석 사 학 위 논 문
Master's Thesis

하 적분의 계산

Computing Haar Integrals

2019

선 동 성 (宣 東 聲 Seon, Dongseong)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

석 사 학 위 논 문

하 적분의 계산

2019

선 동 성

한 국 과 학 기 술 원

전산학부

하 적분의 계산

선 동 성

위 논문은 한국과학기술원 석사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2019년 6월 12일

심사위원장	Martin Ziegler	(인)
심 사 위 원	Svetlana Selivanova	(인)
심 사 위 원	양홍석	(인)

Computing Haar Integrals

Dongseong Seon

Advisor: Martin Ziegler

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Daejeon, Korea
June 12, 2019

Approved by

Martin Ziegler
Professor of School of Computing

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

MCS 20174306 선동성. 하 적분의 계산. 전산학부 . 2019년. 18+iii 쪽. 지도교수: 지글러 마틴. (영문 논문)
Dongseong Seon. Computing Haar Integrals. School of Computing . 2019. 18+iii pages. Advisor: Martin Ziegler. (Text in English)

초 록

적분을 보렐 확률 측도 (μ) 들에 대해 어떤 함수를 전체 공간 위의 보렐 확률 측도에 대한 적분값에 대응시키는 범함수 ($f \mapsto \int_X f d\mu$) 로 정의하자. 그러면 [0,1] 위의 리만적분은 [0,1] 위의 실직선의 보렐 확률 측도에 대한 적분이 된다. 이 [0,1] 위의 적분은 계산가능함이 알려져 있으며, 계산 복잡도마저 분석되어있다. 이후의 연구에서, 더 일반적으로 특정 공간들에서는 적분의 계산가능성과 대응하는 측도의 계산가능성이 동치임이 알려졌다. 따라서 공간들에서 실제로 어떤 자연스러운 적분들을 계산하려면, 해당하는 자연스러운 측도들을 계산해야한다. 하 측도는 자연스러운 측도라고 볼 수 있는데, 이는 하 정리가 하 측도는 평행변환불변인 정칙 확률 측도이며, 모든 옹골찬 위상군에 유일하게 존재함을 의미하기 때문이다. 따라서 자연스러운 적분들을 계산하기 위해 우리는 하 측도들을 자연스러운 측도들로 고려하고 이 측도들이 계산가능함을 증명한다. 하 측도의 계산가능성을 증명하려는 또 다른 동기는, 하 측도의 계산가능성을 증명하는 것이 하 정리의 계산적인 형태를 증명하는 것으로도 해석될 수 있기 때문이다. 이 논문에서 우리는 하 적분들 (하 측도들에 대한 적분들) 이 하 정리의 가정의 계산적인 형태를 가정하면 계산가능함을 보인다. 또한, 우리는 가장 중요한 옹골찬 위상군이라 할 수 있는 3차원 특수직교군의 하 적분의 계산가능성을 증명하고, 계산 복잡도를 분석하며, 알고리즘을 구현한다.

핵 심 낱 말 정확한 실수 연산, 계산해석학, 하 측도, 하 적분, 하 정리, 3차원 특수직교군, 옹골찬 거리군

Abstract

Define *integral*: $\mathcal{C}(X) \to \mathbb{R}$ to be a functional $f \mapsto \int_X f d\mu$ with a Borel probability measure μ. Then, the usual Riemann integral on [0,1] is an integral on [0,1] with the Borel probability measure on the real line. This integral on [0,1] is known to be computable, and its complexity was analyzed. After that, more generally it is known that, on some spaces, an integral is computable if and only if its corresponding measure is computable. Consequently, to actually compute natural integrals on spaces, one should compute natural measures. Arguably, Haar measures can be seen as natural measures. This is because Haar's theorem states that for any compact topological group, there exists a unique Haar probability measure which is translation-invariant and regular. Thus, to compute natural integrals, we consider Haar measures to be natural and prove that these measures are computable. Another motivation to prove computability of Haar measures is that proving that Haar measures are computable can be interpreted as proving a computable version of Haar's theorem. In this paper, we prove that Haar integrals (integrals with their Haar measures) are computable under computable version of assumptions of Haar's theorem. Moreover, we prove computability, analyze complexity, and implement the Haar integral on arguably the most important compact topological group $\mathcal{SO}(3)$.

Keywords Exact Real Computation, Computable Analysis, Haar measure, Haar integral, Haar's theorem, 3D Rotation group, compact metric group

Contents

List of Tables

List of Figures

Chapter 1. Introduction

1.1 Motivation

Define *integral*: $\mathcal{C}(X) \to \mathbb{R}$ to be the functional $f \mapsto \int_X f d\mu$ with a Borel probability measure μ. Then, the usual Riemann integral on [0,1] is an integral on [0,1] with the Borel probability measure on the real line. In [6, Theorem 5.32], this integral is known to be computable, and in $\#P_1$.
Since this integral is already well-known, the next natural question would be the generalization of it. Not surprisingly, there are several relevant studies. In particular, [12] provides a way to represent measures on some topological spaces and to compute integrals with corresponding measures. In later works, computability of integrals is even claimed to be an elementary result [3, Proposition 7]. These works say that an integral is computable if and only if its corresponding measure is computable.
Consequently, on an arbitrary space, to actually compute the natural integral on the space, we should compute the natural measure. For natural measures of spaces, Haar measures are arguably the most suitable. Recall that Haar's theorem states that on any compact topological group, there uniquely exists a Borel probability measure (call it a Haar measure) which is translation-invariant and regular. Thus, to compute natural integrals, we consider Haar measures to be natural and prove that these measures are computable.
Since computing Haar integrals is computably equivalent to computing Haar measures, our result can be also interpreted as a proof of a computable version of Haar's theorem. Studying a computable version of a pure mathematical theorem is not a new topic. There are such studies for Brouwer's Fixed Point Theorem [8, 1] and Peano's Theorem [11].

1.2 Result

The result is a co-work of Arno Pauly, Dongseong Seon (the author), and Martin Ziegler. We proved computability of Haar integrals on a large subset of computable metric groups. We have two proofs: a proof by giving an explicit algorithm (explicit proof), and a proof by synthetic approach (synthetic proof). However, since only explicit proof is done by the author under supervision of Martin Ziegler, synthetic proof will be only briefly discussed in chapter 2. On the other hand, explicit proof is fully explained in chapter 3.
After proving computability, the next question was complexity of the problem. However, on such broad domain of functionals, *standard* complexity definitions are not established yet. For instance, even the complexity definitions of functionals on functions on Euclidean spaces are not so obvious, so that the definition of polynomial-time computable involves modulus of continuity (see [6, Definition 2.37]). Thus, we proved computability, analyze complexity, implement, and experiment an algorithm computing the Haar integral on $\mathcal{SO}(3)$, instead of doing these on general groups. This is because complexity definitions of $\mathcal{SO}(3)$ are unproblematic, since it is a subset of a Euclidean space, $\mathbb{R}^{3\times3}$. Also, $\mathcal{SO}(3)$ is the simplest and arguably the most important among such spaces other than $\mathcal{U}(1)$. Our algorithm on $\mathcal{SO}(3)$ is different from our general algorithm, since we adapted a standard method of calculating Haar integral on $\mathcal{SO}(3)$ in pure mathematics. Since this is done by the author and Martin Ziegler, it is fully explained in chapter 4.

