Universal equivalence and majority of probabilistic programs over finite fields

Gilles Barthe, Charlie Jacomme, Steve Kremer

Universal equivalence and majority of probabilistic programs over finite fields

Gilles Barthe[1], Charlie Jacomme[2], and Steve Kremer[3]

[1]MPI-SP & IMDEA Software Institute
[2]LSV, CNRS & ENS Paris-Saclay & Inria & Université Paris-Saclay
[3]LORIA, Inria Nancy-Grand Est & CNRS & Université de Lorraine

November 13, 2020

Abstract

We study decidability problems for equivalence of probabilistic programs, for a core probabilistic programming language over finite fields of fixed characteristic. The programming language supports uniform sampling, addition, multiplication and conditionals and thus is sufficiently expressive to encode boolean and arithmetic circuits. We consider two variants of equivalence: the first one considers an interpretation over the finite field $\mathbb{F}_q$, while the second one, which we call universal equivalence, verifies equivalence over all extensions $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$. The universal variant typically arises in provable cryptography when one wishes to prove equivalence for any length of bitstrings, i.e., elements of $\mathbb{F}_{2^k}$ for any k. While the first problem is obviously decidable, we establish its exact complexity which lies in the counting hierarchy. To show decidability, and a doubly exponential upper bound, of the universal variant we rely on results from algorithmic number theory and the possibility to compare local zeta functions associated to given polynomials. Finally we study several variants of the equivalence problem, including a problem we call majority, motivated by differential privacy.

Contents

1 Introduction

Program equivalence is one of the most fundamental tools in the theory of programming languages and arguably the most important example of relational property. Program equivalence has been studied extensively, leading to numerous decidability results and sound proof methods. This paper is concerned with the decidability of equivalence and relational properties for a core imperative probabilistic programming language. Like many other probabilistic programming languages, our language supports sampling from distributions, and conditioning distributions on an event. The specificity of our language is that it operates over finite fields of the form $\mathbb{F}_{q^k}$. Therefore, expressions are interpreted as polynomials and assertions are boolean combinations of polynomial identities. Sampling is interpreted using the uniform distributions over sets defined by assertions, and branching and conditioning are relative to assertions.

We consider two relational properties, equivalence and majority, which we define below, and several related properties, which we explain in the next paragraph. For each property, we consider two variants of the problem. In the first variant, which we call the fixed case, the value of k is fixed. In the second variant, which we call the universal variant, we require the property to hold for all possible values of k. Consider two programs P_1 and P_2 with m inputs and n outputs. These programs are interpreted as functions $[[P_1]]^{q^k}, [[P_2]]^{q^k} : \mathbb{F}_{q^k}^m \to \mathsf{Distr}(\mathbb{F}_{q^k}^n)$.

- q^k*-equivalence* (denoted $P_1 \approx_{q^k} P_2$) requires that P_1 and P_2 define the same distributions: $[[P_1]]^{q^k} = [[P_2]]^{q^k}$. Equivalently, for every input $\vec{a} \in \mathbb{F}_{q^k}^m$ and output $\vec{b} \in \mathbb{F}_{q^k}^n$,

  $$\mathbb{P}\{\vec{x} = \vec{b} \mid \vec{x} \xleftarrow{\$} [[P_1]]^{q^k}(\vec{a})\} = \mathbb{P}\{x = \vec{b} \mid \vec{x} \xleftarrow{\$} [[P_2]]^{q^k}(\vec{a})\}.$$

  q^∞-equivalence requires the property to hold on all extensions of a field, i.e.,

  $$P_1 \approx_{q^\infty} P_2 \text{ iff } \forall k.\ P_1 \approx_{q^k} P_2$$

- q^k*-majority* requires that for a fixed $r \in \mathbb{Q}$, and for every input $\vec{a} \in \mathbb{F}_{q^k}^m$ and output $\vec{b} \in \mathbb{F}_{q^k}^n$, we have

  $$\mathbb{P}\{\vec{x} = \vec{b} \mid \vec{x} \xleftarrow{\$} [[P_1]]^{q^k}(\vec{a})\} \leq r \cdot \mathbb{P}\{\vec{x} = \vec{b} \mid \vec{x} \xleftarrow{\$} [[P_2]]^{q^k}(\vec{a})\}.$$

  q^k*-0-majority* (denoted $P_1 \prec^r_{q^k} P_2$) is a variant of majority, where we only consider the output $b = 0^n$, rather than quantifying over all outputs. q^∞*-0-majority* requires the property to hold on all extensions of a field, i.e.,

  $$P_1 \prec^r_{q^\infty} P_2 \text{ iff } \forall k.\ P_1 \prec^r_{q^k} P_2$$

The following two boolean programs illustrate the difference between equivalence and universal equivalence.

