

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

On Measure Quantifiers in First-Order Arithmetic

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:

Antonelli M., Dal Lago U., Pistone P. (2021). On Measure Quantifiers in First-Order Arithmetic. Cham : Springer [10.1007/978-3-030-80049-9_2].

Availability:

This version is available at: <https://hdl.handle.net/11585/834267> since: 2021-10-04

Published:

DOI: http://doi.org/10.1007/978-3-030-80049-9_2

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>).
When citing, please refer to the published version.

(Article begins on next page)

This is the final peer-reviewed accepted manuscript of:

Antonelli M., Dal Lago U., Pistone P. (2021) On Measure Quantifiers in First-Order Arithmetic. In: De Mol L., Weiermann A., Manea F., Fernández-Duque D. (eds) Connecting with Computability. CiE 2021. Lecture Notes in Computer Science, vol 12813. Springer, Cham.

The final published version is available online at: https://doi.org/10.1007/978-3-030-80049-9_2

Rights / License:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>)

When citing, please refer to the published version.

On Measure Quantifiers in First-Order Arithmetic*

(Long Version)

Melissa Antonelli

Ugo Dal Lago

Paolo Pistone

Abstract

We study the logic obtained by endowing the language of first-order arithmetic with second-order measure quantifiers. This new kind of quantification allows us to express that the argument formula is true *in a certain portion* of all possible interpretations of the quantified variable. We show that first-order arithmetic with measure quantifiers is capable of formalizing simple results from probability theory and, most importantly, of representing every recursive random function. Moreover, we introduce a realizability interpretation of this logic in which programs have access to an oracle from the Cantor space.

1 Introduction

The interactions between first-order arithmetic and the theory of computation are plentiful and deep. On the one side, proof systems for arithmetic can be used to prove termination of certain classes of algorithms [61], or to establish complexity bounds [8]. On the other, higher-order programming languages, such as typed λ -calculi, can be proved to capture the computational content of arithmetical proofs. These insights can be pushed further, giving rise to logical and type theories of various strengths. Remarkably, all the quoted research directions rely on the tight connection between the concepts of *totality* (of functions) and *termination* (of algorithms).

However, there is one side of the theory of computation which was only marginally touched by this fruitful interaction, that is, randomized computation. Probabilistic models have been widely investigated and are nowadays pervasive in many areas of computer science. The idea of relaxing the notion of algorithm to account for computations involving random decisions appeared early in the history of modern computability theory and studies on probabilistic computation have been developed since the 1950s and 1960s [9, 40, 14, 51, 52, 57]. Today several formal models are available, such as probabilistic automata [55, 49], both Markovian and oracle probabilistic Turing machines (from now on, PTMs) [14, 53, 21, 22], and probabilistic λ -calculi [50, 31]. At this point randomized computation is ubiquitous.

In probabilistic computation, behavioral properties, such as termination, have a *quantitative* nature: any computation terminates *with a given probability*. Can such quantitative properties be studied within a logical system? Of course, logical systems for set-theory and second-order logic can be expressive enough to represent measure theory [58] and, thus, are inherently capable of talking about randomized computations. Yet, what should one add to *first-order* arithmetic to make it capable of describing probabilistic computation?

In this paper we provide an answer to this question by introducing a somehow *minimal* extension of first-order Peano Arithmetic by means of measure quantifiers. We will call this system MQPA. Its language is obtained by enriching the language of PA with a special unary predicate, $\text{FLIP}(\cdot)$, whose interpretation is an element of the Cantor space, $\{0, 1\}^{\mathbb{N}}$, and with measure-quantified formulas, such as $\mathbf{C}^{\frac{1}{2}}F$, which expresses the fact that F has probability $\geq \frac{1}{2}$ of being true (that is, the subset of $\{0, 1\}^{\mathbb{N}}$ which makes A true has measure $\geq \frac{1}{2}$). The appeal to the Cantor space is essential here, since there is no *a priori* bound on the amount of random bits a given

*Supported by ERC CoG “DIAPASoN”, GA 818616.

computation might need; at the same time, we show that it yields a very natural measure-theoretic semantics.

The rest of this paper is structured as follows. In Section 2 we introduce the syntax and semantics of MQPA. Section 3 shows that some non-trivial results in probability theory can be naturally expressed in MQPA. In Section 4 we establish our main result, that is, a representation theorem within MQPA for random functions computed by PTMs, which is the probabilistic analogous to Gödel’s arithmetization theorem for recursive functions in PA [23]. Finally, in Section 5, a realizability interpretation for MQPA in terms of computable functions with oracles on the Cantor space is presented.

2 Measure-Quantified Peano Arithmetic

This section is devoted to the introduction of the syntax and semantics for formulas of MQPA. Before the actual presentation, we need some (very modest) preliminaries from measure theory.

Preliminaries. The standard model $(\mathbb{N}, +, \times)$ has nothing probabilistic in itself. Nevertheless, it can be naturally extended into a probability space: arithmetic being discrete, one may consider the underlying sample space as just $\mathbb{B}^{\mathbb{N}}$, namely the set of all infinite sequences of elements from $\mathbb{B} = \{0, 1\}$. We will use metavariables, such as $\omega_1, \omega_2, \dots$, for the elements of $\mathbb{B}^{\mathbb{N}}$. As it is known, there are standard ways of building a well behaved σ -algebra and a probability space on $\mathbb{B}^{\mathbb{N}}$, which we will briefly recall here. The subsets of $\mathbb{B}^{\mathbb{N}}$ of the form

$$C_X = \{s \cdot \omega \mid s \in X \text{ \& } \omega \in \mathbb{B}^{\mathbb{N}}\},$$

where $X \subseteq \mathbb{B}^n$ and \cdot denotes sequence concatenation, are called *n-cylinders* [7]. Specifically, we are interested in X s defined as follows: $X_n^b = \{s \cdot b \mid s \in \mathbb{B}^n \text{ \& } b \in \mathbb{B}\} \subseteq \mathbb{B}^{n+1}$, with $n \in \mathbb{N}$. We will often deal with cylinders of the form $C_{X_n^1}$. We let \mathcal{C}_n and \mathcal{C} indicate the set of all n -cylinders and the corresponding algebra, made of the open sets of the natural topology on $\mathbb{B}^{\mathbb{N}}$. The smallest σ -algebra including \mathcal{C} , which is Borel, is indicated as $\sigma(\mathcal{C})$. There is a natural way of defining a probability measure $\mu_{\mathcal{C}}$ on \mathcal{C} , namely by assigning to C_X the measure $\frac{|X|}{2^n}$. There exists canonical ways to extend this to $\sigma(\mathcal{C})$. In doing so, the standard model $(\mathbb{N}, +, \times)$ can be generalized to $\mathcal{P} = (\mathbb{N}, +, \times, \sigma(\mathcal{C}), \mu_{\mathcal{C}})$, which will be our standard model for MQPA [1]. When interpreting sequences in $\mathbb{B}^{\mathbb{N}}$ as infinite supplies of random bits, the set of sequences such that the k -th coin flip’s result is 1 (for any fixed k) is assigned measure $\frac{1}{2}$, meaning that each random bit is uniformly distributed and independent from the others.

Syntax. We now introduce the syntax of MQPA. Terms are defined as in classic first-order arithmetic. Instead, the formulas of MQPA are obtained by endowing the language of PA with *flipcoin formulas*, such as $\text{FLIP}(t)$, and *measure-quantified formulas*, as for example $\mathbf{C}^{t/s}F$ and $\mathbf{D}^{t/s}F$. Specifically, $\text{FLIP}(\cdot)$ is a special unary predicate with an intuitive computational meaning. It basically provides an infinite supply of independently and randomly distributed bits. Intuitively, given a closed term t , $\text{FLIP}(t)$ holds if and only if the n -th tossing returns 1, where n is the denotation of $t + 1$.

Definition 1 (Terms and Formulas of MQPA) *Let \mathcal{G} be a denumerable set of ground variables, whose elements are indicated by metavariables such as x, y . The terms of MQPA, denoted by t, s , are defined as follows:*

$$t, s := x \mid 0 \mid \mathbf{S}(t) \mid t + s \mid t \times s.$$

The formulas of MQPA are defined by the following grammar:

$$F, G := \text{FLIP}(t) \mid (t = s) \mid \neg F \mid F \vee G \mid F \wedge G \mid \exists x.F \mid \forall x.F \mid \mathbf{C}^{t/s}F \mid \mathbf{D}^{t/s}F.$$

¹Here, we will focus on this structure as a “standard model” of MQPA, leaving the study of alternative models for future work.

Semantics. Given an environment $\xi : \mathcal{G} \rightarrow \mathbb{N}$, the interpretation $\llbracket t \rrbracket_\xi$ of a term t is defined as usual.

Definition 2 (Semantics for Terms of MQPA) An environment ξ is a mapping that assigns to each ground variable a natural number, $\xi : \mathcal{G} \rightarrow \mathbb{N}$. Given a term t and an environment ξ , the interpretation of t in ξ is the natural number $\llbracket t \rrbracket_\xi \in \mathbb{N}$, inductively defined as follows:

$$\begin{aligned} \llbracket x \rrbracket_\xi &:= \xi(x) \in \mathbb{N} & \llbracket t + s \rrbracket_\xi &:= \llbracket t \rrbracket_\xi + \llbracket s \rrbracket_\xi \\ \llbracket 0 \rrbracket_\xi &:= 0 & \llbracket t \times s \rrbracket_\xi &:= \llbracket t \rrbracket_\xi \times \llbracket s \rrbracket_\xi \\ \llbracket S(t) \rrbracket_\xi &:= \llbracket t \rrbracket_\xi + 1 \end{aligned}$$

Instead, the interpretation of formulas requires a little care, being it inherently *quantitative*: any formula F is associated with a *measurable* set, $\llbracket F \rrbracket_\xi \in \sigma(\mathcal{C})$ (similarly to e.g. [44]).