To elaborate, let us state the computability theorem we proved. This theorem is proved by giving an explicit algorithm:

Theorem 1.2.1. *Let $(X, d, \circ)$ be a computable metric space and a compact topological group. Suppose that the metric d is bi-invariant (That is, $d(a \circ c, b \circ c) = d(a, b) = d(c \circ a, c \circ b)$) and the sizes of maximal packings κ_X are computable. Then the Haar measure is computable.*

Seemingly, Theorem 1.2.1 partially fails to be a computable version of Haar's theorem. The failing point is that it requires bi-invariance of the metric and computability of sizes of maximal packings. Without some work, these are not a computable version of hypothesis of Haar's theorem. Instead, more natural computable version of Haar's theorem is the following:

Theorem 1.2.2. *Let $(X, d, \circ)$ be a computably compact computable metric space. Suppose that the group operation $\circ$ is computable. Then the Haar measure is computable (note that $\circ$ being computable is a computable version of X being a topological group).*

In fact, Arno Pauly proved the equivalence of Theorem 1.2.1 and Theorem 1.2.2, and the proof is presented in [10]. This fact implies that our explicit algorithm is computing Haar measures with natural assumptions (that is, assumptions of Theorem 1.2.2).
Arno Pauly proved Theorem 1.2.2 using the synthetic approach, and our two proofs (explicit proof and synthetic proof) have their own pros and cons. In short, explicit proof is easy to implement, and synthetic proof is elegant but hard to implement and includes unbounded search. Especially, because of this unbounded search, synthetic approach is not designed to aim for complexity in its nature, whereas the explicit proof has possibilities. In this paper, we only present the explicit proof. For the synthetic proof, the sketch is given in chapter 2. For more details of the synthetic proof, see [10]. From now on, we will informally explain why each assumption of Theorem 1.2.1 corresponds to an assumption of Theorem 1.2.2. First, one of these assumptions is knowing the *sizes of maximal packings*. This is a notion very similar to the *separation bound* from [15, Definition 6.2] and the *capacity* from [5, Definition 12]. In these papers, separation bound and capacity are known to quantitatively capture the compactness of the space, in being closely related to a *modulus of total boundedness* [7, Def 17.106]. Consequently, the sizes of maximal packings being computable can be interpreted as the space being computably compact in Theorem 1.2.2.
Second is bi-invariance of the metric. Only this property makes a dependency with the group structure of the space. If the assumption does not include this property, then there is no algorithmic way to compute anything about the group structure under the assumption. In this informal sense, bi-invariance of the metric combined with the space being a computable metric space corresponds to the group operation being computable.
The next is a complexity result on $\mathcal{SO}(3)$:

Theorem 1.2.3. *If $f \in \mathcal{C}(\mathcal{SO}(3))$ is polynomial-time computable, then $\int f \, d\mu \in \mathbb{R}$ is $\#P_1$-computable. If for all $f \in \mathcal{C}(\mathcal{SO}(3)), (f$ is polynomial-time computable implies $\int f \, d\mu \in \mathbb{R}$ is polynomial-time computable), then $(FP_1 = \#P_1)$.*

We prove this theorem by giving an algorithm. The algorithm on $\mathcal{SO}(3)$ is implemented and experimented with. With the experiment result, we claim that our algorithm on $\mathcal{SO}(3)$ asymptotically runs in exponential time.

1.3 Background

In this section, we give brief explanations for the background. The followings are explained: The theory of computation of reals, computable versions of definitions in mathematical analysis for chapter 2, unary complexity classes $FP_1, \#P_1$ and consequences of $FP_1 = \#P_1$ for chapter 4.
On Turing machines, natural numbers are represented as finite binary strings by the binary representation. Rational numbers can be represented similarly, since they can be represented as pairs of natural numbers. However, reals cannot be represented as finite strings, since they are uncountably many, whereas finite strings are only countably many. Thus, the representation of reals is defined using infinite strings:

Definition 1.3.1. The representation of reals $\delta :\subseteq \Sigma^\omega \to \mathbb{R}$ is a partial surjective function that maps an encoding of $d_1 d_2 \ldots$ to r, where $d_i \in \mathbb{Q}$ and $|r - d_i| \leq 2^{-i}$. If $\delta(s) = r$, s is said to represent r.

With this definition, we can define computability notions involving reals.

Definition 1.3.2. A real number r is called computable iff there exists an algorithm computing a representation of r. A real function $f : \mathbb{R} \to \mathbb{R}$ is called computable iff there exists an algorithm computing a representation of $f(r)$ from a representation of r.

Here, the algorithm is assumed to input and output infinite strings. The expansion from finite strings to infinite strings is straightforward. For the input, we assume an infinite length input tape. For the output, the algorithm recieves an additional index input n and outputs the n-th word of the infinite output string. In other words,

Definition 1.3.3. A real number r is called computable iff there exists an algorithm that recieves n and outputs $d_n \in \mathbb{Q}$ where $|r - d_n| \leq 2^{-n}$.

Intuitively speaking, a representation of a real is its sequence of approximations, and the real is computable iff the sequence is computable. For example, if $a \in \mathbb{R}$ and $b \in \mathbb{R}$ are computable, then $a + b$ is also computable because there is an algorithm that generates sequences of approximations of a, b, reads index n, and outputs an n-th approximation of $a + b$ by summing $(n + 1)$-th approximations of a and b. However, even if for all n, $r_n \in \mathbb{R}$ is computable, $\lim_{n\to\infty} r_n$ could be not computable, because the algorithm can only inspect finite amount of the real sequence $\{r_n\}_{n=1}^{\infty}$, and with that information the algorithm can never be sure how close the approximation to the limit is.
Next is the definition of computable metric space:

Definition 1.3.4. (X, d, A, α) is called a computable metric space if (X, d) is a metric space, $A \subseteq X$ is a dense subset, $\alpha :\subseteq \Sigma^* \to A$ is surjective, $\mathrm{dom}(\alpha)$ is recursively enumerable, and $(a, b) \mapsto d(a, b) : A \times A \to \mathbb{R}$ is computable (α is the representation of A). The representation of X, $\delta : \Sigma^\omega \to \mathbb{R}$, is a function that maps an encoding of $a_1 a_2 \ldots$ to x, where $\forall i\; a_i \in A$ and $d(x, a_i) \leq 2^{-i}$.

Intuitively speaking, a space is a computable metric space iff there is a dense subset which is recursively enumerable, and the distance between two points is computable. Note that $(\mathbb{R}, d_\mathbb{R}, \mathbb{Q}, \text{encoding of } \mathbb{Q})$ is a computable metric space. That is, the dense subset A works as $\mathbb{Q}$ in $\mathbb{R}$. The representation is a rapidly converging sequence of elements of A.
To define computability of functionals from $\mathcal{C}(X)$ to $\mathbb{R}$, we need a representation of real continuous functions $\mathcal{C}(X)$. Intuitively, an algorithm computing a real function can be a representation of the function. However, since not every continuous functions are computable, this intuition cannot work. Thus, to

be precise, we should define the representation of $f \in \mathcal{C}(X)$ as the enumeration of representation of $f(A) \in \mathbb{R}^{\mathbb{N}}$. However, in computability sense, recieving an enumeration of $f(A)$ is equivalent to having a subroutine that returns $f(a)$ when we input $a \in A$. Thus, if X is a computable metric space, one can simply think that we have a (possibly uncomputable) subroutine computing the real function. To sum up, if X is a computable metric space, a functional $F : \mathcal{C}(X) \to \mathbb{R}$ is computable iff there is an algorithm that outputs $F(f) \in \mathbb{R}$ by having a subroutine that computes $f \in \mathcal{C}(X)$.
Recall that FP and $\#P$ are complexity classes of function problems. FP is, as you can guess, a class of polynomial-time function problems. However, $\#P$ is more tricky:

Definition 1.3.5. A function problem $f : \mathbb{N} \to \mathbb{N}$ is in $\#P$ iff there exists an non-deterministic Turing machine s.t. for input $n \in \mathbb{N}$, the number of accepting path of the machine is $f(n)$.

