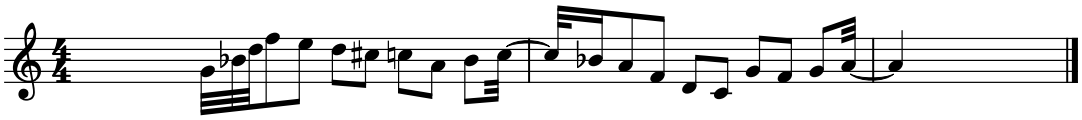
Coltrane



Bill Evans #1



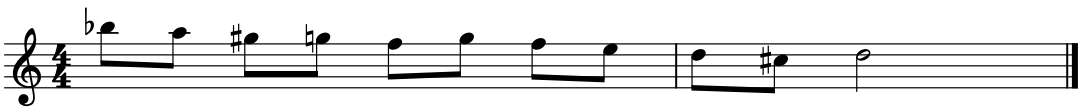
Charlie Parker #2



F bebop-major
(2, 4, 7, 1, loop 4 ↓, 3, loop 2 ↑, loop 2 ↓)

F pentatonic
(loop 3 ↓, 2, ↓, loop 2 ↑)

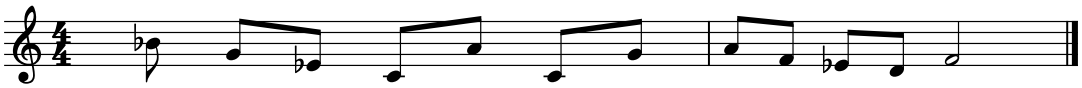
Barry Harris



B♭ chromatic
(loop 4 ↓)

F bebop-major
(1, ↑, loop 4 ↓, ↑)

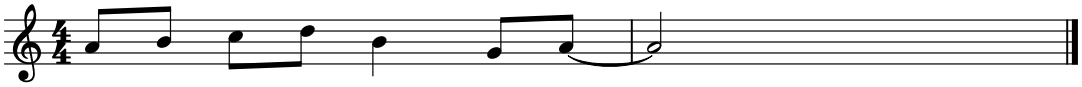
Bill Evans #2



C minor-chord
(4, loop 3 ↓)

F mixolydian
(3, 5, 2, ↑, 1, loop 2 ↓, 1)

The Lick



A minor
(loop 4 ↑, 2)

G minor
(1, ↑)

Bibliography

Bechtel, William (1991). "Connectionism and the philosophy of mind: an overview". In: *Connectionism and the Philosophy of Mind*, pp. 30–59.

Chaitin, Gregory J (1966). "On the length of programs for computing finite binary sequences". In: *Journal of the ACM (JACM)* 13.4, pp. 547–569.

— (2004). "Meta math! the quest for omega". In: *arXiv preprint math/0404335*.

Cheli, Alessandro (2021). "Metatheory.jl: Fast and Elegant Algebraic Computation in Julia with Extensible Equality Saturation". In: *Journal of Open Source Software* 6.59, p. 3078. DOI: 10.21105/joss.03078. URL: https://doi.org/10.21105/joss.03078.

Cuthbert, Michael Scott and Christopher Ariza (2010). "music21: A toolkit for computer-aided musicology and symbolic music data". In: URL: http://web.mit.edu/music21/.

Detlefs, David, Greg Nelson, and James B Saxe (2005). "Simplify: a theorem prover for program checking". In: *Journal of the ACM (JACM)* 52.3, pp. 365–473.

Dhariwal, Prafulla et al. (2020). "Jukebox: A generative model for music". In: *arXiv preprint arXiv:2005.00341*. URL: https://arxiv.org/pdf/2005.00341.

Ellis, Kevin et al. (2021). "DreamCoder: Bootstrapping Inductive Program Synthesis with Wake-Sleep Library Learning". In: *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*. New York, NY, USA: Association for Computing Machinery, pp. 835850. ISBN: 9781450383912. URL: https://doi.org/10.1145/3453483.3454080.

Fodor, Jerry A (1975). *The language of thought*. Vol. 5. Harvard university press.