Definition 3 (Semantics for Formulas of MQPA) Given a formula F and an environment ξ , the interpretation of F in ξ is the measurable set of sequences $\llbracket F \rrbracket_\xi \in \sigma(\mathcal{C})$ inductively defined as follows:

$$\begin{aligned} \llbracket \text{FLIP}(t) \rrbracket_\xi &:= C_{X_{\llbracket t \rrbracket_\xi}^1} & \llbracket G \vee H \rrbracket_\xi &:= \llbracket G \rrbracket_\xi \cup \llbracket H \rrbracket_\xi \\ \llbracket t = s \rrbracket_\xi &:= \begin{cases} \mathbb{B}^\mathbb{N} & \text{if } \llbracket t \rrbracket_\xi = \llbracket s \rrbracket_\xi \\ \emptyset & \text{otherwise} \end{cases} & \llbracket G \wedge H \rrbracket_\xi &:= \llbracket G \rrbracket_\xi \cap \llbracket H \rrbracket_\xi \\ \llbracket \neg G \rrbracket_\xi &:= \mathbb{B}^\mathbb{N} - \llbracket G \rrbracket_\xi & \llbracket \exists x. G \rrbracket_\xi &:= \bigcup_{i \in \mathbb{N}} \llbracket G \rrbracket_{\xi\{x \leftarrow i\}} \\ & & \llbracket \forall x. G \rrbracket_\xi &:= \bigcap_{i \in \mathbb{N}} \llbracket G \rrbracket_{\xi\{x \leftarrow i\}} \\ \llbracket \mathbf{C}^{t/s} G \rrbracket_\xi &:= \begin{cases} \mathbb{B}^\mathbb{N} & \text{if } \llbracket s \rrbracket_\xi > 0 \text{ and } \mu_{\mathcal{C}}(\llbracket G \rrbracket_\xi) \geq \llbracket t \rrbracket_\xi / \llbracket s \rrbracket_\xi \\ \emptyset & \text{otherwise} \end{cases} \\ \llbracket \mathbf{D}^{t/s} G \rrbracket_\xi &:= \begin{cases} \mathbb{B}^\mathbb{N} & \text{if } \llbracket s \rrbracket_\xi = 0 \text{ or } \mu_{\mathcal{C}}(\llbracket G \rrbracket_\xi) < \llbracket t \rrbracket_\xi / \llbracket s \rrbracket_\xi \\ \emptyset & \text{otherwise} \end{cases} \end{aligned}$$

The semantics is well-defined since the sets $\llbracket \text{FLIP}(t) \rrbracket_\xi$ and $\llbracket t = s \rrbracket_\xi$ are measurable, and measurability is preserved by all the logical operators. It is not difficult to see that any n -cylinder can be captured as the interpretation of some MQPA formula. However, the language of MQPA allows us to express more and more complex measurable sets, as illustrated in the next sections.

The notions of validity and logical equivalence are defined in a standard way.

Definition 4 A formula of MQPA, F , is valid if and only if for every ξ , $\llbracket F \rrbracket_\xi = \mathbb{B}^\mathbb{N}$. Two MQPA formulas F, G are logically equivalent $F \equiv G$ if and only if for every ξ , $\llbracket F \rrbracket_\xi = \llbracket G \rrbracket_\xi$.

Notably, the two measure quantifiers are inter-definable, since one has $\llbracket \mathbf{C}^{t/s} F \rrbracket_\xi = \llbracket \neg \mathbf{D}^{t/s} F \rrbracket_\xi$.

Lemma 1 For every formula of MQPA, call it F :

$$\mathbf{C}^{t/s} F \equiv \neg \mathbf{D}^{t/s} F.$$

Proof. The proof is based on Definition 3.

$$\begin{aligned} \llbracket \neg \mathbf{D}^{t/s} F \rrbracket_\xi &= \mathbb{B}^\mathbb{N} - \llbracket \mathbf{D}^{t/s} F \rrbracket_\xi \\ &= \mathbb{B}^\mathbb{N} - \begin{cases} \mathbb{B}^\mathbb{N} & \text{if } \mu_{\mathcal{C}}(\llbracket F \rrbracket_\xi) < \llbracket t \rrbracket_\xi / \llbracket s \rrbracket_\xi \\ \emptyset & \text{otherwise} \end{cases} \\ &= \begin{cases} \emptyset & \text{if } \mu_{\mathcal{C}}(\llbracket F \rrbracket_\xi) < \llbracket t \rrbracket_\xi / \llbracket s \rrbracket_\xi \\ \mathbb{B}^\mathbb{N} & \text{otherwise} \end{cases} \\ &= \llbracket \mathbf{C}^{t/s} F \rrbracket_\xi. \end{aligned}$$

□

The following examples illustrate the use of measure-quantifiers $\mathbf{C}^{t/s}$ and $\mathbf{D}^{t/s}$ and, in particular, the role of probabilities of the form $\frac{t}{s}$.

Example 1 The formula $F = \mathbf{C}^{1/1} \exists x. \text{FLIP}(x)$ states that a true random bit will almost surely be met. It is valid, as the set of constantly 0 sequences forms a singleton, which has measure 0.

Example 2 The formula² $F = \forall x. \mathbf{C}^{1/2^x} \forall y \leq x. \text{FLIP}(y)$ states that the probability for the first x random bits to be true is at least $\frac{1}{2^x}$. This formula is valid too.

3 On the Expressive Power of MQPA

As anticipated, the language of MQPA allows us to express some elementary results from probability theory, and to check their validity in the structure \mathcal{P} . In this section we sketch a couple of examples.

The Infinite Monkey Theorem. Our first example is the so-called *infinite monkey theorem* (IMT). It is a classic result from probability theory stating that a monkey randomly typing on a keyboard has probability 1 of ending up writing the *Macbeth* (or any other fixed string), sooner or later. Let the formulas $F(x, y)$ and $G(x, y)$ of PA express, respectively, that “ y is strictly smaller than the length of (the binary sequence coded by) x ”, and that “the $y+1$ -th bit of x is 1”. We can formalize IMT through the following formula:

$$F_{\text{IMT}} : \forall x. \mathbf{C}^{1/1} \forall y. \exists z. \forall w. F(x, w) \rightarrow (G(x, w) \leftrightarrow \text{FLIP}(y + z + w)).$$

Indeed, let x be a binary encoding of the *Macbeth*. The formula F_{IMT} says then that for all choice of start time y , there exists a time $y + z$ after which $\text{FLIP}(\cdot)$ will evolve exactly like x with probability 1.

How can we justify F_{IMT} using the semantics of MQPA? Let $\varphi(x, y, z, w)$ indicate the formula $F(x, w) \rightarrow (G(x, w) \leftrightarrow \text{FLIP}(y + z + w))$. We must show that for all natural number $n \in \mathbb{N}$, there exists a measurable set $S^n \subseteq \mathbb{B}^{\mathbb{N}}$ of measure 1 such that any sequence in S^n satisfies the formula $\forall y. \exists z. \forall w. \varphi(n, y, z, w)$. To prove this fact, we rely on a well-known result from measure theory, namely the *second Borel-Cantelli Lemma*:

Theorem 1 ([7], Thm. 4.4, p. 55) *If $(U_y)_{y \in \mathbb{N}}$ is a sequence of independent events in $\mathbb{B}^{\mathbb{N}}$, and $\sum_y \mu_{\mathcal{C}}(U_y)$ diverges, then $\mu_{\mathcal{C}}(\bigcap_y \bigcup_{z > y} U_z) = 1$.*

Let us fix $n \in \mathbb{N}$ and let $\ell(n)$ indicate the length of the binary string encoded by n . We suppose for simplicity that $\ell(n) > 0$ (as the case $\ell(n) = 0$ is trivial). We construct S^n in a few steps as follows:

- for all $p \in \mathbb{N}$, let U_p^n be the cylinder of sequences which, after p steps, agree with n ; observe that the sequences in U_p^n satisfy the formula $\forall w. \varphi(n, p, 0, w)$;
- for all $p \in \mathbb{N}$, let $V_p^n = U_{p \cdot \ell(n) + 1}^n$; observe that the sets V_p^n are pairwise independent and $\mu_{\mathcal{C}}(\sum_p V_p^n) = \infty$;
- for all $p \in \mathbb{N}$, let $S_p^n = \bigcup \{U_{p+q}^n \mid \exists s > p. p + q = s \cdot \ell(n) + 1\}$. Observe that any sequence in S_p^n satisfies $\exists z. \forall w. \varphi(n, p, z, w)$; Moreover, one can check that $S_p^n = \bigcup_{q > p} V_q^n$;
- we finally let $S^n := \bigcap_p S_p^n$.

We now have that any sequence in S^n satisfies $\forall y. \exists z. \forall w. \varphi(n, y, z, w)$; furthermore, by Theorem 1, $\mu_{\mathcal{C}}(S^n) = \mu_{\mathcal{C}}(\bigcap_p \bigcup_{q > p} V_q^n) = 1$. Thus, for each choice of $n \in \mathbb{N}$, $\mu_{\mathcal{C}}(\llbracket \forall y. \exists z. \forall w. \varphi(x, p, z, w) \rrbracket_{\{x \leftarrow n\}}) \geq \mu_{\mathcal{C}}(S^n) \geq 1$, and we conclude that $\llbracket F_{\text{IMT}} \rrbracket_{\mathcal{E}} = \mathbb{B}^{\mathbb{N}}$.

²For the sake of readability, F has been written with a little abuse of notation the actual MQPA formula being $\forall x. \mathbf{C}^{1/z} (\text{EXP}(z, x) \wedge \forall y. (\exists w. (y + w = x) \rightarrow \text{FLIP}(y)))$, where $\text{EXP}(z, x)$ is an arithmetical formula expressing $z = 2^x$ and $\exists w. y + w = x$ expresses $y \leq x$.

The Random Walk Theorem. A second example we consider is the *random walk theorem* (RW): any simple random walk over \mathbb{Z} starting from 1 will pass through 1 infinitely many times with probability 1. More formally, any $\omega \in \mathbb{B}^{\mathbb{N}}$ induces a simple random walk starting from 1, by letting the n -th move be right if $\omega(n) = 1$ holds and left if $\omega(n) = 0$ holds. One has then:

Theorem 2 ([7], Thm. 8.3, p. 117) *Let $U_{ij}^{(n)} \subseteq \mathbb{B}^{\mathbb{N}}$ be the set of sequences for which the simple random walk starting from i leads to j in n steps. Then $\mu_{\mathcal{C}}\left(\bigcap_x \bigcup_{y \geq x} U_{11}^{(y)}\right) = 1$.*

Similarly, the random predicate $\text{FLIP}(n)$ induces a simple random walk starting from 1, by letting the n -th move be right if $\text{FLIP}(n)$ holds and left if $\neg\text{FLIP}(n)$ holds. To formalize RW in MQPA we make use two arithmetical formulas:

- $H(y, z)$ expresses that y is even and z is the code of a sequence of length $\frac{y}{2}$, such that for all $i, j < \frac{y}{2}$, $z_i < y$, and $z_i = z_j \Rightarrow i = j$ (that is, z codes a subset of $\{0, \dots, y-1\}$ of cardinality $\frac{y}{2}$);
- $K(y, z, v) = H(y, z) \wedge \exists i. i < \frac{y}{2} \wedge z_i = v$.

The formula of MQPA expressing RW is as follows:

$$F_{\text{RW}} : \mathbf{C}^{1/1} \forall x. \exists y. \exists z. y \geq x \wedge H(y, z) \wedge \forall v. \left(v < y \rightarrow (K(y, z, v) \leftrightarrow \text{FLIP}(v)) \right).$$

F_{RW} basically says that for any fixed x , we can find $y \geq x$ and a subset z of $\{0, \dots, y-1\}$ of cardinality $\frac{y}{2}$, containing all and only the values $v < y$ such that $\text{FLIP}(v)$ holds (so that the number of $v < y$ such that $\text{FLIP}(v)$ holds coincides with the number of $v < y$ such that $\neg\text{FLIP}(v)$ holds). This is the case precisely when the simple random walk goes back to 1 after exactly y steps.