The class $\#P$ is also called the class of counting problems. FP_1 and $\#P_1$ are similar to FP and $\#P$. The difference is, in the former the problem is assumed to recieve an input n in unary, whereas in the latter the input is in binary. One similar notion is pseudo-polynomial time, which means the algorithm runs in polynomial of the numeric value of the input. Another relevant keyword that might help your search is *tally sets*, which are subsets of unary encodings $\{0^*\}$.
Our complexity result depends on whether $FP_1 = \#P_1$. Since function problems are harder than decision problems, it is easy to see that $FP_1 = \#P_1 \Rightarrow P_1 = NP_1$. The fact that $P_1 = NP_1 \Leftrightarrow EXP = NEXP$ is well-known in textbooks such as [6, Proposition 1.33]. Also, it is known that $P = NP \Rightarrow EXP = NEXP$, but the other way around is not known. Of course, whether $EXP = NEXP$ is not known. To sum up, $FP_1 = \#P_1 \Rightarrow EXP = NEXP$.
Here, we only briefly explained the concepts. For details, see the following references: For the computable metric space, see [14, Definition 8.1.2]. For the complexity/computability of reals/real functions/functionals, see [6, Section 2]. For unary complexity classes such as $\#P_1$ and FP_1, see [6, Section 5]. For the synthetic approach, see [9]. For computable versions of definitions in measure theory, see [3].

Chapter 2. The Synthetic Approach

In this chapter, we give a sketch of the synthetic proof that proves theorem 1.2.2. The synthetic proof is from [10], which is our another presentation of the results. To do that, we first briefly explain what synthetic approach means and how it differs from an explicit approach. Moreover, we discuss about its benefits and limitations.

By *synthetic approach*, we mean the way of proving theorems or establishing definitions presented in [9]. However, majority of this chapter is devoted to explain the way of proving theorems, instead of establishing definitions. Our synthetic proof follows the synthetic approach. Also, the definitions in Theorem 1.2.2 are from the synthetic approach.

2.1 Overview on the synthetic approach

The synthetic approach differs from classical ways of proving computability. The difference is that we prove computability results by composing computable operations on represented spaces (mathematical spaces with their representations), rather than directly using representations. This way of using represented space is similar to using abstract data type (ADT) in programming. There is an implementation of ADT, but programmers using the ADT do not directly work with its implementation. Rather, programmers only use exposed operations on the ADT. For example, the space of open sets of X, denoted as $\mathcal{O}(X)$, can be studied only with its permitted operations (see [9, Proposition 6]). Also, the representation of $\mathcal{O}(X)$ is not directly used. Instead, the standard way to generate representations of continuous function spaces (denoted as $\mathcal{C}(X, Y)$) and Sierpinski space (denoted as $\mathbb{S}$) is given once, and the representation of $\mathcal{O}(X)$ is assumed to be made by those ways, from noting that $\mathcal{O}(X) = \mathcal{C}(X, \mathbb{S})$.

Permitted operations of $\mathcal{O}(X)$ is essentially that of semi-decidable sets. That is, the membership to U is semi-decidable with the representation of $U \in \mathcal{O}(X)$. Similarly, the membership to an element of $\mathcal{A}(X)$ (closed sets) is co-semi-decidable with its representation. Thus, for $S \in \mathcal{A}(X)$, the membership to S is co-semi-decidable iff the representation of S is computable. Representation of S being computable is also said that S is computably closed. In this way, we say a predicate on X is computably open/closed iff the predicate viewed as a set is computably open/closed.

Permitted operations of $\mathcal{K}(X)$ (compact sets) is similar to that of compact sets in mathematics. That is, a finite subcover is semi-decidable from any cover with its representation. Consequently, whether an open set is the full space is semi-dedicable on computably compact spaces. This means a universal quantification of a computably open predicate on a computably compact space is also computably open. Similarly, by noting the dual relationships, an existential quantification of a computably closed predicate on a computably compact space is computably closed. $\mathcal{V}(X)$ (called overt sets) is a dual notion of compactness, so a universal quantification of a computably closed predicate is computably closed on a computably overt spaces.

Since the synthetic approach ignores actual representation, it gives more clear and elegant proof of computability of the theorem. Moreover, using category theory, the approach gives more natural computability definitions of mathematical definitions. On the other hand, the synthetic approach only gives implicit algorithms opposed to the explicit, imperative algorithms that can be obtained in classical ways.

Moreover, since definitions of the synthetic approach such as computably compact involves unbounded search, it cannot aim for complexity. The complexity issue of the synthetic approach is not surprising, since to establish right complexity definitions, more cares are needed than to establish computability definitions such as the case of reals in [6, Section 2.2]

2.2 Sketch of the synthetic proof

In the synthetic proof, the strategy is the following:

1. Prove that the set of Haar measures is computably closed.
2. Prove that the set of probability measures including the set of Haar measures is computably compact. Combine item 1 and 2 to get the fact that the set of Haar measures is computably compact.
3. Use Haar's theorem to prove the set of Haar measures is a singleton set
4. Use admissibility to compute an element from its computably compact singleton set

This actually follows the general strategy to use admissibility outlined in [9] as "How to use admissibility". Relatively the hardest part following this strategy is to prove that the set of Haar measures is computably closed. This is because it is not straightforward to prove $\forall U \in \mathcal{O}(X).\forall x \in X.\mu(U) = \mu(xU)$ is computably closed, because representations of measures only output lower reals, so we cannot even *refute* equality. To remedy this issue, we instead consider a predicate $\forall (U, V) \in DPO(X).\forall x \in X.\mu(U) + \mu(xV) \leq 1$ where $DPO(X) := \{(U, V) \in \mathcal{O}(X) \times \mathcal{O}(X) \mid U \cap V = \emptyset\}$, and prove that the predicate is equivalent to the first predicate. In this case, the fact that $\mu(U) + \mu(xV) \leq 1$ is computably closed is straightforward from noting that representations of measures output lower reals, and computable overtness of $DPO(X)$ and X preserves the predicate being computably closed under universal quantifications.

Chapter 3. Explicit Algorithm for General Haar Integrals

Recall the goal of this chapter:

Theorem 3.0.1. *Let $(X, d, \circ)$ be a computable metric space and a compact topological group. Suppose that the metric d is bi-invariant (That is, $d(a \circ c, b \circ c) = d(a, b) = d(c \circ a, c \circ b)$) and the sizes of maximal packings κ_X is computable. Then the Haar measure is computable.*

Here is the definition of maximal packings:

Definition 3.0.2. For any compact metric space (X, d) and its subset $U \subseteq X$,

1. $T \subseteq U$ is called an n-packing of U if $\forall x, y \in T.(x \neq y) \rightarrow (d(x, y) > 2^{-n})$.

2. $\kappa_U : \mathbb{N} \rightarrow \mathbb{N}$ is called the sizes of maximal packings of U if

$$\kappa_U(n) := \max_{T \textit{ is an } n\text{-packing of } U} |T|$$

3. T is called a maximal n-packing of U if it is a n-packing of U and $|T| = \kappa_U(n)$.

4. $\{T_m\}_{m=1}^{\infty}$ is called a sequence of maximal packings if each T_m is a maximal m-packing.

If $U = X$, the term 'of U' is omitted.

In this chapter, we give the explicit algorithm that computes Haar integrals. The key lemma is lemma 3.1.5. With this lemma, by computing μ_{T_n} (lemma 3.2.4), the algorithm is able to compute Haar measures for particular sets. Thus, by partitioning the space with those particular sets that their measures are computable (**findNicePartition**), the algorithm can compute Haar integrals (section 3.3).