Example 1.

$$P_1 = x \xleftarrow{\$} \mathbb{F};\ \mathsf{return}\ (x^2 + x) \qquad\qquad P_2 = \mathsf{return}\ 0$$

are 2- but not 2^2-*equivalent*, and hence not 2^∞-equivalent. Indeed, when instantiating $\mathbb{F}$ with $\mathbb{F}_2$, the left hand side program simply evaluates to zero, which is not the case with $\mathbb{F}_4$. On the other hand, the programs

$$Q_1 = x \xleftarrow{\$} \mathbb{F};\ \mathsf{return}\ (x) \qquad\qquad Q_2 = x \xleftarrow{\$} \mathbb{F};\ \mathsf{return}\ (x + 1)$$

are q^∞-equivalent as both programs define the uniform distribution over $\mathbb{F}$, whatever finite field is used for the interpretation of $\mathbb{F}$. These examples also illustrate the difference with the well-studied polynomial identity testing (PIT) problem, as the first two programs are 2-equivalent, while PIT does not consider $x^2 + x$ and 0 to be equal on $\mathbb{F}_2$, nor would Q_1 and Q_2 be considered identical.

The fixed and universal variants of the equivalence and majority problems are directly inspired from applications in security and privacy. In the fixed setting, the equivalence and majority problems are related to probabilistic non-interference and differential privacy. The relationships between probabilistic non-interference and equivalence and between differential privacy and majority are explained informally as follows:

- probabilistic non-interference: for simplicity, assume that P has two inputs x (secret) and y (public), and a single (public) output. For every x, let P_x be the unique program such that $P_x(y) = P(x, y)$. Then P is non-interfering iff for every x_1 and x_2, the two programs P_{x_1} and P_{x_2} are equivalent.
- differential privacy: for simplicity consider the case where the base field is $\mathbb{F}_2$. For every program P with n inputs, define the residual programs $P_{i,0}$ and $P_{i,1}$ obtained by fixing the i-th output to 0 and 1 respectively. Then the program P is $log(r)$-differentially private iff for every i, $P_{i,0}$ and $P_{i,1}$ (and $P_{i,1}$ and $P_{i,0}$) satisfy r-majority.

In the universal setting, the parameter k can loosely be understood as the security parameter. Universal equivalence is a special case of statistical indistinguishability and as such arises naturally in provable security, where the goal is to prove (depending on applications either as end goal, or as an intermediate goal) that two programs are equivalent for all possible interpretations (e.g. for all possible lengths of bitstrings, i.e. for all $\mathbb{F}_{2^k}$).

Summary of results

We also consider the following problems, which are also motivated by security and privacy and are directly related to equivalence:

- (bounded) simulatability: given programs P_1 and P_2, does there exist a context $C[\cdot]$ (of bounded degree) such that $C[P_1]$ is equivalent to P_2;
- independence: are outputs Y and Y' of program P independent conditioned on Z, i.e. for every input x, is the distribution of Y independent from the distribution of Y', when conditioning on the value of Z? Although independence is not naturally expressed as a relational property, it has been shown in [4] that relational methods are useful for proving independence.

The first contribution of the paper is a systematic study of the complexity of the aforementioned problems in the fixed setting. We prove that the q^k-*equivalence* problem is $\mathsf{coNP}^{\mathsf{C_=P}}$-complete for any fixed k. We also study the special case of *linear* programs, i.e. multiplication, conditional and conditioning free, for which the problem can be decided in polynomial time. For the majority problem, we consider two settings: programs with and without inputs. We show that the k-majority problem for inputless programs is PP-complete, whereas the k-majority for arbitrary programs is $\mathsf{coNP}^{\mathsf{PP}}$-complete—thus the second problem is strictly harder than the first, unless $\mathsf{PH} \subset \mathsf{PP}$[1]. The proofs are given by reductions to MAJSAT and E−MAJSAT respectively. Note that we do not include any result about bounded simulatability in the finite case, since we only derive easy consequences of equivalence. These results complement recent work on the complexity of checking differential privacy for arithmetic circuits [14], see Related Work below.