To show the validity of F_{RW} we can use the measurable set $S = \bigcap_n \bigcup_{p \geq n} U_{11}^{(p)}$. Let $\psi(y, z, v)$ be the formula $(v < y \rightarrow (K(y, z, v) \leftrightarrow \text{FLIP}(v)))$. Observe that any sequence in $U_{11}^{(n)}$ satisfies the formula $\exists z. H(n, z) \wedge \forall v. \psi(y, z, v, w)$. Then, any sequence in S satisfies the formula $\forall x. \exists y. \exists z. y \geq x \wedge H(y, z) \wedge \forall v. \psi(y, z, v)$. Since, by Theorem 2, $\mu_{\mathcal{C}}(S) = 1$, we conclude that $\mu_{\mathcal{C}}(\llbracket F_{\text{RW}} \rrbracket_{\xi}) \geq \mu_{\mathcal{C}}(S) \geq 1$, and thus that $\llbracket F_{\text{RW}} \rrbracket_{\xi} = \mathbb{B}^{\mathbb{N}}$.

4 Arithmetization

It is a classical result in computability theory [23, 25, 26, 59, 61] that all computable functions are arithmetical, that is, for each partial recursive function $f : \mathbb{N}^m \rightarrow \mathbb{N}$ there is an arithmetical formula F_f , such that for every $n_1, \dots, n_m, l \in \mathbb{N}$: $f(n_1, \dots, n_m) = l \Leftrightarrow (\mathbb{N}, +, \times) \models F_f(n_1, \dots, n_m, l)$. In this section we show that, by considering arithmetical formulas of MQPA, this fundamental result can be generalized to computable *random* functions.

Computability in Presence of Probabilistic Choice. Although standard computational models are built around determinism, from the 1950s on, models for *randomized* computation started to receive wide attention [40, 14, 49, 53, 54, 21, 22, 57]. The first formal definitions of probabilistic Turing machines (for short, PTM) are due to Santos [53, 54] and Gill [22, 21]. Roughly, a PTM is an ordinary Turing machine (for short, TM) with the additional capability of making random decisions. Two alternative paradigms have been developed in the literature, the so-called *Markovian* and *oracle* PTMs. Here, we consider the definition by Gill, in which the probabilistic choices performed by the machines are binary and fair.

Definition 5 (Probabilistic Turing Machine [21, 22]) *A (one-tape) probabilistic Turing machine is a 5-tuple $(Q, \Sigma, \delta, q_0, Q_f)$, whose elements are defined as in a standard TM, except for the probabilistic transition function δ , which, given the current (non-final) state and symbol, specifies two equally-likely transition steps.*

As any ordinary TM computes a partial function on natural numbers, PTM can be seen as computing a so-called *random function* [53, pp. 706–707]. Let $\mathbb{D}(\mathbb{N})$ indicate the set of *pseudo-distributions* on \mathbb{N} , i.e. of functions $f : \mathbb{N} \rightarrow \mathbb{R}_{[0,1]}$, such that $\sum_{n \in \mathbb{N}} f(n) \leq 1$. Given a PTM, \mathcal{M} , a random function is a function $\langle \mathcal{M} \rangle : \mathbb{N} \rightarrow \mathbb{D}(\mathbb{N})$ which, for each natural number n , returns a pseudo-distribution supporting all the possible outcomes \mathcal{M} produces when fed with (an encoding of) n in input, each with its own probability.³ As expected, the random function $f : \mathbb{N} \rightarrow \mathbb{D}(\mathbb{N})$ is said to be *computable* when there is a PTM, \mathcal{M} , such that $\langle \mathcal{M} \rangle = f$.

Another widespread definition of TM is the one based on the notion of oracle. An *oracle TM* is a pair consisting of a standard deterministic TM and a random-bit oracle (for further details, see for example [16]). The random-bit oracle takes the form of an *oracle tape*, which is consulted whenever a coin-tossing state is encountered. The two definitions are assumed to be equivalent but, to the best of the authors’ knowledge, no formal proof of this equivalence has been presented in the literature yet.

Stating the Main Result. In order to generalize Gödel’s arithmetization of partial recursive functions to the class of computable random functions, we start by introducing the notion of arithmetical random function.

Definition 6 (Arithmetical Random Function) *A random function $f : \mathbb{N}^m \rightarrow \mathbb{D}(\mathbb{N})$ is said to be arithmetical if and only if there is a formula of MQPA, call it F_f , with free variables x_1, \dots, x_m, y , such that for every $n_1, \dots, n_m, l \in \mathbb{N}$, it holds that:*

$$\mu_{\mathcal{C}}(\llbracket F_f(n_1, \dots, n_m, l) \rrbracket) = f(n_1, \dots, n_m)(l). \quad (1)$$

The *arithmetization theorem* below relates random functions and the formulas of MQPA, and is the main result of this paper.

Theorem 3 *All computable random functions are arithmetical.*

Actually, we establish a stronger fact. Let us call a formula A of MQPA Σ_1^0 if A is equivalent to a formula of the form $\exists x_1 \dots \exists x_n. A'$, where A' contains neither first-order nor measure quantifiers. Then Theorem 3 can be strengthened by saying that any computable random function is represented (in the sense of Definition 6) by a Σ_1^0 -formula of MQPA.

Moreover, we are confident that a sort of converse of this fact can be established, namely that for any Σ_1^0 -formula $A(x_1, \dots, x_m)$ there exists a computable random relation $r(x_1, \dots, x_m)$ (i.e. a computable random function such that $r(x_1, \dots, x_m)(i) = 0$ whenever $i \neq 0, 1$) such that $\mu_{\mathcal{C}}(\llbracket A(n_1, \dots, n_m) \rrbracket) = r(n_1, \dots, n_m)(0)$ and $\mu_{\mathcal{C}}(\llbracket \neg A(n_1, \dots, n_m) \rrbracket) = r(n_1, \dots, n_m)(1)$. However, we leave this fact and, more generally, the exploration of an *arithmetical hierarchy* of randomized sets and relations, to future work.

Given the conceptual distance existing between TMs and arithmetic, a direct proof of Theorem 3 would be cumbersome. It is thus convenient to look for an alternative route.

On Function Algebras. In [11, 13], the class \mathcal{PR} of *probabilistic or random recursive functions* is defined as a generalization of Church and Kleene’s standard one [10, 32, 34, 35]. \mathcal{PR} is characterized as the smallest class of functions, which (i) contains some basic random functions, and (ii) is closed under composition, primitive recursion and minimization. For all this to make

³The seminal definition of *random function* already appeared in [53]:

DEFINITION. A k -ary random function ϕ is a function from E^{n+1} , the collection of all $(k+1)$ -tuples of nonnegative integers, to $[0,1]$ satisfying

$$\sum_{m=0}^{\infty} \phi(m_1, m_2, \dots, m_k, m) \leq 1$$

for every k -tuple (m_1, m_2, \dots, m_k) . [53, pp. 706–707]

Remarkably, in the same paper, Santos also delineated the notion of *probabilistic transition function*, on which the Markovian paradigm of PTM is based [53, p. 705] [54, p. 170].

sense, composition and primitive recursion are defined following the *monadic* structure of $\mathbb{D}(\cdot)$. In order to give a presentation as straightforward as possible, we preliminarily introduce the notion of *Kleisli extension* of a function with values in $\mathbb{D}(\mathbb{N})$.

Definition 7 (Kleisli Extension) *Given a function $f : \mathbb{N} \rightarrow \mathbb{D}(\mathbb{N})$, its (simple) Kleisli extension $f^K : \mathbb{D}(\mathbb{N}) \rightarrow \mathbb{D}(\mathbb{N})$ is defined as follows:*

$$f^K(d)(n) = \sum_{i \in \mathbb{N}} d(i) \cdot f(i)(n).$$

More in general, given a k -ary function $f : X_1 \times \dots \times X_{i-1} \times \mathbb{N} \times X_{i+1} \times \dots \times X_k \rightarrow \mathbb{D}(\mathbb{N})$, its i -th Kleisli extension $f_i^K : X_1 \times \dots \times X_{i-1} \times \mathbb{D}(\mathbb{N}) \times X_{i+1} \times \dots \times X_k \rightarrow \mathbb{D}(\mathbb{N})$ is defined as follows:

$$f_i^K(x_1, \dots, x_{i-1}, d, x_{i+1}, \dots, x_k)(n) = \sum_{j \in \mathbb{N}} d(j) \cdot f(x_1, \dots, x_{i-1}, j, x_{i+1}, \dots, x_k)(n).$$

The construction at the basis of the **K**-extension's general case can be applied more than once^[4]. Specifically, given a function $f : \mathbb{N}^k \rightarrow \mathbb{D}(\mathbb{N})$, its *total K*-extension $f^K : (\mathbb{D}(\mathbb{N}))^k \rightarrow \mathbb{D}(\mathbb{N})$ is defined as follows:

$$f^K(d_1, \dots, d_k)(n) = \sum_{i_1, \dots, i_k \in \mathbb{N}} f(i_1, \dots, i_k)(n) \cdot \prod_{1 \leq j \leq k} d_j(i_j).$$

We can now define the class \mathcal{PR} formally as follows:

Definition 8 (The Class \mathcal{PR} [11]) *The class of probabilistic recursive functions, \mathcal{PR} , is the smallest class of probabilistic functions containing:*

- The zero function, $z : \mathbb{N} \rightarrow \mathbb{D}(\mathbb{N})$, such that for every $x \in \mathbb{N}$, $z(x)(0) = 1$;
- The successor function, $s : \mathbb{N} \rightarrow \mathbb{D}(\mathbb{N})$, such that for every $x \in \mathbb{N}$, $s(x)(x+1) = 1$;
- The projection function, $\pi_m^n : \mathbb{N}^n \rightarrow \mathbb{D}(\mathbb{N})$, defined as for $1 \leq m \leq n$, $\pi_m^n(x_1, \dots, x_n)(x_m) = 1$;
- The fair coin function, $r : \mathbb{N} \rightarrow \mathbb{D}(\mathbb{N})$, defined as follows:

$$r(x)(y) = \begin{cases} \frac{1}{2} & \text{if } y = x \\ \frac{1}{2} & \text{if } y = x + 1 \\ 0 & \text{otherwise;} \end{cases}$$

and closed under:

- Probabilistic composition. *Given $f : \mathbb{N}^n \rightarrow \mathbb{D}(\mathbb{N})$ and $g_1, \dots, g_n : \mathbb{N}^k \rightarrow \mathbb{D}(\mathbb{N})$, their composition is a function $f \odot (g_1, \dots, g_n) : \mathbb{N}^k \rightarrow \mathbb{D}(\mathbb{N})$ defined as follows^[5]*

$$(f \odot (g_1, \dots, g_n))(\mathbf{x}) = f^K(g_1(\mathbf{x}), \dots, g_n(\mathbf{x}));$$