3.1 Mathematical lemma to compute Haar measures

To compute measures, the idea is that Haar measures and maximal packings are, in a sense, both evenly distributed in their spaces. This similarity lets the counting measure of a maximal packing to approximate the corresponding Haar measure. Intuitively speaking, Haar measures are evenly distributed because of its translation-invariance. The translations (applying group operation with various elements) only change the locations and preserve the shapes and sizes of sets, because of the bi-invariance of the metric. Thus Haar measures are evenly distributed, since they are independent of the locations of sets. Similar to that Haar measures are evenly distributed because of left-invariance, maximal packings are evenly distributed because of maximality. Intuitively speaking, if there exists a region in the space that points of a maximal packing are not located enough compare to other congruent regions, there is a contradiction since another packing that has one more element can be made by adding one more point at that region. The below lemma formalizes this informal argument about maximal packings both in its statement and proof. In its statement $|T_n \cap U|$ and $|T_n \cap xU|$ are both bounded in some interval. In its proof it heavily relies on maximality.

Definition 3.1.1. For a metric space (X, d) and its subset $U \subseteq X$, the outer generalized closed ball is denoted as $\overline{B}_r(U)$ and defined as $\bigcup_{x \in U} \overline{B}_r(x)$. Similarly, the inner generalized closed ball is denoted as $\overline{B}_{-r}(U)$ and defined as $B_r(U^c)^c = \{x \in U : B_r(x) \subseteq U\}$.

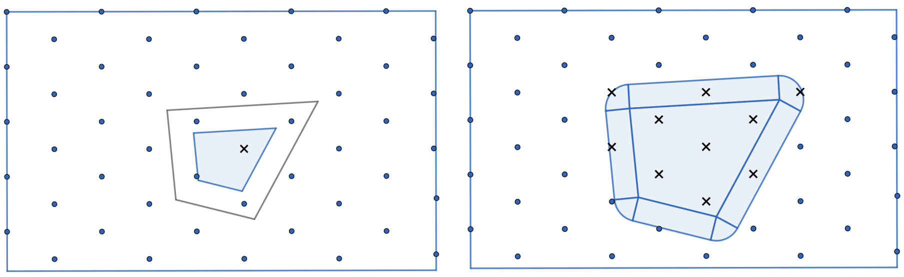

Figure 3.1: Demonstration of lemma 3.1.5. A blue rectangle represents the space. Blue points represent the maximal n-packing. A black shape represents U. Blue colored shapes represent outer and inner generalized balls. Counting cross-marked points and dividing it by the number of (any) points gives μ_{T_n}.

Lemma 3.1.2. *Let $(X, d, \circ)$ be a compact topological group with a bi-invariant metric d. If T_n is a maximal n-packing, then for any $x \in X$ and $U \subseteq X$:*

$$\kappa_{\overline{B}_{-2^{-n}}(U)}(n) \leq |T_n \cap xU| \leq \kappa_{\overline{B}_{2^{-n}}(U)}(n)$$

Proof. $|T_n \cap xU| \leq \kappa_{xU}(n)$ since $T_n \cap xU$ is an n-packing of xU. Because of bi-invariance, $\kappa_U(n) = \kappa_{xU}(n)$. So $|T_n \cap xU| \leq \kappa_U(n) \leq \kappa_{\overline{B}_{2^{-n}}(U)}(n)$ since $U \subseteq \overline{B}_{2^{-n}}(U)$.
Let S_n to be a maximal n-packing of $\overline{B}_{-2^{-n}}(xU)$. Then $S_n \cup (T_n \cap (xU)^c)$ is an n-packing for the whole space X. Since T_n is maximal, $|S_n| + |T_n \cap (xU)^c| = |S_n \cup (T_n \cap (xU)^c)| \leq |T_n|$. Thus $|T_n \cap xU| \geq |S_n| = \kappa_{\overline{B}_{-2^{-n}}(U)}(n) = \kappa_{\overline{B}_{-2^{-n}}(xU)}(n)$.

$\square$

This makes the ratio of points of maximal packings in a set to give an approximation of the Haar measure of the set. The below lemma formalizes this property.

Reminder 3.1.3. δ_p is called a Dirac measure if $\delta_p(S) = \begin{cases} 0 & \text{if } p \notin S \\ 1 & \text{if } p \in S \end{cases}$

Definition 3.1.4. $\mu_T := \frac{1}{|T|} \sum_{p \in T} \delta_p$ where δ_p denotes a Dirac measure.

Lemma 3.1.5. *Let $(X, d, \circ)$ be a compact topological group with a bi-invariant metric d. If T_n is a maximal n-packing and μ is the Haar measure of X, then for any $U \subseteq X$:*

$$\mu(\overline{B}_{-2^{-n+2}}(U)) \leq \mu_{T_n}(\overline{B}_{-2^{-n+1}}(U)) \leq \mu(U) \leq \mu_{T_n}(\overline{B}_{2^{-n+1}}(U)) \leq \mu(\overline{B}_{2^{-n+2}}(U))$$

Proof. Let $T_n = \{p_1, \ldots, p_N\}$ (note that $|T_n| = N$). Then $\mu(U)N = \sum_{i=1}^{N} \int_X \chi_{p_i U} d\mu = \int_X \sum_{i=1}^{N} \chi_{p_i U} d\mu$. Let $f := \sum_{i=1}^{N} \chi_{p_i U}$ then $f(x) = |\{i : x \in p_i U\}| = |\{i : p_i \in xU^{-1}\}| = |T_n \cap xU^{-1}|$. By lemma 3.1.2, $f(x) \leq \kappa_{\overline{B}_{2^{-n}}(U^{-1})} \leq |T_n \cap \overline{B}_{2^{-n+1}}(U^{-1})|$. Dividing sides of $\mu(U)N = \int_X f d\mu \leq |T_n \cap \overline{B}_{2^{-n+1}}(U^{-1})|$ with N gives $\mu(U) \leq \mu_{T_n}(\overline{B}_{2^{-n+1}}(U^{-1}))$. Additionally, $\mu(U) = \mu(U^{-1})$ because if we let $\lambda(U) := \mu(U^{-1})$ then λ is the Haar measure (note that Haar measures are both left and right invariant on compact topological groups), which should coincide with μ. This completes the proof of third inequality and the others can be similarly obtained.

$\square$

3.2 Algorithm to compute Haar measures of particular sets

The measure μ_T is in general not computable, but it can be approximated for closed sets that their distances with arbitrary points are computable (by the procedure **pseudoCount** below). Let us define this property of sets to be *computably located* (there is a similar notion called *Turing located* [4]):

Definition 3.2.1. A closed subset S of a computable metric space is called computably located if $p \mapsto d(p, S)$ is computable.

Computably located sets are sometimes called computably closed sets, but being computably located is different from being a computable element of $\mathcal{A}(X)$.

Thus if $\overline{U}$ is computably located, in order to use lemma 3.1.5 to compute $\mu(U)$, it is sufficient to have $\lim_{r\to 0} \mu(\overline{B}_r(U)) = \mu(U)$. Note that $\lim_{r\to 0} \mu(\overline{B}_r(U)) = \mu(U) \Leftrightarrow \mu(\partial U) = 0$.

Definition 3.2.2. On a topological space (X, τ) with a measure μ, a measurable set U is called co-inner regular iff $\mu(U) = \sup\{\mu(G) \mid G \subseteq U, G \text{ is open and measurable}\}$. The number $r \in \mathbb{R}$ is called co-inner regular radius iff $\overline{B}_r(p)$ is co-inner regular.