The second, and main contribution, is the study of universal equivalence, q^∞-equivalence for short, and universal (0-)majority, q^∞-(0-)majority for short. First, we show that the q^∞-*equivalence* problem is in 2-EXP and $\mathsf{coNP}^{\mathsf{C_=P}}$-hard.

Our proof is based on local zeta Riemann functions, a powerful tool from algebraic geometry, that characterize the number of zeros of a tuple of polynomials in all extensions of a finite field. Lauder and Wan [18] notably propose an algorithm to compute such functions, whose complexity is however exponential. Based on this result, our proof proceeds in three steps.

First, we give a reduction for arithmetic programs (no conditionals, nor conditioning) from universal equivalence to checking that some specific local zeta Riemann functions are always null. Then, we reduce the general case to programs without conditioning, and programs without conditioning to arithmetic programs. To justify the use of the local zeta Riemman functions, we also provide counterexamples why

[1] As $\mathsf{PH} \subset \mathsf{coNP}^{\mathsf{PP}}$, $\mathsf{PP} = \mathsf{coNP}^{\mathsf{PP}}$ would imply $\mathsf{PH} \subset \mathsf{PP}$ which is commonly believed to be false.

	x-(*conditional*)-{*equivalence*, *independence*, *uniformity*}			q^∞-*simulatability*
	linear	arithmetic	general	decidable $\mathsf{coNP}^{\mathsf{C_=P}}$-hard
$x = q^k$	PTIME	$\mathsf{coNP}^{\mathsf{C_=P}}$-complete	$\mathsf{coNP}^{\mathsf{C_=P}}$-complete	
$x = q^\infty$	PTIME	EXP $\mathsf{coNP}^{\mathsf{C_=P}}$-hard	2-EXP $\mathsf{coNP}^{\mathsf{C_=P}}$-hard	

Figure 1: Summary of results related to equivalence

	q^k-0-*majority*	q^k-*majority*	q^∞-0-*majority*	q^∞-*majority*
without inputs	PP-complete	$\mathsf{coNP}^{\mathsf{PP}}$-complete	PP-hard $\leq_{\mathsf{EXP}}$ POSITIVITY	PP-hard ?
with inputs	$\mathsf{coNP}^{\mathsf{PP}}$-complete		? $\mathsf{coNP}^{\mathsf{PP}}$-hard	? $\mathsf{coNP}^{\mathsf{PP}}$-hard

Figure 2: Summary of results related to majority

simpler methods fail or only provide sufficient conditions. Our decidability result significantly generalize prior work on universal equivalence [3], which considers the case of linear programs, see Related Work below. In the special case of *arithmetic* programs, i.e., programs without conditionals nor conditioning, equivalence can be decided in EXP-time, rather than 2-EXP.

Second, we give an exponential reduction from the universal 0-majority problem to the positivity problem for Linear Recurrence Sequences (LRS), which given a LRS, asks whether it is always positive. Despite its apparent simplicity, the positivity problem remains open. Decidability has been obtained independently by Mignotte et al [25] and by Vereshchagin [33] for LRS of order ≤ 4 and later by Ouaknine and Worrell [30] for LRS with order ≤ 5. Moreover, Ouaknine and Worrell prove in the same paper that deciding positivity for LRS of order 6 would allow to solve hard open problems in Diophantine approximation. In the general case, the best known lower bound for the positivity problem is NP-hardness [29].

Our reduction is based on the observation that the Taylor series of any rational functional satisfies a linear recurrence sequence. Therefore, every tuple of polynomials yields a linear recurrence sequence via its local zeta Riemann function. Unfortunately, the order of the linear reccurence sequence is related to the degree of the local zeta Riemann function, and thus decidability results for small orders do not apply. This suggests that the problem may not have an efficient solution. Using the results from [17], we observe that the reduction extends to a more general form of universal majority problem.

Finally, we obtain lower complexity bounds by reducing the finite case to the universal case. It remains an interesting open question whether the universal case is strictly harder than the finite case.