⁴For example, let us consider a binary function $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{D}(\mathbb{N})$, its *total K*-extension is as follows:

$$\begin{aligned} f_1^K(d_1, d_2)(y) &= \sum_{i_1 \in \mathbb{N}} d_1(i_1) \cdot f_2^K(i_1, d_2)(y) \\ &= \sum_{i_1 \in \mathbb{N}} d_1(i_1) \cdot \left(\sum_{i_2 \in \mathbb{N}} d_2(i_2) \cdot f(i_1, i_2)(y) \right) \\ &= \sum_{i_1, i_2 \in \mathbb{N}} f(i_1, i_2)(y) \cdot d_1(i_1) \cdot d_2(i_2) \\ &= \sum_{i_1, i_2 \in \mathbb{N}} f(i_1, i_2)(y) \cdot \prod_{k \in \{1, 2\}} d_k(i_k). \end{aligned}$$

⁵That is,

$$\begin{aligned} ((f \odot (g_1, \dots, g_n))(\mathbf{x}))(y) &= (f^K(g_1(\mathbf{x}), \dots, g_n(\mathbf{x}))(y)) \\ &= \sum_{i_1, \dots, i_n} f(i_1, \dots, i_n)(y) \cdot \prod_{1 \leq j \leq n} g_j(\mathbf{x})(i_j). \end{aligned}$$

The simplest case, is test of unary composition which is defined as follows, given $f : \mathbb{N} \rightarrow \mathbb{D}(\mathbb{N})$ and $g : \mathbb{N} \rightarrow \mathbb{D}(\mathbb{N})$, the function $h : \mathbb{N} \rightarrow \mathbb{D}(\mathbb{N})$ obtained by composition from f and g is:

$$(f \odot g)(x)(y) = f^K(g(x))(y)$$

- Probabilistic primitive recursion. Given $f : \mathbb{N}^k \rightarrow \mathbb{D}(\mathbb{N})$, and $g : \mathbb{N}^{k+2} \rightarrow \mathbb{D}(\mathbb{N})$, the function $h : \mathbb{N}^{k+1} \rightarrow \mathbb{D}(\mathbb{N})$ obtained from them by primitive recursion is as follows:

$$\begin{aligned} h(\mathbf{x}, 0) &= f(\mathbf{x}) \\ h(\mathbf{x}, y + 1) &= g_{k+2}^{\mathbf{K}}(\mathbf{x}, y, h(\mathbf{x}, y)); \end{aligned}$$

- Probabilistic minimization. Given $f : \mathbb{N}^{k+1} \rightarrow \mathbb{D}(\mathbb{N})$, the function $h : \mathbb{N}^k \rightarrow \mathbb{D}(\mathbb{N})$, obtained from it by minimization is defined as follows:

$$\mu f(\mathbf{x})(y) = f(\mathbf{x}, y)(0) \cdot \left(\prod_{z < y} \left(\sum_{k > 0} f(\mathbf{x}, z)(k) \right) \right).$$

Proposition 1 ([11]) \mathcal{PR} coincides with the class of computable random functions.

The class \mathcal{PR} is still conceptually far from MQPA. In fact, while the latter has access to randomness in the form of a global supply of random bits, the former can fire random choices locally through a dedicated initial function. To bridge the gap between the two, we introduce a third characterization of computable random functions, which is better-suited for our purposes. In doing so, we will define the class of oracle recursive functions, \mathcal{OR} . Our definition is loosely inspired from *oracle TM*, which can be defined as deterministic TM whose transition function can query a *random-bit tape* $\omega \in \mathbb{B}^{\mathbb{N}}$.

The class of *oracle recursive functions*, \mathcal{OR} , is the smallest class of partial functions of the form $f : \mathbb{N}^m \times \mathbb{B}^{\mathbb{N}} \rightarrow \mathbb{N}$, which (i) contains the class of *oracle basic functions*, and (ii) is closed under composition, primitive recursion, and minimization. Formally,

Definition 9 (The Class \mathcal{OR}) The class of oracle recursive functions, \mathcal{OR} , is the smallest class of probabilistic functions containing:

- The zero function, f_0 , such that $f_0(x_1, \dots, x_k, \omega) = 0$;
- The successor function, f_s , such that $f_s(x, \omega) = x + 1$;
- The projection function, f_{π_i} , such that, for $1 \leq i \leq k$, $f_{\pi_i}(x_1, \dots, x_k, \omega) = x_i$;
- The query function, f_q , such that $f_q(x, \omega) = \omega(x)$;

and closed under:

- Oracle composition. Given the oracle functions h from $\mathbb{N}^n \times \mathbb{B}^{\mathbb{N}}$ to \mathbb{N} and g_1, \dots, g_n (from \mathbb{N}^m), the function f obtained by composition from them, is defined as follows:

$$f(x_1, \dots, x_m, \omega) = h(g_1(x_1, \dots, x_m, \omega), \dots, g_n(x_1, \dots, x_m, \omega), \omega);$$

- Oracle primitive recursion. Given two oracle functions h and g , from respectively $\mathbb{N}^n \times \mathbb{B}^{\mathbb{N}}$ and $\mathbb{N}^{n+2} \times \mathbb{B}^{\mathbb{N}}$ to \mathbb{N} , the function f , obtained by primitive recursion from them, is defined as follows:

$$f(x, x_1, \dots, x_n) = \begin{cases} f(0, x_1, \dots, x_n, \omega) = h(x_1, \dots, x_n, \omega) \\ f(x + 1, x_1, \dots, x_n, \omega) = g(f(x, x_1, \dots, x_n, \omega), x, x_1, \dots, x_n, \omega); \end{cases}$$

- Oracle minimization. Given the oracle function g from $\mathbb{N}^{n+1} \times \mathbb{B}^{\mathbb{N}}$ to \mathbb{N} , the function f , obtained by minimization from g , is defined as follows:

$$f(x_1, \dots, x_n, \omega) = \mu x (g(x_1, \dots, x_n, x, \omega) = 0).$$

Remarkably, the only basic function depending on ω is the query function. All the closure schemes are independent from ω as well.

But in what sense do functions in \mathcal{OR} represent random functions? In order to clarify the relationship between \mathcal{OR} and \mathcal{PR} , we associate each \mathcal{OR} function with a corresponding auxiliary function.

Otherwise said,

$$((f \odot g)(x))(y) = \sum_{z \in \mathbb{N}} g(x)(z) \cdot f(z)(y).$$

Definition 10 (Auxiliary Function) Given an oracle function $f : \mathbb{N}^m \times \mathbb{B}^{\mathbb{N}} \rightarrow \mathbb{N}$, the corresponding auxiliary function, $f^* : \mathbb{N}^m \times \mathbb{N} \rightarrow \mathcal{P}(\mathbb{B}^{\mathbb{N}})$, is defined as follows: $f^*(x_1, \dots, x_m, y) = \{\omega \mid f(x_1, \dots, x_m, \omega) = y\}$.

The following lemma ensures that the value of f^* is always a measurable set:

Lemma 2 For every oracle recursive function $f \in \mathcal{OR}$, $f : \mathbb{N}^m \times \mathbb{B}^{\mathbb{N}} \rightarrow \mathbb{N}$, and natural numbers $x_1, \dots, x_m, y \in \mathbb{N}$, the set $f^*(x_1, \dots, x_m, y)$ is measurable.

Proof. We will show that, for each $f \in \mathcal{OR}$, $f^* \in \sigma(\mathcal{C})$, by induction on the structure of oracle recursive functions:

- Let $f \in \mathcal{OR}$ be an oracle basic function. There are four possible cases:

Zero Function. Let f_0 be the zero function, $f_0(x_1, \dots, x_n, \omega) = 0$. Then,

$$f_0^*(x_1, \dots, x_n, 0) = \{\omega \mid f_0(x_1, \dots, x_n, \omega) = 0\} = \mathbb{B}^{\mathbb{N}}.$$

For Axioms 1 of the σ -algebra, $\mathbb{B}^{\mathbb{N}} \in \sigma(\mathcal{C})$, so $f_0^* \in \sigma(\mathcal{C})$.

Successor Function. Let f_s be the successor function $f_s(x, \omega) = x + 1$. Then,

$$f_s^*(x, x + 1) = \{\omega \mid f_s(x, \omega) = x + 1\} = \mathbb{B}^{\mathbb{N}}.$$

As before, $f_s^* \in \sigma(\mathcal{C})$.

Projection Function. Let f_{π_i} be the projection function, $f_{\pi_i}(x_1, \dots, x_n, \omega) = x_i$, with $1 \leq i \leq n$. Then,

$$f_{\pi_i}^*(x_1, \dots, x_n, x_i) = \{\omega \mid f_{\pi_i}(x_1, \dots, x_n, \omega) = x_i\} = \mathbb{B}^{\mathbb{N}}.$$

Again, $f_{\pi_i}^* \in \sigma(\mathcal{C})$.

Query Function. Let f_q be the query function, $f_q(x, \omega') = \omega'(x)$. Then,

$$f_q^*(x, \omega') = \{\omega \mid f_q(x, \omega) = \omega'(x)\} = \{\omega \mid \omega(x) = 0\}$$

for $\omega'(x) = 0$ or $f_q^*(x, \omega') = \{\omega \mid \omega(x) = 1\}$ if $\omega'(x) = 1$, in both cases $f_q^*(x, \omega')$ is a (thin) cylinder, so $f_q^*(x, 0) \in \sigma(\mathcal{C})$. Therefore, $f_q^* \in \sigma(\mathcal{C})$.

- Let $f \in \mathcal{OR}$ be obtained by oracle composition, recursion or minimization from \mathcal{OR} functions. Since the three cases are proved in a similar way, let us take into account (simple) composition only. Let $f : \mathbb{N}^n \times \mathbb{B}^{\mathbb{N}} \rightarrow \mathbb{N}$ be obtained by (unary) composition from $h : \mathbb{N} \times \mathbb{B}^{\mathbb{N}} \rightarrow \mathbb{N}$ and $g : \mathbb{N}^n \times \mathbb{B}^{\mathbb{N}} \rightarrow \mathbb{N}$. Assume $f(x_1, \dots, x_n, \omega) = h(g(x_1, \dots, x_n, \omega), \omega) = v$. By Definition 10, $f^*(x_1, \dots, x_n, v) = \bigcup_{z \in \mathbb{N}} \{\omega \mid h(z, \omega) = v\} \cap \{\omega \mid g(x_1, \dots, x_n, \omega) = z\} = \bigcup_{z \in \mathbb{N}} h^*(z, v) \cap g^*(x_1, \dots, x_n, z)$. By IH, $h^*, g^* \in \sigma(\mathcal{C})$ so, by Axiom 3 of σ -algebras, $h^* \cap g^* \in \sigma(\mathcal{C})$ as well. Therefore, since f^* is a countable union of measurable sets, again for Axiom 3, it is measurable as well, so $f^* \in \sigma(\mathcal{C})$.