This is well defined since bi-invariance of the metric implies $\forall p, q, r\ \mu(\overline{B}_r(p)) = \mu(\overline{B}_r(q))$.

Remark 3.2.3. Note that a closed set S is co-inner regular iff $\mu(S) = \sup\{\mu(G) \mid G \subseteq S, G \text{ is open and measurable}\} = \mu(S^\circ)$. Furthermore, $\mu(S) = \mu(S^\circ)$ iff $\mu(\partial S) = 0$.

Lemma 3.2.4. *Let $(X, d, \circ)$ be a compact topological group with a bi-invariant metric d. If U has a computably located, co-inner regular closure, then its measure is computable by the below procedure **computeMeasure**.*

```
Data: $\overline{U}$ is a co-inner regular, computably located set. $\{T_m\}_{m=1}^{\infty}$ is a sequence of maximal
      packings. $n$ is a target precision.
Result: A rational number $q$ s.t. $|q - \mu(S)| \leq 2^{-n}$.
error ← $\infty$;
$n \leftarrow 0$;
while error $> 2^{-n}$ do
    $r \leftarrow 2^{-n+1}$;
    interval ← (pseudoCount($\overline{B}_{-r}(\overline{U})$, $T_n$, $n$), pseudoCount($\overline{B}_r(\overline{U})$, $T_n$, $n+1$));
    error ← length(interval);
    $n \leftarrow n + 1$;
end
return any $p \in$ interval;
```

Procedure computeMeasure(U, $\{T_m\}_{m=1}^{\infty}$, n)

Proof. In **computeMeasure**, for each while loop, the interval is a subinterval of $[\mu_{T_n}(\overline{B}_{-2^{-n}}(\overline{U})), \mu_{T_n}(\overline{B}_{2^{-n}}(\overline{U}))]$ because of the postcondition of **pseudoCount**. This interval converges to $\mu(U)$ because of lemma 3.1.5 and that $\lim_{r\to 0} \mu(\overline{B}_r(\overline{U})) = \mu(\overline{U}) = \mu(U)$ (from $\overline{U}$ being co-inner regular). The postcondition of **pseudoCount** is satisfied, because for any $p \in T$, every $p \in S$ is counted and every $p \notin \overline{B}_{2^{-n}}(S)$ is not counted. □

```
Data: $S$ is a computably located closed set. $T$ is a set of points. $n$ is an error precision.
      dist($p, S, n$) computes distance between a point $p$ and a closed set $S$ with precision $n$.
Result: A rational $q$ where $\mu_T(S) \leq q \leq \mu_T(\overline{B}_{2^{-n}}(S))$
count $\leftarrow 0$;
foreach $p \in T$ do
    if dist($p$,$S$,$n+2$) $< 2^{-n-1}$ then
        count $\leftarrow$ count $+ 1$;
    end
end
return $\frac{\text{count}}{|T|}$ ;
```

Procedure pseudoCount(S, T, n)

Not every closures of sets are co-inner regular, but they are sufficiently many. In fact, co-inner regular radii can be effectively found to compute Haar measures in the form of Haar integral with **findCoInnerRegularRadius**.

3.3 Algorithm to compute Haar integrals

Explicit algorithm for theorem 1.2.1. The below procedure **computeIntegral** computes the Haar integral functional of X. The procedure can be a proof even though it recieves a sequence of maximal packings, not the sizes of maximal packings. This is because it is known and not hard to see that on computable metric spaces, there is an algorithm computing a sequence of maximal packings if the sizes of maximal packings is computable (fact 3.4.1). For correctness of the procedures, see section 3.4.

```
Data: $f : X \to \mathbb{R}$ is a real function. $\{T_m\}_{m=1}^{\infty}$ is a sequence of maximal packings. $n$ is a target
      precision.
Result: A rational number $q$ s.t. $|q - \int_X f d\mu| \leq 2^{-n}$
$m_f \leftarrow$ modulus($f$, $n+1$) ;                    // modulus is from [6, Definition 2.12]
$\{U_i\}_{i=1}^{N} \leftarrow$ findNicePartition($\{T_m\}_{m=1}^{\infty}$, $m_f$);
$M \leftarrow$ bound($|f|$);
foreach $U_i$ in $\{U_i\}_{i=1}^{N}$ do
    $p_i \leftarrow$ center($U_i$);
    $m_i \leftarrow$computeMeasure($U_i$, $\{T_m\}_{m=1}^{\infty}$, $n + 1 + i + \log M$);
end
return $\sum_{i=1}^{N} m_i f(p_i)$ ;
```

Procedure computeIntegral(f, $\{T_m\}_{m=1}^{\infty}$, n)

□

```
Data: $\{T_m\}_{m=1}^{\infty}$ is a sequence of maximal packings. $n$ is a target precision.
Result: A partition $P = \{U_i\}_{i=1}^{N}$ s.t. each $\overline{U_i}$ is a co-inner regular, computably located closed
        set and contained in a closed ball with radius $2^{-n}$. Representation of each $U_i$ is
        $\{R, p_1, \ldots, p_i\}$ and it is interpreted as $\overline{B}_R(p_i) \setminus \bigcup_{1 \le j \le i-1} \overline{B}_R(p_j)$. This representation
        immediately gives an algorithm computing $p \mapsto d(p, \overline{U_i})$.
$P \leftarrow \{\}$;
$R \leftarrow (m \mapsto$ findCoInnerRegularRadius($(2^{-n-1}, 2^{-n}), \{T_m\}_{m=1}^{\infty}, m$));
foreach $p_i$ in $T_{n+1}$ do
    $U_i \leftarrow \overline{B}_R(p_i) \setminus \bigcup_{U \in P} U$;
    $P \leftarrow P \cup \{U_i\}$;
end
return $P$ ;
```

Procedure findNicePartition($\{T_m\}_{m=1}^{\infty}$, n)

```
Data: $I$ is an interval which should include the output (a co-inner regular radius). $\{T_m\}_{m=1}^{\infty}$ is
      a sequence of maximal packings. $n$ is a target precision.
Result: A rational interval $I_n := [a_n, b_n]$ s.t.
        $(len(I_n) \le 2^{-n}) \wedge (I_n \subseteq I_{n-1}) \wedge |\mu(\overline{B}_{a_n}(p)) - \mu(\overline{B}_{b_n}(p))| \le 2^{-n}$.
$I_{n-1} := [a_{n-1}, b_{n-1}] \leftarrow$ findCoInnerRegularRadius($I, \{T_m\}_{m=1}^{\infty}, n-1$);
foreach $i$ in $\{0, \ldots, 10\}$ do $r_i \leftarrow \frac{i a_{n-1} + (10-i) b_{n-1}}{10}$;
Pick sufficiently large $N$ s.t. $2^{-N+2} \le \frac{b_{n-1} - a_{n-1}}{10}$;
foreach $i$ in $\{0, \ldots, 4\}$ do $m_{2i+1} \leftarrow$ pseudoCount($\overline{B}_{r_{2i+1}}(p), T_N, N$) ;
if $|m_5 - m_1| \le |m_9 - m_1|$ then $(i, j) \leftarrow (1, 5)$ else $(i, j) \leftarrow (5, 9)$;
return any subinterval in $[r_{i+1}, r_{j-1}]$ with length smaller than $2^{-n}$ ;
```

Procedure findCoInnerRegularRadius($I, \{T_m\}_{m=1}^{\infty}, n$)