Figures 1 and 2 summarize our results for the equivalence and majority problems.

Related work

Universal equivalence The case of linear programs is studied in [3]. The authors propose a decision procedure for universal equivalence based on the classic XOR-lemma [10]. We give an alternative decision procedure and analyze its complexity.

The case of linear programs with random oracles is considered in [8]. The authors give a polynomial time decision procedure for computational indistinguishability of two inputless programs. Informally, computational indistinguishability is an approximate notion of universal equivalence, stating that the statistical distance between the output of two programs on the same input is upper bounded by a negligible function of the parameter k. Their proof is based on linear algebra.

The case of pseudo-linear (i.e. linear with conditionals) programs is considered in [16]. The authors consider the universal simulatability problem, rather than the universal equivalence problem. The crux of their analysis is a completeness theorem for pseudo-linear functions. In Section 4.3, we show that universal equivalence reduces to universal simulatability. As [16] shows the decidability of universal simulatability for pseudo-linear programs, it therefore follows that universal equivalence of pseudo-linear programs is decidable.

Fixed equivalence There is a vast amount of literature on proving equivalence of probabilistic programs. We only review the most relevant work here.

Murawski and Ouaknine [27] prove decidability of equivalence of second-order terms in probabilistic ALGOL. Their proof is based on a fully abstract game semantics and a connection between program equivalence and equivalence of probabilistic automata.

Legay *et al* [19] prove decidability of equivalence for a probabilistic programming language over finite sets. Their language supports sampling from non-uniform distributions, loops, procedure calls, and open code, but not conditioning. They show that program equivalence can be reduced to language equivalence for probabilistic automata, which can be decided in polynomial time.

Barthe *et al* [5] develop a relational program logic for probabilistic programs without conditioning. Their logic has been used extensively for proving program equivalence, with applications in provable security and side-channel analysis.

Majority problems The closest related work develops methods for proving differential privacy or for quantifying information flow.

Frederikson and Jha [13] develop an abstract decision procedure for satisfiability modulo counting, and then use a concrete instantiatiation of their procedure for checking representative examples from multi-party computation.

Barthe et al [2] show decidability of ϵ-differential privacy for a restricted class of programs. They allow loops and sampling from Laplace distributions, but impose several other constraints on programs. An important aspect of their work is that programs are parametrized by $\epsilon > 0$, so their decision procedure establishes ϵ-differential privacy for all values of ϵ. Technically, their decision procedure relies on the decidability of a fragment of the reals with exponentials by McCallum and Weispfenning [24].

Gaboardi, Nissim and Purser [14] study the complexity of verifying pure and approximate (ϵ, δ)-differential privacy for arithmetic programs, as well as approximations of the parameters ϵ and δ. The parameter δ quantifies the approximation and $\delta = 0$ corresponds to the pure case. Our majority problem can be seen as a subcase of differential privacy, where r corresponds to ϵ, and $\delta = 0$. In particular, the complexity class they obtain for pure differential privacy coincides with the complexity of our 0-majority problem, even when restricted to the case $r = 1$. This means that the ϵ parameter does not essentially contribute to the complexity of the verification problem. Also, while they consider arithmetic programs, we consider the more general case of programs with conditioning.

Chistikov, Murawski and Purser [9] also study the complexity of approximating differential privacy, but in the case of Markov Chains.

Theory of fields A celebrated result by Ax [1] shows that the theory of finite fields is decidable. In a recent development based on Ax's result, Johnson [15] proves decidability of the theory of rings extended with quantifiers $\mu_k^n x.\ P$, stating that the number of x such that P holds is equal to k modulo n. Although closely related, these results do not immediately apply to the problem of equivalence.