□

Thanks to Lemma 2, we can associate any oracle recursive function $f : \mathbb{N}^m \times \mathbb{B}^{\mathbb{N}} \rightarrow \mathbb{N}$ with a random function $f^\# : \mathbb{N}^m \rightarrow \mathbb{D}(\mathbb{N})$, defined as: $f^\#(x_1, \dots, x_m)(y) = \mu_{\mathcal{C}}(f^*(x_1, \dots, x_m, y))$. This defines a close correspondence between the classes \mathcal{PR} and \mathcal{OR} .

Proposition 2 For each $f \in \mathcal{PR}$, there is an oracle function $g \in \mathcal{OR}$, such that $f = g^\#$. Symmetrically, for any $f \in \mathcal{OR}$, there is a random function $g \in \mathcal{PR}$, such that $g = f^\#$.

For our purpose, only the first part of Proposition 2 is necessary, namely that for every $f \in \mathcal{PR}$, there is a $g \in \mathcal{OR}$ such that $g^\# = f$. This is established by means of a few intermediate steps.

First, some auxiliary notions are introduced. A *computable bijection* between \mathbb{N} and $\mathbb{N} \times \mathbb{N}$ and the corresponding maps $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ and $\pi_1, \pi_2 : \mathbb{N} \rightarrow \mathbb{N}$ is fixed. A *tree* is defined as a subset X of \mathbb{B}^* (the finite set of strings) such that if $t \in X$ and $v \sqsubset t$ (\sqsubset being the prefix relation), then $v \notin X$. Given $t \in \mathbb{B}^*$, $\omega \in \mathbb{B}^{\mathbb{N}}$ is said to be an *n-extension* of t if and only if $\omega = v \cdot t \cdot \omega'$, where $|v| = n$ and $\omega' \in \mathbb{B}^{\mathbb{N}}$. The set of all *n-extensions* of t is indicated as EXT_t^n and is measurable. Moreover, $\mu(\text{EXT}_t^n) = \frac{1}{2^{|v|}}$ for every n and t . Given a tree X , every function $f : X \rightarrow \mathbb{N}$ is said to be an *X-function*. Thus, an oracle function $f : \mathbb{N}^{n+1} \times \mathbb{B}^{\mathbb{N}} \rightarrow \mathbb{N} \in \mathcal{OR}$ returns a tree X and an *X-function* g on input (m_1, \dots, m_n) if and only if for every $k \in \mathbb{N}$, it holds that:

- $f(m_1, \dots, m_n, k, \omega)$ is defined if and only if $\omega \in \text{EXT}_t^k$, where $t \in X$.
- if $\omega \in \text{EXT}_t^k$ and $t \in X$, then

$$f(m_1, \dots, m_n, k, \omega) = q,$$

where $\pi_1(q) = g(t)$ and $\pi_2(q) = |t|$.

It is now possible to prove the following preliminary lemma.

Lemma 3 *For every $f : \mathbb{N}^n \rightarrow \mathbb{D}(\mathbb{N}) \in \mathcal{PR}$, there is an oracle recursive function $g : \mathbb{N}^{n+1} \times \mathbb{B}^{\mathbb{N}} \rightarrow \mathbb{N} \in \mathcal{OR}$, such that for every m_1, \dots, m_n , g returns a tree X_{m_1, \dots, m_n} and an X_{m_1, \dots, m_n} -function h_{m_1, \dots, m_n} on input m_1, \dots, m_n , and*

$$f(m_1, \dots, m_n)(y) = \sum_{h_{m_1, \dots, m_n}(t)=y} \frac{1}{2^{|t|}}.$$

Proof. The proof is by induction on the structure of f as an element of \mathcal{PR} .

- Let $f \in \mathcal{PR}$ be a basic probabilistic function. Then, there are four possible cases.

Zero Function. Let $z \in \mathcal{PR}$ be the zero function. Then, $g \in \mathcal{OR}$ is an oracle function, so defined that on inputs m_1, k, ω , it returns the value $\langle 0, 0 \rangle$. Indeed, g returns the tree $X_{m_1} = \{\epsilon\}$ and the X_{m_1} -function h_{m_1} always returning 0, as it can be easily checked. Moreover:

$$\begin{aligned} z(m_1)(y) &= \begin{cases} 1 & \text{if } y = 0 \\ 0 & \text{otherwise} \end{cases} \\ &= \sum_{h_{m_1}(s)=y} \frac{1}{2^{|s|}}. \end{aligned}$$

Successor Function. Let $s \in \mathcal{PR}$ be the successor function. Then, $g \in \mathcal{OR}$ is an oracle function, so defined that on inputs m_1, k, ω , it returns the value $\langle m_1 + 1, 0 \rangle$. So, g returns the tree $X_{m_1} = \{\epsilon\}$ and the X_{m_1} -function h_{m_1} always returning $m_1 + 1$. Therefore:

$$\begin{aligned} s(m_1)(y) &= \begin{cases} 1 & \text{if } y = m_1 + 1 \\ 0 & \text{otherwise} \end{cases} \\ &= \sum_{h_{m_1}(t)=y} \frac{1}{2^{|t|}}. \end{aligned}$$

Projection Function. Let $\pi_i^n \in \mathcal{PR}$, with $1 \leq i \leq n$, be the projection function. Then, $g \in \mathcal{OR}$ is an oracle function, so defined that, on inputs $m_1, \dots, m_n, k, \omega$ it returns the value $\langle m_i, 0 \rangle$. Indeed, g returns the tree $X_{m_1, \dots, m_n} = \{\epsilon\}$ and the X_{m_1, \dots, m_n} -function h_{m_1, \dots, m_n} always returning m_i . Moreover:

$$\begin{aligned} \pi_i^n(m_1, \dots, m_n)(y) &= \begin{cases} 1 & \text{if } y = m_i \\ 0 & \text{otherwise} \end{cases} \\ &= \sum_{h_{m_1, \dots, m_n}(t)=y} \frac{1}{2^{|t|}}. \end{aligned}$$

Fair Coin Function. Let $r \in \mathcal{PR}$ be the fair coin function. Then, $g \in \mathcal{OR}$ is an oracle function, so defined that $g(m_1, k, \omega) = \langle l, 1 \rangle$ where,

$$l = \begin{cases} m_1 & \text{if } \omega[k] = 0 \\ m_{1+1} & \text{if } \omega[k] = 1. \end{cases}$$

- If f is obtained by either composition, primitive recursion or minimization, the argument is a bit more involved, as in defining the function g we must take into account how the bits of the oracle accessed by g are distributed in an independent way to each of the component functions. We will here only illustrate how this works in the case of composition.

Let then f be obtained by composition from $f_1, \dots, f_p : \mathbb{N}^n \rightarrow \mathbb{D}(\mathbb{N})$ and $f' : \mathbb{N}^p \rightarrow \mathbb{D}(\mathbb{N})$, i.e. $f(m_1, \dots, m_n)(y) = \sum_{i_1, \dots, i_p} f'(i_1, \dots, i_p)(y) \cdot \prod_{j=1}^p f_j(m_1, \dots, m_n)(i_j)$. By induction hypothesis there exist functions $g_1, \dots, g_p : \mathbb{N}^{n+1} \times \mathbb{B}^{\mathbb{N}} \rightarrow \mathbb{N}$ and $g' : \mathbb{N}^{p+1} \times \mathbb{B}^{\mathbb{N}} \rightarrow \mathbb{N}$ such that

1. for all $m_1, \dots, m_n \in \mathbb{N}$, each g_i returns a tree X_{m_1, \dots, m_n}^i and an X_{m_1, \dots, m_n}^i -function h_{m_1, \dots, m_n}^i , and $f_i(m_1, \dots, m_n)(y) = \sum_{h_{m_1, \dots, m_n}^i(t)=y} \frac{1}{2^{|t|}}$;
2. for all $m_1, \dots, m_p \in \mathbb{N}$, g' returns a tree Y_{m_1, \dots, m_p} and a Y_{m_1, \dots, m_p} -function h'_{m_1, \dots, m_p} , and $f'(m_1, \dots, m_p)(y) = \sum_{h'_{m_1, \dots, m_p}(t)=y} \frac{1}{2^{|t|}}$.

We thus have that

$$\begin{aligned} f(m_1, \dots, m_n)(y) &= \sum_{i=1, \dots, i_p} \left(\sum_{h'_{i_1, \dots, i_p}(s)=y} \frac{1}{2^{|s|}} \right) \cdot \left(\prod_{j=1}^p \sum_{h_{m_1, \dots, m_n}^j(t)=i_j} \frac{1}{2^{|t|}} \right) \\ &= \sum_{i=1, \dots, i_p} \left(\sum_{h'_{i_1, \dots, i_p}(s)=y, h_{m_1, \dots, m_n}^j(t_j)=i_j} \frac{1}{2^{|s| + \sum_{j=1}^p |t_j|}} \right) \end{aligned}$$

For all $m_1, \dots, m_n \in \mathbb{N}$, let $X_{m_1, \dots, m_n} = \{t_1 \cdots t_p \mid s \in t_j \in X_{m_1, \dots, m_n}^j, s \in Y_{h_{m_1, \dots, m_n}^1(t_1), \dots, h_{m_1, \dots, m_n}^p(t_p)}\}$. Observe that any $v \in X_{m_1, \dots, m_n}$ can be decomposed in a unique way as $v = t_0 \cdot t_1 \cdots t_p$. In fact, if $t'_0 \cdot t'_1 \cdots t'_p$ is any other decomposition, let $j \leq p$ be minimum such that $t_j \neq t'_j$. Then it must be either $t_j \sqsubset t'_j$ or $t'_j \sqsubset t_j$, which contradicts the fact that X_{m_1, \dots, m_n}^j and $Y_{h_{m_1, \dots, m_n}^1(t_1), \dots, h_{m_1, \dots, m_n}^p(t_p)}$ are all trees.

Using this fact we can show that X_{m_1, \dots, m_n} is also a tree: suppose $v = t_0 \cdot t_1 \cdots t_p \in X_{m_1, \dots, m_n}$ and suppose $v' \in X_{m_1, \dots, m_n}$, where $v' \sqsubset v$. Then v' has a unique decomposition $t'_0 \cdots t'_p$ and one can easily show by induction on $j \leq p$ that $t'_j = t_j$ holds. Hence it must be $v' = v$, against the assumption.