3.4 Rigorous correctness proof of the explicit algorithm

Fact 3.4.1. *For any computable metric space (M, d, A, α), if the sizes of maximal packings κ_X is computable, then for any n, we can compute an encoding of a maximal n-packing T_n in A. In other words, we can compute $S \subseteq \Sigma^*$ s.t. $\alpha(S)$ is a maximal n-packing.*

Proof. First, let's show that A includes a maximal n-packing as a subset. Fix a maximal n-packing T_n and let $T_n := \{p_1, \ldots, p_N\}$. Since $\forall i, j.(i \neq j) \Rightarrow d(p_i, p_j) > 2^{-n}$, if we let $R := \min_{i \neq j} d(p_i, p_j)$ then $R > 2^{-n}$. Let $\delta = R - 2^{-n}$. Then $\forall p_i \in T_n \ \exists p_i' \in A$ s.t. $p_i' \in B_{\frac{\delta}{2}}(p_i)$ since A is dense. Thus, $\{p'_1, \ldots, p'_N\} \subseteq A$ and $\forall i, j.(i \neq j) \Rightarrow d(p'_i, p'_j) > d(p_i, p_j) - d(p_i, p'_i) - d(p_j, p'_j) > R - \delta \geq 2^{-n}$, which means $\{p'_1, \ldots, p'_N\}$ is a maximal n-packing.
Second, let's show that there is an algorithm that outputs a maximal n-packing in A if there is one. The algorithm is to dovetail the test of distance between $\kappa_X(n)$ element subsets of A. since the test does not includes equality, it is semidecidable. Thus, the algorithm will output a maximal n-packing if there is one.
Combining the first and the second step gives the computability of a maximal n-packing. □

Lemma 3.4.2. *Procedure* ***computeIntegral*** *is correct.*

Proof. m_f is a modulus of continuity [6, Definition 2.12] of f with precision $n+1$. This means $d(x, y) \leq 2^{-m_f} \Rightarrow |f(x) - f(y)| \leq 2^{-n-1}$. The partition $P = \{U_i\}_{i=1}^N$ satisfies that every U_i has radius smaller than 2^{-m_f}. So

$$\begin{aligned}
|q - \int_X f d\mu| &= |\sum_{i=1}^N m_i f(p_i) - \int_X f d\mu| \\
&\leq \sum_{i=1}^N 2^{-n-1-i-\log M} f(p_i) + |\sum_{i=1}^N \mu(U_i) f(p_i) - \int_X f d\mu| \\
&\leq \sum_{i=1}^N 2^{-n-1-i-\log M} M + \sum_{i=1}^N \mu(U_i) 2^{-n-1} \\
&\leq 2^{-n-1} 2^{-\log M} M + 2^{-n-1} \sum_{i=1}^N \mu(U_i) \leq 2^{-n}
\end{aligned}$$

□

Lemma 3.4.3. *Procedure* ***findNicePartition*** *is correct.*

Proof. The followings are proofs of postconditions in the same order in the pseudocode.

1. $\{U_i\}_{i=1}^N$ is a partition because they are disjoint, and they covers the whole space since $X = \bigcup_{p \in T_{n+1}} \overline{B}_{2^{-n-1}}(p) \subseteq \bigcup_{U \in P} U$.

2. $\overline{U_i}$ are clearly closed. They are computably located because $\bigcup_{U \in P} U = \bigcup \overline{B}_R(p_i) \Rightarrow \overline{U_i}$ is of the form $\overline{\overline{B}_R(p) \setminus \bigcup \overline{B}_R(p_i)}$ and every R, p_i are computable.

3. $\overline{U_i}$ are co-inner regular because $\overline{B}_R(p_i)$ are co-inner regular and it is preserved under intersection, union, complement, and closure. $\overline{B}_R(p_i)$ are co-inner regular because the postcondition of the procedure findInterval ensures that R is a co-inner regular radius.

4. $U_i \subseteq \overline{B}_R(p_i) \subseteq \overline{B}_{2^{-n}}(p_i) \Rightarrow U_i$ is contained in a closed ball with radius 2^{-n}.

□

Lemma 3.4.4. *Procedure **findCoInnerRegularRadius** is correct.*

Proof. Because of lemma 3.1.5 and the fact that N is sufficiently large, $\mu(\overline{B}_{r_{2i}}(p)) \leq \mu_{T_N}(\overline{B}_{r_{2i+1}}(p)) \leq m_{2i+1} \leq \mu_{T_N}(\overline{B}_{r_{2i+1}+2^{-N}}(p)) \leq \mu(\overline{B}_{r_{2i+2}}(p))$. Then since $2^{-n+1} \geq |\mu(\overline{B}_{a_{n-1}}(p)) - \mu(\overline{B}_{b_{n-1}}(p))| \geq |m_9 - m_1| \geq |m_9 - m_5| + |m_5 - m_1|$, WLOG $|m_5 - m_1| \leq 2^{-n}$. Then $|\mu(\overline{B}_{r_4}(p)) - \mu(\overline{B}_{r_2}(p))| \leq |m_5 - m_1| \leq 2^{-n}$. □

Remark 3.4.5. If $\circ$ and $\circ'$ are both bi-invariant for compact separable (X, d), then $\circ$ and $\circ'$ induce the same Haar measure. In fact, this irrelevance is originated from lemma 3.1.5. Note that the approximation of the measure, μ_{T_n}, is independent of the given group operation.

Chapter 4. Computing the Haar Integral on $\mathcal{SO}(3)$

In this chapter, we present an algorithm computing Haar integrals on $\mathcal{SO}(3)$ and prove that it is in the same complexity class with computing the Haar integrals on $\mathcal{U}(1)$ (note that the Haar integral on $\mathcal{U}(1)$ is equivalent to the usual integral on [0,1]). The key observations are fact 4.2.1 and that integral on $\mathbb{R}^3$ is well-known. The argument also holds true for $\mathcal{SU}(2), \mathcal{O}(3)$, and $\mathcal{U}(2)$ with only small adjustments. Recall the goal of this chapter:

Theorem 4.0.1. *If $f \in \mathcal{C}(\mathcal{SO}(3))$ is polynomial-time computable, then $\int f\ d\mu \in \mathbb{R}$ is $\#P_1$-computable. If for all $f \in \mathcal{C}(\mathcal{SO}(3)), (f$ is polynomial-time computable implies $\int f\ d\mu \in \mathbb{R}$ is polynomial-time computable), then $(FP_1 = \#P_1)$.*

4.1 Algorithm

The algorithm has two differences from the algorithm of the previous chapter. First, since now the domain of the functions that algorithm recieves is fixed to $\mathcal{SO}(3) \subseteq \mathbb{R}^{3\times 3}$, it is unproblematic to analyze its complexity. Second, it can be easily implemented in the framework **iRRAM** that realizes paradigm ERC'[2], and actually an implementation will be presented in chapter 5. Implementation of Haar integrals that maps functions on compact metric groups to reals is not easy to implement in this framework, since it does not support a data structure for the computable metric space yet.

The idea of the algorithm is to exploit the canonical surjective mapping $g :\subseteq \mathbb{R}^3 \to \mathcal{SO}(3)$ that arises from the unit Fraternisation representation of $\mathcal{SU}(2)$. With this canonical mapping, it is sufficient to transform $f : \mathcal{SO}(3) \to \mathbb{R}$ to $f \circ g :\subseteq \mathbb{R}^3 \to \mathbb{R}$, compute $\int f \circ g$ on $\mathbb{R}^3$, and prove that what this process actually computes is the Haar integral on $\mathcal{SO}(3)$. The proof is known in pure mathematics, so we will only give a sketch of the proof.