Freeman, Sophie, Martin Gibbs, and Bjørn Nansen (2022). "Dont mess with my algorithm: Exploring the relationship between listeners and automated curation and recommendation on music streaming services". In: *First Monday*. URL: https://firstmonday.org/ojs/index.php/fm/article/download/11783/10589.

Grunwald, Peter (2004). "A tutorial introduction to the minimum description length principle". In: *arXiv preprint math/0406077*.

Gwern (2020a). "GPT-3 Creative Fiction". In: *gwern.net*. URL: https://www.gwern.net/GPT-3.

— (2020b). "The Scaling Hypothesis". In: *gwern.net*. URL: https://www.gwern.net/Scaling-hypothesis.

Hegel, Georg Wilhelm Friedrich (1874). *The logic of Hegel*. Clarendon Press.

Hindley, J Roger and Jonathan P Seldin (2008). *Lambda-calculus and Combinators, an Introduction*. Vol. 2. Cambridge University Press Cambridge.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.676.4320&rep=rep1&type=pdf.

Hutter, Marcus (2006). "Human knowledge compression prize". In: *hutter1.net*. URL: http://prize.hutter1.net/.

— (2009). "Open Problems in Universal Induction & Intelligence". In: *Algorithms* 3.2, pp. 879–906. ISSN: 1999-4893. DOI: 10.3390/a2030879. URL: http://arxiv.org/abs/0907.0746.

Kaelbling, Leslie Pack, Michael L Littman, and Andrew W Moore (1996). "Reinforcement learning: A survey". In: *Journal of artificial intelligence research* 4, pp. 237–285. URL: https://www.jair.org/index.php/jair/article/download/10166/24110/.

Kolmogorov, Andrei N (1965). "Three approaches to the quantitative definition of information". In: *Problems of information transmission* 1.1, pp. 1–7.

Kwak, Yuna and Clayton E. Curtis (2022). "Unveiling the abstract format of mnemonic representations". In: *Neuron*. DOI: 10.1016/j.neuron.2022.03.016. URL: https://doi.org/10.1016%2Fj.neuron.2022.03.016.

Morales, Lucas Eduardo (2018). "On the representation and learning of concepts: programs, types, and bayes". PhD thesis. Massachusetts Institute of Technology.

Nandi, Chandrakana et al. (2021). "Rewrite Rule Inference Using Equality Saturation". In: *Proc. ACM Program. Lang.* 5.OOPSLA. DOI: 10.1145/3485496. URL: https://doi.org/10.1145/3485496.

Nelson, Charles Gregory (1980). *Techniques for program verification*. Stanford University.

Ramesh, Aditya et al. (2022). "Hierarchical text-conditional image generation with clip latents". In: *arXiv preprint arXiv:2204.06125*. URL: https://arxiv.org/abs/2204.06125.

Schmidhuber, Jürgen (2008). "Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes". In: *Workshop on anticipatory behavior in adaptive learning systems*. Springer, pp. 48–76.

— (2010). "Formal theory of creativity, fun, and intrinsic motivation (1990–2010)". In: *IEEE transactions on autonomous mental development* 2.3, pp. 230–247.

— (2012). "A formal theory of creativity to model the creation of art". In: *Computers and creativity*. Springer, pp. 323–337.

Solomonoff, Ray J (1964). "A formal theory of inductive inference. Part I". In: *Information and control* 7.1, pp. 1–22.

Tsú-thuàn, Lîm (2020). "De Bruijn index: why and how". In: *dannypsnl.github.io*. URL: https://dannypsnl.github.io/blog/2020/05/16/cs/de-bruijn-index/.

Willsey, Max et al. (2021). "Egg: Fast and Extensible Equality Saturation". In: *Proc. ACM Program. Lang.* 5.POPL. DOI: 10.1145/3434304. URL: https://doi.org/10.1145/3434304.

Wright, Anthony A et al. (2000). "Music perception and octave generalization in rhesus monkeys." In: *Journal of Experimental Psychology: General* 129.3, p. 291.

Zucker, Philip (2021). "A Simplified E-graph Implementation". In: *philipzucker.com*. URL: https://www.philipzucker.com/a-simplified-egraph/.