Let $h_{m_1, \dots, m_n} : X_{m_1, \dots, m_n} \rightarrow \mathbb{N}$ be defined by $h_{m_1, \dots, m_n}(v) = y$, where v uniquely decomposes as $s \cdot t_1 \cdots t_p$, $h_{m_1, \dots, m_n}^j(t_j) = i_j$ and $h'_{i_1, \dots, i_p}(s) = y$. We can finally define:

$$g(m_1, \dots, m_n, k, \omega) = \left\langle \pi_1 \left(g'(\pi_1(L_1), \dots, \pi_1(L_p), k + \sum_{j=1}^p L_j, \omega) \right), \pi_2 \left(g'(\pi_1(L_1), \dots, \pi_1(L_p), k + \sum_{j=1}^p L_j, \omega) \right) + R_p \right\rangle$$

where the L_j, R_j are defined inductively as follows:

$$\begin{aligned} L_1 &= g_1(\vec{m}, k, \omega) & R_1 &= 0 \\ L_{j+1} &= g_{j+1}(\vec{m}, k + R_{j+1}, \omega) & R_{j+1} &= R_j + \pi_2(L_j) \end{aligned}$$

It can be checked that, by construction, $g(m_1, \dots, m_n, k, \omega) = \langle h_{m_1, \dots, m_n}(v), |v| \rangle$, where $v \in \text{EXT}_{v'}^k$ and v' uniquely decomposes as $s \cdot t_1 \cdots t_p$. Using the equations above we thus have:

$$f(m_1, \dots, m_n)(y) = \sum_{h_{m_1, \dots, m_n}(v)=y} \frac{1}{2^{|v|}}$$

□

The desired Proposition 2 can now be proved as a corollary of Lemma 3. Indeed, once it is observed that if g is obtained from $f \in$, as in Lemma 3 and $h(m_1, \dots, m_n, \omega) = \pi_1(g(m_1, \dots, m_n, 0, \omega))$, then it follows that $f = h^\#$.

The Proof of the Main Result. The last ingredient to establish Theorem 3 is the following lemma, easily proved by induction on the structure of \mathcal{OR} functions.

Lemma 4 *For every oracle function $f \in \mathcal{OR}$, the random function $f^\#$ is arithmetical.*

Proof. By Definition 10, given an arbitrary oracle function $f \in \mathcal{OR}$, $f(x_1, \dots, x_m, \omega) = y$, the corresponding $f^\#$ is defined as follows

$$f^\# = \mu_{\mathcal{C}}(f^*(x_1, \dots, x_m, y)) = \mu_{\mathcal{C}}(\{\omega \mid f(x_1, \dots, x_m, \omega) = y\}).$$

Lemma 4 states that such $f^\#$ is arithmetical, i.e. there is an MQPA formula $F_{f^\#}$, such that for every n_1, \dots, n_m, l ,

$$\mu_{\mathcal{C}}(\llbracket F_{f^\#}(n_1, \dots, n_m, l) \rrbracket) = f^\#(n_1, \dots, n_m)(l).$$

The proof of the Lemma is by induction on the structure of oracle recursive functions. Actually, the only case which is worth-considering is the one of *query* functions. Indeed, all the other cases are obtained by trivial generalizations of the standard proof by Gödel [23].

- For each *basic* oracle function, $f \in \mathcal{OR}$, the corresponding random function, $f^\#$, is arithmetical. There are four possible cases:

Oracle zero function. Let $f_0 \in \mathcal{OR}$ be the oracle zero function. For Definition 10, $f_0(x_1, \dots, x_n, \omega) = y$, with $y = 0$, is such that the corresponding $f_0^\#$ is defined by the MQPA formula 6

$$F_{f_0^\#} : y = 0.$$

Oracle successor function. Let $f_s \in \mathcal{OR}$ be the successor function. For Definition 10, $f_s(x, \omega) = y$, with $y = x+1$, is such that $f_s^\#$ is defined by the MQPA formula,

$$F_{f_s^\#} : S(x) = y.$$

Oracle projection function. Let $f_{\pi_i} \in \mathcal{OR}$ be the projection function Definition 10, $f_{\pi_i}(x_1, \dots, x_k, \omega) = y$, with $y = x_i$, be the is such that $f_{\pi_i}^\#$ is defined by the MQPA formula

$$F_{f_{\pi_i}^\#} : x_i = y.$$

Oracle query function. Let $q \in \mathcal{OR}$ be the query function. For Definition 10, $q(x, \omega) = \omega(x)$, is such that $f_q^\#$ is defined by the MQPA formula

$$F_q : x = \text{FLIP}(x).$$

- For each oracle function $f \in \mathcal{OR}$ obtained composition, primitive recursion, or minimization from oracle recursive functions, whose corresponding random functions are arithmetical, the corresponding random function $f^\#$ is arithmetical. The proofs for these three cases are very similar to the standard ones. As an example, let us consider the case of (simple) composition only.

Oracle (simple) composition. Let f be obtained by composition from h and g , i.e. $f(x_1, \dots, x_n, \omega) = h(g(x_1, \dots, x_n, \omega), \omega) = v$. It is possible show that $f^\#(x_1, \dots, x_n)(v)$ is arithmetical, which is, for every $x_1, \dots, x_n, v \in \mathbb{N}$, there is a MQPA formula, $F_{f^\#}(x_1, \dots, x_n, v)$, such that $\mu_{\mathcal{C}}(\llbracket F_{f^\#}(x_1, \dots, x_n, v) \rrbracket) = f^\#$. Indeed, the desired formula is

$$F_{f^\#} = \exists v (F_{h^\#}(v, y) \wedge F_{g^\#}(x_1, \dots, x_n, v))$$

where, by IH, $\mu_{\mathcal{C}}(\llbracket F_{h^\#} \rrbracket) = h^\#$ and $\mu_{\mathcal{C}}(\llbracket F_{g^\#} \rrbracket) = g^\#$. □

⁶Indeed, $f_0^\# = \mu_{\mathcal{C}}(\{\omega \mid f(x_1, \dots, x_m, \omega) = 0\}) = 1$ and $F_{f_0^\#} : y = 0$ and, since $y = 0$ as $\llbracket y \rrbracket = 0$ and $\llbracket 0 \rrbracket = 0$, $\mu_{\mathcal{C}}(F_{f_0^\#}) = 1$.

As seen, since for both \mathcal{OR} and MQPA the source of randomness consists in a denumerable amount of random bits, the proof of Lemma 4 is easy, and follows the standard induction of [23]. Theorem 3 comes out as a corollary of the Lemma 4 above, together with Proposition 1: any computable random function is in \mathcal{PR} , by Proposition 1, and each \mathcal{PR} function is arithmetical, by Lemma 4 and Proposition 2. Indeed, by Proposition 2 for any $f \in \mathcal{PR}$ there is a $g \in \mathcal{OR}$ such that $f = g^\#$ and, since $g \in \mathcal{OR}$, by Lemma 4, $g^\# (= f)$ is arithmetical.

5 Realizability

In this section we sketch an extension of realizability, a well-known computational interpretation of Peano Arithmetics, to MQPA. The theory of realizability [63, 24, 39, 61], which dates back to Kleene's 1945 paper [36], provides a strong connection between logic, computability, and programming language theory. The fundamental idea behind realizability is that from every proof of an arithmetical formula in HA or equivalently (via the Gödel-Gentzen translation) in PA, one can extract a program, called the *realizer* of the formula, which encodes the computational content of the proof. In Kreisel's *modified-realizability* [39] realizers are typed programs: any formula A of HA is associated with a type A^* and any proof of A yields a realizer of type A^* .

Our goal is to show that the modified-realizability interpretation of HA can be extended to the language MQPA. As we have not introduced a proof system for MQPA yet, we limit ourselves to establishing the soundness of modified-realizability with respect to the semantics of MQPA. Similarly to what happens with the class \mathcal{OR} , the fundamental intuition is that realizers correspond to programs which can query an *oracle* $\omega \in \mathbb{B}^\mathbb{N}$. For instance, a realizer of $\mathbf{C}^{t/s}A$ is a program which, for a randomly chosen oracle, yields a realizer of A with probability at least $\llbracket t \rrbracket_\xi / \llbracket s \rrbracket_\xi$.

Our starting point is a PCF-style language with oracles. The types of this language are generated by basic types nat , bool and the connectives \rightarrow and \times . We let $\mathbf{O} := \text{nat} \rightarrow \text{bool}$ indicate the type of *oracles*. For any type σ , we let $[\sigma]$ (resp. $[\sigma]_\mathbf{O}$) indicate the set of closed terms of type σ (resp. of terms of type σ with a unique free variable o of type \mathbf{O}). Moreover, for all $i \in \{0, 1\}$ (resp. $n \in \mathbb{N}$), we indicate as \bar{i} (resp. \bar{n}) the associated normal form of type bool (resp. nat). For all term M and normal form N , we let $M \Downarrow N$ indicate that M converges to N . For any term $M \in [\sigma_\mathbf{O}]$ and oracle $\omega \in \mathbb{B}^\mathbb{N}$, we let $M^\omega \in [\sigma]$ indicate the closed program in which any call to the variable o is answered by the oracle ω .

We consider the language of MQPA without negation and disjunction, enriched with implication $A \rightarrow B$. As is usually done in modified-realizability, we take $\neg A$ and $A \vee B$ as *defined* connectives, given by $A \rightarrow (\mathbf{0} = \mathbf{S}(\mathbf{0}))$ and $\exists x.(x = \mathbf{0} \rightarrow A) \wedge (x = \mathbf{S}(\mathbf{0}) \rightarrow B)$, respectively. With any closed formula A of MQPA we associate a type A^* defined as follows:

$$\begin{aligned} \text{FLIP}(t)^* &= \text{nat} & (\forall x.A)^* &= \text{nat} \rightarrow A^* \\ (t = u)^* &= \text{bool} & (\exists x.A)^* &= \text{nat} \times A^* \\ (A \wedge B)^* &= A^* \times B^* & (\mathbf{C}^{t/s}A)^* &= (\mathbf{D}^{t/s}A)^* = \mathbf{O} \rightarrow A^* \\ (A \rightarrow B)^* &= A^* \rightarrow B^* \end{aligned}$$

We define by induction the realizability relation $M, \omega \Vdash A$ where $\omega \in \mathbb{B}^\mathbb{N}$ and, if $A = \mathbf{C}^{t/s}B$ or $A = \mathbf{D}^{t/s}B$, $M \in [\sigma]$, and otherwise $M \in [\sigma]_\mathbf{O}$.

1. $M, \omega \Vdash \text{FLIP}(t)$ iff $\omega(\llbracket t \rrbracket) = 1$;
2. $M, \omega \Vdash t = s$ iff $\llbracket t \rrbracket = \llbracket s \rrbracket$;
3. $M, \omega \Vdash A_1 \wedge A_2$ iff $\pi_1(M), \omega \Vdash A_1$ and $\pi_2(M), \omega \Vdash A_2$;
4. $M, \omega \Vdash A \rightarrow B$ iff $\omega \in \llbracket A \rightarrow B \rrbracket$ and $P, \omega \Vdash A$ implies $MP, \omega \Vdash B$;
5. $M, \omega \Vdash \exists x.A$ iff $\pi_1(M^\omega) \Downarrow \bar{k}$ and $\pi_2(M), \omega \Vdash A(k/x)$;
6. $M, \omega \Vdash \forall x.A$ iff for all $k \in \mathbb{N}$, $M\bar{k}, \omega \Vdash A(k/x)$;
7. $M, \omega \Vdash \mathbf{C}^{t/s}A$ iff $\llbracket s \rrbracket > 0$ and $\mu_{\mathcal{C}}(\{\omega' \mid Mo, \omega' \Vdash A\}) \geq \llbracket t \rrbracket / \llbracket s \rrbracket$;
8. $M, \omega \Vdash \mathbf{D}^{t/s}A$ iff $\omega \in \llbracket \mathbf{D}^{t/s}A \rrbracket$, and $\llbracket s \rrbracket = 0$ or $\mu_{\mathcal{C}}(\{\omega' \mid Mo, \omega' \Vdash A\}) < \llbracket t \rrbracket / \llbracket s \rrbracket$.