2 Programming Language

We consider a high-level probabilistic programming language with sampling from semi-algebraic sets and conditioning, as well as a more pure, yet equi-expressive, core language that can encode all previous constructs and define its formal semantics.

$$
\begin{array}{llll}
P & ::= & & \textit{polynomials} \\
 & | & i \in \mathbb{F}_q & \text{fixed value} \\
 & | & x & \text{variable} \\
 & | & P_1 + P_2 & \text{field addition} \\
 & | & P_1 \times P_2 & \text{field multiplication} \\
 & & & \\
b & ::= & & \textit{boolean conditions} \\
 & | & P = 0 & \text{atomic formula} \\
 & | & b_1 \wedge b_2 & \text{and} \\
 & | & b_1 \vee b_2 & \text{or} \\
 & | & \neg b & \text{not} \\
 & & & \\
e & ::= & & \textit{program expressions} \\
 & | & x := P & \text{assignment} \\
 & | & \vec{r} \xleftarrow{\$} \{X \in \mathbb{F}^m \mid b\} & \text{sampling} \\
 & | & \text{observe } b & \text{observe} \\
 & | & e_1; e_2 & \text{sequential composition} \\
 & | & \text{if } b \text{ then } e_1 \text{ else } e_2 & \text{conditional branching} \\
 & | & \text{return } (P_1, \dots, P_n) & \text{return of arity } n
\end{array}
$$

Figure 3: Program syntax

2.1 Syntax and informal semantics

We define in Figure 3 the syntax for simple probabilistic programs (without loops nor recursion[2]). Our programs will operate on finite fields. We denote by $\mathbb{F}_q$ the (unique) finite field with q elements, where $q = p^s$ for some integer s and prime p. Programs are parametrized by a finite field $\mathbb{F}$, which will be instantiated by some $\mathbb{F}_{q^k}$ during the intepretation. Given a polynomial $P \in \mathbb{F}_q[x_1, \dots, x_m]$ and $X \in \mathbb{F}_{q^k}^m$, we denote by $P(X)$ the evaluation of P given X inside $\mathbb{F}_{q^k}$.

The expressions of our programs provide constructs for assigning a polynomial P to a variable ($x := P$), as well as, for randomly sampling values. With for instance $\vec{r} = r_1, \dots, r_m$, the expression $r_1, \dots, r_m \xleftarrow{\$} \{X \in \mathbb{F}^m \mid b\}$ uniformly samples m values from the set of m-tuples of values in $\mathbb{F}$ such that the condition b holds, and assigns them to variables $r_1, \dots, r_m$. For example, $r \xleftarrow{\$} \{x \in \mathbb{F} \mid 0 = 0\}$ (which we often simply write $r \xleftarrow{\$} \mathbb{F}$) uniformly samples a random element in $\mathbb{F}$, while $r_1, r_2 \xleftarrow{\$} \{x_1, x_2 \in \mathbb{F}^2 \mid \neg(x_1 = 0)\}$ samples two random variables, ensuring that the first one is not 0. Note that the use of polynomial conditions allows to express any rational distribution over the base field $\mathbb{F}_q$.

The construct observe b allows to condition the continuation by b: if b evaluates to false the program fails; the semantics of a program is the conditional distribution where b holds. Expressions also allow classical constructs for sequential composition, conditional branching and returning a result.

In a well-formed program we suppose that every variable is bound at most once, and if it is bound, then it is only used after the binding. Unbound variables correspond to the inputs of the program. We moreover suppose that each branch of a program P ends with a return instruction that returns the same number n of elements; n is then called the arity of the program and denoted $|P|$. Given two sets of variables I and R, we denote by $\mathcal{P}_q(I, R)$ the set of such well-formed programs, where I is the set of unbound variables (intuitively, the set of input variables) and R the set of variables sampled by the program.

Example 2. Consider the following simple program

$$
inv(i) ::= \text{if } i = 0 \text{ then return } 0 \text{ else } r \xleftarrow{\$} \mathbb{F}; \text{observe } r \times i = 1; \text{return } r
$$

This program defines a probabilistic algorithm for computing the inverse of a field element i. If i is 0, by convention the algorithm returns 0. Otherwise, the algorithm uniformly samples an element r. This is

[2]Universal equivalence for programs over finite fields with loops becomes undecidable.

obviously not a practical procedure for computing an inverse, but we use it to illustrate the semantics of conditioning. The observe instruction checks whether r is the inverse of i. If this is the case we return r, otherwise the program fails. As we will see below, our semantics normalizes the probability distribution to only account for non-failing executions. Hence, this algorithm will return the inverse of any positive i with probability 1. Equivalently, this program can be written by directly conditioning the sample .

$$inv'(i) ::= \text{if } i = 0 \text{ then return } 0 \text{ else } r \xleftarrow{\$} \{x \in \mathbb{F} \mid x \times i = 1\}; \text{return } r$$