First, recall that quaternions are an extension of complex numbers that has i, j, k for fundamental quaternion units. Unit quaternions are quaternions of norm 1, denoted by $\mathbb{H}_1$. $\mathbb{H}_1$ has a trivial bijection to $\mathcal{S}^3$. That is, $G = (a, b, c, d) \mapsto a + bi + cj + dk : \mathcal{S}^3 \to \mathbb{H}_1$. $\mathcal{SU}(2)$ is a group isomorphic to $\mathbb{H}_1$ with the map

$$a + bi + cj + dk = \alpha + \beta j \mapsto \begin{bmatrix} \alpha & -\overline{\beta} \\ \beta & \overline{\alpha} \end{bmatrix} \tag{4.1}$$

where $\alpha := a+bi, \beta := c+di$. Let this isomorphism to be $F_{\mathcal{SU}(2)}$. The isomorphism gives a representation of $\mathcal{SU}(2)$ with $\mathbb{H}_1$, since $\mathbb{H}_1$ can be represented with $\mathcal{S}^3$, and then $\mathcal{S}^3$ can be represented as a subset of $\mathbb{R}^4$. To elaborate, if we let $\delta : \Sigma^\omega \to \mathcal{S}^3$ to denote a representation of $\mathcal{S}^3$, then $F_{\mathcal{SU}(2)} \circ G \circ \delta : \Sigma^\omega \to \mathcal{SU}(2)$ is a representation of $\mathcal{SU}(2)$.

Similarly, $\mathcal{SO}(3)$ has a 2-to-1 and onto homomorphism from $\mathbb{H}_1$ (here, 2-to-1 means exactly two elements of the domain are mapped to one element in the codomain). To define this homomorphism, first note that $v := (x, y, z) \in \mathbb{R}^3$ with $\|v\| = 1$ can be identified with $xi + yj + zk \in \mathbb{H}_1$. Using this embedding and the fact that $\mathcal{SO}(3)$ represents the rotations in $\mathbb{R}^3$,

$$q \in \mathbb{H}_1 \mapsto (v \in \mathbb{R}^3 \mapsto qvq^{-1} \in \mathbb{R}^3) \in \mathcal{SO}(3) \tag{4.2}$$

is well defined and gives the desired homomorphism. Let this homomorphism to be $F_{\mathcal{SO}(3)}$. This also gives a representation of $\mathcal{SO}(3)$, since if δ is a representation of $\mathbb{H}_1$ then $F_{\mathcal{SO}(3)} \circ G \circ \delta$ is a representation

of $\mathcal{SO}(3)$.
Now we present the algorithm and the fact that our algorithm is based on.

Theorem 4.1.1. *The below procedure **computeIntegralSO3** computes the Haar integral on $\mathcal{SO}(3)$.*

```
Data: $f : \mathcal{SO}(3) \to \mathbb{R}$ where $\mathcal{SO}(3)$ is represented with $\mathbb{H}_1$ and $\mathbb{H}_1$ is represented with $\mathcal{S}^3$.
Result: $\int_{\mathcal{SO}(3)} f d\mu$ where $\mu$ is the Haar measure of $\mathcal{SO}(3)$.
const $g \leftarrow$ unitCubeToH1 ;  // unitCubeToH1 is essentially $F_{\mathcal{SO}(3)} \circ G \circ \Psi$ in fact 4.2.1
$f \circ g \leftarrow$ composition($f$, $g$);
return integralOnUnitCube($f \circ g$);
Function unitCubeToH1($x : \mathbb{R}, y : \mathbb{R}, z : \mathbb{R}$): $\mathcal{SO}(3)$ is
    $\eta, \vartheta, \varphi \leftarrow \pi x, \pi y, 2\pi z$;
    return $(\cos\eta,\ \sin\eta\cos\vartheta,\ \sin\eta\sin\vartheta\cos\varphi,\ \sin\eta\sin\vartheta\sin\varphi)$;
end
```

Procedure computeIntegralSO3(f)

Proof. It is immediate from fact 4.2.1. □

Remark 4.1.2. This algorithm can be used to get Haar integral on $\mathcal{SU}(2)$, by simply inputting the function on $\mathcal{SU}(2)$ (the algorithm still can be executed since $\mathcal{SU}(2)$ also can be represented with $\mathbb{H}_1$). Additionally, note that simply $\mathcal{O}(3) \cong \mathcal{SO}(3) \times \{\pm 1\}$ and $\mathcal{U}(2) \cong \mathcal{SU}(2) \times \mathcal{U}(1)$. Consequently, $\mathcal{O}(3)$ can be represented by $(p, sign) \in \mathbb{H}_1 \times \{\pm 1\}$ and $\mathcal{U}(2)$ can be represented by $(p, z) \in \mathbb{H}_1 \times \mathcal{U}(1)$. If $f : \mathcal{O}(3) \to \mathbb{R}$ then simply **computeIntegralSO3**$(q \mapsto f(q, +1))$ + **computeIntegralSO3**$(q \mapsto f(q, -1))$ gives $2\int_{\mathcal{O}(3)} f d\mu$. Similarly, if $f : \mathcal{U}(2) \to \mathbb{R}$ then simply **computeIntegralSO3**$(q \mapsto \int_{\mathcal{U}(1)} f(q, z)dz)$ gives $\int_{\mathcal{U}(2)} f d\mu$.

4.2 Correctness

Fact 4.2.1. *Let $f : \mathcal{SO}(3) \to \mathbb{R}$ be continuous, $G : \mathcal{S}^3 \to \mathbb{H}_1$ be the trivial bijection, $F_{\mathcal{SO}(3)}$ be the 2-to-1 surjective group homomorphism, and $\Psi : [0, \pi) \times [0, \pi) \times [0, 2\pi) \to \mathcal{S}^3$ be the generalized spherical coordinates. Then the following equation holds:*

$$\int f d\mu_{\mathcal{SO}(3)} = \int f \circ F_{\mathcal{SO}(3)} d\mu_{\mathbb{H}_1} = \int f \circ F_{\mathcal{SO}(3)} \circ G dV = \int_0^{\pi} \int_0^{\pi} \int_0^{2\pi} f \circ F_{\mathcal{SO}(3)} \circ G \circ \Psi det|\Psi'| d\eta d\theta d\phi \tag{4.3}$$

Proof. The first equality comes from the fact that $\mu_{\mathcal{SO}(3)}(U) = \mu_{\mathbb{H}_1}(F^{-1}_{\mathcal{SO}(3)}(U))$, since $F_{\mathcal{SO}(3)}$ is a group homomorphism, and the measures are translation-invariant. The second equality similarly comes since $V(G^{-1}(U)) = \mu_{\mathbb{H}_1}(U)$ because of the invariance of $V(\cdot)$ under rigid motion and the fact that the group operation in $\mathbb{H}_1$ is a rigid motion in $\mathcal{S}^3$. The last equality comes from the change of coordinate. □

Although it is proven here, it is stated as a fact because a similar way to calculate Haar integral on compact Lie groups is already known as Weyl's integration formula.

4.3 Complexity

Complexity of our algorithm is equivalent to that of $\mathbb{R}^3$, since the change of coordinate is done in polynomial-time. It is not hard to see that integration on $\mathbb{R}^3$ is in the same complexity class with $\mathbb{R}$ (see [13]). Recall the following complexity result on $\mathbb{R}$:

Fact 4.3.1. *[6, Theorem 5.32] If $f \in \mathcal{C}([0,1])$ is polynomial-time computable, then $\int f\ d\mu \in \mathbb{R}$ is $\#P_1$ computable. If for all $f \in \mathcal{C}([0,1]), (f$ is polynomial-time computable implies $\int f\ d\mu \in \mathbb{R}$ is polynomial-time computable), then $(FP_1 = \#P_1)$.*

Note that it is very same with theorem 4.0.1, and only $\mathcal{SO}(3)$ is changed to $[0,1]$.