Condition 7. is justified by the fact that for all term M and formula A , the set $\{\omega \mid M, \omega \Vdash A\}$ can be shown to be measurable. Conditions 5. and 9. include a semantic condition of the form $\omega \in \llbracket A \rrbracket$,

which has no computational meaning. This condition is added in view of Theorem 4 below. In fact, also in standard realizability a similar semantic condition for implication is required to show that realizable formulas are true in the standard model, see [63].

Lemma 5 *For each term $M \in [\sigma]_O$ and normal form $N \in [\sigma]$, the set $S_{M,N}$ of oracles ω such that $M^\omega \Downarrow N$ is measurable.*

Proof. Let $\widehat{M} = \{M^\omega \mid \omega \in \mathbb{B}^\mathbb{N}\}$. Let r be a reduction from some $P = M^\omega \in \widehat{M}$ to N ; since r is finite, it can only query finitely many values b_1, \dots, b_{n_q} of ω . Let C_r be the cylinder of all sequences which agree with ω at b_1, \dots, b_{n_q} . It is clear that for all $\omega' \in C_r$, $M^{\omega'}$ reduces to N . We have then that $S_{M,N} = \bigcup \{C_r \mid \exists P \in \widehat{M} \text{ and } r \text{ is a reduction from } P \text{ to } N\}$, so $S_{M,N}$ is a countable union of measurable sets, and it is thus measurable. \square

Lemma 6 *For each term M and closed formula A , the set $S_{M,A} = \{\omega \mid M, \omega \models A\}$ is measurable.*

Proof. By induction on the structure of A :

1. if $A = \text{FLIP}(t)$, then $S_{M,A}$ is the cylinder of all ω such that $\omega(\llbracket t \rrbracket) = 1$;
2. if $A = t = s$, then $S_{M,A} = \begin{cases} \mathbb{B}^\mathbb{N} & \text{if } \llbracket t \rrbracket = \llbracket s \rrbracket \\ \emptyset & \text{otherwise;} \end{cases}$
3. if $A = A_1 \wedge A_2$, then $S_{M,A} = S_{\pi_1(M), A_1} \cap S_{\pi_2(M), A_2}$;
4. if $A = B \rightarrow C$, then $S_{M,A} = \bigcap_{P \in [B^*]_O} (S_{P,B} \cap S_{MP,C})$;
5. if $A = \exists x.B$, then $S_{M,A} = \bigcup_k (S_{\pi_1(M), \bar{k}} \cap S_{\pi_2(M), A(k/x)})$;
6. if $A = \forall x.B$, then $S_{M,A} = \bigcap_k S_{M\bar{k}, A(k/x)}$;
7. if $A = \mathbf{C}^{t/s} B$, then $S_{M,A} = \begin{cases} \mathbb{B}^\mathbb{N} & \text{if } \llbracket s \rrbracket > 0 \text{ and } \mu_{\mathcal{C}}(S_{Mo,B}) \geq \llbracket t \rrbracket / \llbracket s \rrbracket \\ \emptyset & \text{otherwise;} \end{cases}$
8. if $A = \mathbf{D}^{t/s} B$, then $S_{M,A} = \begin{cases} \mathbb{B}^\mathbb{N} & \text{if } \llbracket s \rrbracket = 0 \text{ or } \mu_{\mathcal{C}}(S_{Mo,B}) < \llbracket t \rrbracket / \llbracket s \rrbracket \\ \emptyset & \text{otherwise.} \end{cases}$

\square

Theorem 4 (Soundness) *For a closed formula A , if $M, \omega \Vdash A$, then $\omega \in \llbracket A \rrbracket$.*

Proof. By induction on A :

1. if $A = \text{FLIP}(t)$ and $M, \omega \Vdash A$, then $\omega(\llbracket t \rrbracket) = 1$, so $\omega \in \llbracket \text{FLIP}(t) \rrbracket$;
2. if $A = t = s$ and $M, \omega \Vdash A$, then $\llbracket t \rrbracket = \llbracket s \rrbracket$, so $\omega \in \mathbb{B}^\mathbb{N} = \llbracket A \rrbracket$;
3. if $A = A_1 \wedge A_2$ and $M, \omega \Vdash A$, then from $\pi_1(M), \omega \models A_1$ and $\pi_2(M), \omega \models A_2$ we deduce $\omega \in \llbracket A_1 \rrbracket \cap \llbracket A_2 \rrbracket = \llbracket A \rrbracket$;
4. if $A = B \rightarrow C$ and $M, \omega \Vdash A$, then by definition $\omega \in \llbracket A \rrbracket$;
5. if $A = \exists x.B$ and $M, \omega \Vdash A$ then $\pi_1(M^\omega) \Downarrow \bar{k}$ and $\pi_2(M), \omega \Vdash B(k/x)$, so by IH $\omega \in \llbracket B(k/x) \rrbracket \subseteq \llbracket A \rrbracket$;
6. if $A = \forall x.B$ and $M, \omega \Vdash A$ then for all $k \in \mathbb{N}$, $M\bar{k}, \omega \Vdash A(k/x)$, so by IH $\omega \in \llbracket A(k/x) \rrbracket$ and we conclude $\omega \in \llbracket A \rrbracket = \bigcap_k \llbracket A(k/x) \rrbracket$;
7. if $A = \mathbf{C}^{t/s} B$ and $M, \omega \Vdash A$ then $\llbracket s \rrbracket > 0$ and $\mu_{\mathcal{C}}(S) \geq \llbracket t \rrbracket / \llbracket s \rrbracket$, where $S = \{\omega' \mid Mo, \omega' \Vdash B\}$. Then, by IH $S \subseteq \llbracket B \rrbracket$, hence $\mu_{\mathcal{C}}(\llbracket B \rrbracket) \geq \mu_{\mathcal{C}}(S) \geq \llbracket t \rrbracket / \llbracket s \rrbracket$. We conclude then that $\llbracket A \rrbracket = \mathbb{B}^\mathbb{N}$ and thus $\omega \in \llbracket A \rrbracket$;
8. if $A = \mathbf{D}^{t/s} B$ and $M, \omega \Vdash A$ then by definition $\omega \in \llbracket \mathbf{D}^{t/s} B \rrbracket$.

\square

For example, the term $M = \lambda o. \text{fix}(\lambda f x. (\text{iszero}(ox))(f(x+1))\langle x, x \rangle) \bar{0}$ realizes the valid formula $\mathbf{C}^{1/1} \exists x. \text{FLIP}(x)$. M looks for the first value k such that $o(k) = 1$ and returns the pair $\langle \bar{k}, \bar{k} \rangle$. Similarly, the program $\lambda x o y z. o(y)$, which checks whether the y -th bit of ω is true, realizes the formula $\forall x. \mathbf{C}^{1/2^x} \forall_{y \leq x} \text{FLIP}(y)$. With the same intuition, one can imagine how a realizer M of the formula F_{IMT} can be constructed: given inputs x, o, y , M looks for the first k such that the finite sequence $o(y+k), o(y+k+1), \dots, o(y+k+\ell(n))$ coincides with the string coded by x (where this last check can be encoded by a program $\lambda w. P(x, o, y, z, w)$), and returns the pair $\langle \bar{k}, \lambda w. P(x, 0, y, \bar{k}, w) \rangle$.

6 Related Works

To the best of the authors’ knowledge, the term “measure quantifier” was first introduced by Morgenstern in 1979 in order to formalize the idea that a formula $F(x)$ is true *for almost all* x [45].⁷ In the same years, similar quantifiers were investigated from a model-theoretic perspective by H. Friedman (see [62] for a survey). More recently, Mio et al. [43, 44] investigated the possibility for such quantifiers to define extensions of MSO. Generally speaking, all these works have been strongly inspired by the notion of generalized quantifiers, which already appeared in a seminal work by Mostowski [46].⁸ Nevertheless, the main source of inspiration for our treatment of measure quantifiers comes from computational complexity, namely from Wagner’s counting operators on classes of languages [65, 66, 67].⁹

On the other hand, there is an extensive amount of publications dealing with different forms of probabilistic reasoning (without references to arithmetic). Most of the recent probability systems have been developed in the realm of modal logic, starting with the seminal (propositional) work by Nilsson [47]. From the 1990s on, first-order probability logic and (axiomatic) proof systems, have been independently introduced by Bacchus [6, 4, 5] and Fagin, Halpern and Megiddo [19, 18, 27, 28]. Remarkably, Bacchus defined *probability terms*, by means of a modal operator **prob** computing the probability of certain events, and *probability formulas*, which are equalities between probability terms and numbers. A similar first-order probability logic was introduced by Fagin, Halpern and Megiddo [19] (and later studied in [18, 27, 28]), in which probability spaces define the underlying models, and can be accessed through the so-called weight terms. Another class of probabilistic modal logics have been designed to model Markov chains and similar structure, for example in [38, 29, 41, 20]. However, once again, no reference to arithmetic is present in these works.

From the 1950s on, the interest for probabilistic algorithms and models started spreading [40, 14, 9, 49, 51, 21, 56]. Nowadays, random computation is pervasive in many area of computer science and, several formal models are available, such as probabilistic automata [55], both Markovian and oracle probabilistic Turing machines [53, 54, 21, 22], and probabilistic λ -calculi [50, 31, 15, 12, 17]. As seen, also a well-defined probabilistic recursion theory has been developed by [11, 13]. Our definition of the class \mathcal{OR} is guided by both by recent \mathcal{PR} [11, 13] and by classical recursion theory [23, 10, 32, 33, 34, 35, 64, 48].¹⁰ Our definition of random arithmetical formulas and Theorem 3 generalize the original results by Gödel [26, pp. 63–65]. Also our study of realizability is inspired by classical works. The functional or D-interpretation was first introduced by Gödel in 1958 in order to prove the consistency of arithmetic [24].¹¹ The theory was further developed by Kreisel, who introduced the notion of modified-realizability [39], starting from Kleene’s realizability [36].

⁷Morgenstern’s definition was inspired by the notion of generalized quantifier, which was “introduced to specify that a given formula was true for “many x ’s”” [45, p. 103]. Morgenstern defined a language L_μ , obtained by adding the measure quantifier Q_μ to the standard first-order grammar and presented the central notions of his logic as follows (actually, other extended languages are considered in [45]):

DEFINITION 2.1 A *measure structure* \mathcal{U} is a pair $\mathcal{U} = (\mathcal{U}, \mu^\mathcal{U})$, where \mathcal{U}' is a first-order structure, $\text{card}|\mathcal{U}| = k$, a measurable cardinal, and $\mu^\mathcal{U}$ is a nontrivial k -additive measure on $|\mathcal{U}'|$ which satisfies the partition property. [...]

DEFINITION 2.2 Define a language L_μ to be a first-order language together with a quantifier Q_μ , binding one free variable, where a measure $\mathcal{U} \models Q_\mu v_0 \varphi(v_0)$ iff $\{x \in |\mathcal{U}| \mid \mathcal{U}' \models \varphi[x]\} \in \mu^\mathcal{U}$. [45, pp. 103-104]

⁸Specifically, generalized quantifiers were first introduced by Mostowski, as “operators which represent a natural generalization of the logical quantifiers” [46, p. 13] and have then been extensively studied in the context of finite-model theory [42, 37]. Second-order generalized quantifiers have been recently defined as well [1].

⁹For further details, see [2], where the model theory and proof theory of an extension of propositional logic with counting quantifiers is studied (in particular, the logic CPL_0 can be seen as a “finitary” fragment of MQPA).

¹⁰For further details on the history of the notion of recursion, see [60].

¹¹Actually, Gödel started conceiving the D-interpretation in the late 1930s [60, 3].

7 Conclusion

This paper can be seen as “a first exploration” of MQPA, providing some preliminary results, but also leaving many problems and challenges open. The most compelling one is certainly that of defining a proof system for MQPA, perhaps inspired from realizability. Furthermore, our extension of PA is *minimal* by design. In particular, we confined our presentation to a unique predicate variable, $\text{FLIP}(x)$. Yet, it is possible to consider a more general language with countably many predicate variables $\text{FLIP}_a(x)$, and suitably-*named* quantifiers $\mathbf{C}_a^{t/s}$ and $\mathbf{D}_a^{t/s}$ (as in [2]). We leave the exploration of this more sophisticated syntax to future work. Another intriguing line of work concerns the study of bounded versions of MQPA, which may suggest novel ways of capturing probabilistic complexity classes, different from those in the literature, e.g. [30].

References

- [1] Andersson, A.: On second-order generalized quantifiers and finite structures. *Annals of Pure and Applied Logic* **115**(1–3), 1–32 (2002)
- [2] Antonelli, M., Dal Lago, U., Pistone, P.: On counting propositional logic (2021), available from: <https://arxiv.org/abs/2103.12862>
- [3] Avigad, J., Feferman, S.: Gödel’s functional (“Dialectica”) interpretation. In: Buss, S. (ed.) *Handbook of Proof Theory*, vol. 137, pp. 337–405. Elsevier Science (1998)
- [4] Bacchus, F.: On probability distributions over possible worlds. *Machine Intelligence and Pattern Recognition* **9**, 217–226 (1990)
- [5] Bacchus, F.: *Representing and Reasoning with Probabilistic Knowledge*. MIT Press (1990)
- [6] Bacchus, F.: Lp, a logic for representing and reasoning with statistical knowledge. *Comput. Intell.* **6**(4), 209–231 (1990)
- [7] Billingsley, P.: *Probability and Measure*. Wiley (1995)
- [8] Buss, S.: *Bounded Arithmetic*. Ph.D. thesis, Princeton University (1986)
- [9] Carlyle, J.: Reduced forms for stochastic sequential machines. *J. Math. Anal. Appl.* **7**, 167–174 (1963)
- [10] Church, A., Kleene, S.: Formal definitions in the theory of ordinal numbers. *Fund. Math.* **28**, 11–21 (1936)
- [11] Dal Lago, U., Gabbrielli, M., Zuppiroli, S.: Probabilistic recursion theory and implicit computational complexity. *Sci. Ann. Comput. Sci.* **24**(2), 177–216 (2014)
- [12] Dal Lago, U., Zorzi, M.: Probabilistic operational semantics for the lambda calculus. *RAIRO* **46**(3), 413–450 (2012)
- [13] Dal Lago, U., Zuppiroli, S.: Probabilistic recursion theory and implicit computational complexity. In: *ICTAC. Lecture Notes in Computer Science*, vol. 8687, pp. 97–114 (2014)
- [14] Davis, A.: Markov chains as random input automata. *Am. Math. Mon.* **68**(3), 264–267 (1961)
- [15] Di Pierro, A., Wiklicky, H.: Probabilistic lambda-calculus and quantitative program analysis. *J. Log. Comput.* **15**(2), 159–179 (2005)
- [16] Du, D.Z., Ko, K.I.: *Computational Complexity: A modern approach*. Wiley (2000)
- [17] Ethard, T., Pagani, M., Tasso, C.: The computational meaning of probabilistic coherence spaces. In: *IEEE (ed.) LICS*. pp. 87–96. Toronto, ON, Canada (2011)

- [18] Fagin, R., Halpern, J.: Reasoning about knowledge and probability. J. of ACM **41**(2), 340–367 (1994)
- [19] Fagin, R., Halpern, J., Megiddo, N.: A logic for reasoning about probabilities. Inf. Comput. **87**(1/2), 78–128 (1990)
- [20] Furber, R., Mardare, R., Mio, M.: Probabilistic logics based on Riesz spaces. LMCS **16**(1) (2020)
- [21] Gill, J.: Computational complexity of probabilistic Turing machines. In: STOC. pp. 91–95. ACM (1974)
- [22] Gill, J.: Computational complexity of probabilistic Turing machines. J. Comput. **6**(4), 675–695 (1977)
- [23] Gödel, K.: Über formal unentscheidbare sätze der *Principia Mathematica* und verwandter systeme. Monatsch. Math. Phys. **38**, 173–178 (1931)
- [24] Gödel, K.: Über eine bisher noch nicht benützte Erweiterung des finiten standpunktes. Dialectica **12**, 280–287 (1958)
- [25] Gödel, K.: On undecidable propositions of formal mathematical systems. In: Davis, M. (ed.) The Undecidable: Basic Papers on Undecidable Propositions, Unsolvability Problems and Computable Functions. Dover Publications (1965)
- [26] Gödel, K.: On Formally Undecidable Propositions of *Principia Mathematica* and Related Systems. Dover Publications (1992)
- [27] Halpern, J.: An analysis of first-order logics for probability. Artif. Intell. **46**(3), 311–350 (1990)
- [28] Halpern, J.: Reasoning About Uncertainty. MIT Press (2003)
- [29] Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Form. Asp. Comput. **6**(5), 512–535 (1994)
- [30] Jerábek, E.: Approximate counting in bounded arithmetic. J. Symb. Log. **72**(3), 959–993 (2007)
- [31] Jones, C., Plotkin, G.: A probabilistic powerdomain of evaluations. In: LICS. vol. 4, pp. 186–195. IEEE (1989)
- [32] Kleene, S.: General recursive functions of natural numbers. Math. Ann. **112**, 727–742 (1936)
- [33] Kleene, S.: λ -definability and recursiveness. Duke Math. J. **2**, 340–353 (1936)
- [34] Kleene, S.: A note on recursive functions. Bul. of AMS **42**, 544–546 (1936)
- [35] Kleene, S.: Recursive predicates and quantifiers. Trans. of AMS **53**, 41–73 (1943)
- [36] Kleene, S.: On the interpretation of intuitionistic number theory. J. Symb. Log. **10**(4), 109–124 (1945)
- [37] Kontinen, J.: A logical characterization of the counting hierarchy. ACM **10**(1) (2009)
- [38] Kozen, D.: Semantics of probabilistic programs. JCSS **22**(3), 328–350 (1981)
- [39] Kreisel, G.: Gödel’s interpretation of Heyting’s arithmetic. In: Summaries of talks, Summer Institute for Symbolic Logic (1957)
- [40] de Leeuw, e.a.: Computability by probabilistic machines. In: Press, P.U. (ed.) Automata Studies, pp. 183–212. No. 34, Shannon, C.E. and McCarthy, J. (1956)

- [41] Lehmann, D., Shelah, S.: Reasoning with time and chance. *Inf. Control* **53**(3), 165 – 198 (1982)
- [42] Lindström, P.: First order predicate logic with generalized quantifiers. *Theoria* **32**, 186–195 (1966)
- [43] Michalewski, H., Mio, M.: Measure quantifiers in monadic second order logic. *LFCS* pp. 267–282 (2016)
- [44] Mio, M., Skrzypczak, M., Michalewski, H.: Monadic second order logic with measure and category quantifiers. *LMCS* **8**(2) (2012)
- [45] Morgenstern, C.: The measure quantifier. *J. Symb. Log.* **44**(1) (1979)
- [46] Mostowski, A.: On a generalization of quantifiers. *Fundamenta Mathematicae* **44**, 12–36 (1957)
- [47] Nilsson, N.: Probabilistic logic. *Artif. Intell.* **28**(1), 71–87 (1986)
- [48] Peter, R.: *Rekursive Funktionen*. Akadémiai Kiadó (1951)
- [49] Rabin, M.O.: Probabilistic automata. *Inf. Comput.* **6**(3), 230–245 (1963)
- [50] Saheb-Djaroni, N.: Probabilistic LCF. In: *MFCS*. pp. 442–452. No. 64 in
- [51] Santos, E.: Maximin automata. *Inf. Control* **13**, 363–377 (1968)
- [52] Santos, E.: Maximin sequential-like machines and chains. *Math. Syst. Theory* **3**(4), 300–309 (1969)
- [53] Santos, E.: Probabilistic Turing machines and computability. *AMS* **22**(3), 704–710 (1969)
- [54] Santos, E.: Computability by probabilistic Turing machines. *AMS* **159**, 165–184 (1971)
- [55] Segala, R.: A compositional trace-based semantics for probabilistic automata. In: *CONCUR*. pp. 234–248 (1995)
- [56] Simon, J.: On some central problems in computational complexity. Ph.D. thesis, Cornell University (1975)
- [57] Simon, J.: On tape-bounded probabilistic Turing machine acceptors. *TCS* **16**, 75–91 (1981)
- [58] Simpson, S.: *Subsystems of Second Order Arithmetic*. Cambridge Press (2009)
- [59] Smith, P.: *An Introduction to Gödel’s Theorems*. Cambridge University Press (2013)
- [60] Soare, R.: Computability and recursion. *Bull. Symb. Log.* **2**, 284–321 (1996)
- [61] Sorensen, M., Urzyczyn, P.: *Lectures on the Curry-Howard Isomorphism*. Elsevier (2006)
- [62] Steinhorn, C.I.: *Borel Structures and Measure and Category Logics*, vol. 8, pp. 579–596. Springer-Verlag (1985)
- [63] Troelstra, A.: Realizability. In: Buss, S.R. (ed.) *Handbook of Proof Theory*, vol. 137, pp. 407–473. Elsevier (1998)
- [64] Turing, A.: Computability and λ -definability. *J. Symb. Log.* **2**, 153–163 (1937)
- [65] Wagner, K.: Compact descriptions and the counting polynomial-time hierarchy. In: *Frege Conference 1984*. pp. 383–392 (1984)
- [66] Wagner, K.: The complexity of combinatorial problems with succinct input representation. *Acta Informatica* **23**, 325–356 (1986)
- [67] Wagner, K.: Some observations on the connection between counting and recursion. *TCS* **47**, 131–147 (1986)