Chapter 5. Implementation and Experiment

In this chapter, we discuss implementation of the algorithm computing Haar integral on $\mathcal{SO}(3)$, mentioned in chapter 4. We used the framework **iRRAM** that realizes the Exact Real Computation paradigm [2]. The source code is available at `https://github.com/instigation/HaarIntegral`.

5.1 Implementation

The algorithm assumes that $\mathcal{SO}(3)$ is represented in $\mathcal{S}^3$. Alternatively, $\mathcal{SO}(3)$ can be represented as a subspace of $\mathbb{R}^{3\times 3}$. This decision of representation has two reasons. The first is performance. As you can see in section 5.4, the algorithm is already slow. If $\mathcal{SO}(3)$ is represented as a subspace of $\mathbb{R}^{3\times 3}$, there will be additional performance drop due to the change of representation. Second is that $\mathcal{S}^3$ representation of $\mathcal{SO}(3)$ is not rare. Representing $\mathcal{SO}(3)$ with $\mathbb{H}_1$ (and thus with $\mathcal{S}^3$) is famous in, for instance, Computer Graphics.

5.2 Environment

The experiment was done on a virtual machine. The underlying computer has Intel(R) Core(TM) i7-7700K CPU 4.20GHz and 16Gb RAM. The virtual machine was Ubuntu 64bit with 4 core and 8Gb RAM by VMware Workstation 15 Player.

5.3 Results

The experiment is done on the function $(w, x, y, z) \mapsto |w| + |x| + |y| + |z| : \mathcal{S}^3 \to \mathbb{R}$.

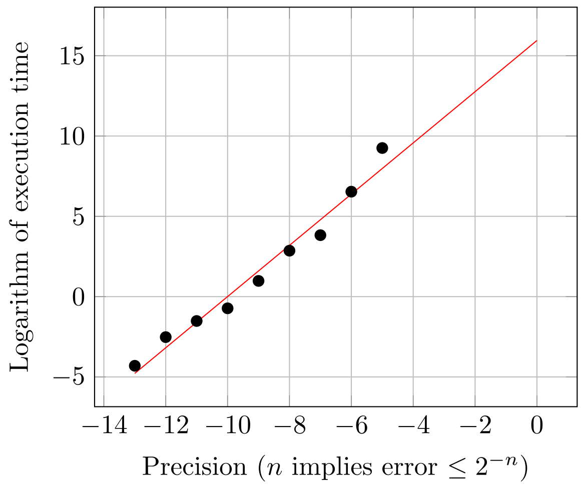

Figure 5.1: x-axis is the precision n. That is, the output is guaranteed to has error at most 2^{-n}. y-axis means the logarithm of the CPU time of execution. Execution time for each precision is the average execution time of 5 executions.

5.4 Discussion

The experiment shows that the execution time of the algorithm is asymptotically exponential time, which is optimal unless $FP_1 = \#P_1$. Note that the y-axis of figure 5.3 is *logarithm* of the execution time. Precision n means the output is guaranteed to has error at most 2^{-n}. That is, higher precision means higher accuracy.
In the experiment, experimental correctness test is not done. It is infeasible to do so since the execution time would be about 2^{14} seconds (about 4.5 hours) even if the output of precision is ± 2 ($n = -1$). Since the Haar measure is probability measure, and the function we integrated is at most 2, it is straightforward that the output is between 0 and 2. Thus the output with the precision ± 2 gives no additional information than the straightforward analysis. That means it will take couple of months to run even a few meaningful test cases.

5.5 Experimental Technique

One technical issue to experiment exponential algorithm with **iRRAM** is that it computes a real number with a precision of 50 first. That is, users cannot demand **iRRAM** to compute a real with arbitrary precision that they want. Consequently, for instance, if the algorithm runs in 2^n seconds, it will take at least 2^{50} seconds and it becomes infeasible.
However, one can workaround this big precision issue. Let an algorithm that computes r to be $\mathcal{A}_r$. If one wants to experiment $\mathcal{A}_r$ with precision $n < 50$, one can easily modify $\mathcal{A}_r$ and make $\mathcal{A}_{2^{n-50}r}$ (which computes $2^{n-50}r$ in a similar way with $\mathcal{A}_r$). Then, if we experiment on $\mathcal{A}_{2^{n-50}r}$, **iRRAM** will compute an approximation of $2^{n-50}r$ with precision 50. Let this approximation to be a. Then $|a - 2^{n-50}r| \leq 2^{-50}$, which gives $|2^{50-n}a - r| \leq 2^n$. Thus, one can get an approximation of r with precision n by computing $2^{50-n}a$.

Bibliography

[1] Vasco Brattka, Joseph Miller, Stéphane Le Roux, and Arno Pauly. Connected choice and Brouwer's fixed point theorem. *Journal for Mathematical Logic*, 20XX. doi:10.1142/S0219061319500041.

[2] Franz Brauße, Pieter Collins, SunYoung Kim, Michal Konecnỳ, Gyesik Lee, Norbert Müller, Eike Neumann, Sewon Park, Norbert Preining, and Martin Ziegler. Semantics, logic, and verification of" exact real computation". *arXiv preprint arXiv:1608.05787*, 2016.

[3] Pieter Collins. Computable stochastic processes. arXiv:1409.4667, 2014.

[4] Xiaolin Ge and Anil Nerode. On extreme points of convex compact turing located set. In Anil Nerode and Yu. V. Matiyasevich, editors, *Logical Foundations of Computer Science*, pages 114–128, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[5] Akitoshi Kawamura, Florian Steinberg, and Martin Ziegler. Complexity theory of (functions on) compact metric spaces. In *Proc. 31st Ann. Symposium on Logic in Computer Science, LICS*, pages 837–846. ACM, 2016. doi:10.1145/2933575.2935311.

[6] Ker-I Ko. *Complexity Theory of Real Functions.* Progress in Theoretical Computer Science. Birkhäuser, Boston, 1991.

[7] Ulrich Kohlenbach. *Applied Proof Theory: Proof Interpretations and their Use in Mathematics.* Springer, Berlin, 2008.

[8] Vladimir P. Orevkov. A constructive mapping of the square onto itself displacing every constructive point (Russian). *Doklady Akademii Nauk*, 152:55–58, 1963. translated in: Soviet Math. - Dokl., 4 (1963) 1253–1256.

[9] Arno Pauly. On the topological aspects of the theory of represented spaces. *Computability*, 5(2):159–180, 2016. doi:10.3233/COM-150049.

[10] Arno Pauly, Dongseong Seon, and Martin Ziegler. Computing haar measures, 2019. arXiv:arXiv:1906.12220.

[11] Marian B. Pour-El and J. Ian Richards. A computable ordinary differential equation which possesses no computable solution. *Annals Math. Logic*, 17:61–90, 1979.

[12] Matthias Schröder and Alex Simpson. Representing probability measures using probabilistic processes. *J. Complexity*, 22(6):768–782, 2006. URL: https://doi.org/10.1016/j.jco.2006.05.003, doi:10.1016/j.jco.2006.05.003.

[13] Florian Steinberg. Complexity theory for spaces of integrable functions. *Logical Methods in Computer Science*, Volume 13, Issue 3, September 2017. doi:10.23638/LMCS-13(3:21)2017.

[14] Klaus Weihrauch. *Computable Analysis.* Springer, Berlin, 2000.

[15] Klaus Weihrauch. Computational complexity on computable metric spaces. *Mathematical Logic Quarterly*, 49(1):3–21, 2003.