2.2 A core language

While the above introduced syntax is convenient for writing programs, we introduce a more pure, core language that is actually equally expressive and will ease the technical developments in the remaining of the paper. To define this core language, we add an explicit failure instruction $\perp$, similarly to [6]. It allows us to get rid of conditioning in random samples and observe instructions. Looking ahead, and denoting by $[[P]]^{q^k}$ the semantics of the program P inside $\mathbb{F}_{q^k}$, we will have that

$$\begin{array}{rcl} [[r_1, \dots, r_m \xleftarrow{\$} \{X \in \mathbb{F}^m \mid b\}; e]]^{q^k} & = & [[r_1, \dots, r_m \xleftarrow{\$} \mathbb{F}^m; \text{if } b \text{ then } e \text{ else } \perp]]^{q^k} \text{ and} \\ [[\text{observe } b; e]]^{q^k} & = & [[\text{if } b \text{ then } e \text{ else } \perp]]^{q^k} \end{array}$$

Without loss of generality, we can inline deterministic assignments, and use code motion to perform all samplings eagerly, i.e., all random samplings are performed upfront. Therefore we can simply consider that each variable in R is implicitly uniformly sampled in $\mathbb{F}_{q^k}$. Programs are then tuples of simplified expressions $(e_1, \dots, e_n)$ defined as follows.

$$\begin{array}{rlll} e & ::= & & \textit{simplified expressions} \\ & | & P & \text{polynomial} \\ & | & \perp & \text{failure} \\ & | & \text{if } b \text{ then } e_1 \text{ else } e_2 & \text{conditional branching} \end{array}$$

We suppose that all nested tuples are flattened and write (P, Q) to denote the program which simply concatenates the outputs of P and Q. When clear from the context, we may also simply write $\vec{0}$ instead of the all zero tuple $(0, \dots, 0)$. We denote by $\overline{\mathcal{P}}_q(I, R)$ the set of arithmetic programs, that are simply tuples of polynomials. Remark that arithmetic programs cannot fail.

One may note that the translation from the surface language to the core language is not polynomial in general. Indeed, constructs of the form (if b then $x := t_1$ else $x := t_2; P$), i.e. sequential composition after a conditional, implies to propagate the branching over the assignment to all branches of P, and doubles the number of conditional branchings of P. All complexity results will be given for the size of the program given inside the core language. Remark that in a functional style version of the surface language, where we replace $x := t$ by let $x = t$ in and removed sequential composition, the translation would however be polynomial. Similarly, for the class of programs without sequential composition after conditional branchings, the translation is also polynomial.

2.3 Semantics

We now define the semantics of our core language. The precise translation from the high level syntax previously presented and our core language is standard and omitted.

Deterministic semantics. We first define a *deterministic* semantics where all random samplings have already been defined. For a set X of variables, with $P \in \mathbb{F}_q[X]$ and $\vec{x} \in \mathbb{F}_{q^k}^{|X|}$, $P(\vec{x})$ classicaly denotes the evaluation of P inside $\mathbb{F}_{q^k}$. We also denote $b(\vec{v})$ the evaluation of a boolean test, where all polynomials are evaluated according to $\vec{v}$. For a program $e \in \mathcal{P}_q(I, R)$ and $\vec{v} \in \mathbb{F}_{q^k}^{|I \cup R|}$, we define a natural evaluation of

e, denoted $[e]^{q^k}_{\vec{v}}$, which is a value inside $\mathbb{F}^{|P|}_{q^k} \times \{\perp\}$:

$$\begin{array}{rcl} [P]^{q^k}_{\vec{v}} & = & P(\vec{v}) \text{ where } P \in \mathbb{F}_q[I \uplus R] \\ [\perp]^{q^k}_{\vec{v}} & = & \perp \\ \left[\begin{array}{c} \text{if } b \text{ then } e_1 \\ \text{else } e_2 \end{array}\right]^{q^k}_{\vec{v}} & = & \begin{cases} [e_1]^{q^k}_{\vec{v}} \text{ if } b(\vec{v}) \text{ holds on } \mathbb{F}_{q^k} \\ [e_2]^{q^k}_{\vec{v}} \text{ if } b(\vec{v}) \text{ does not hold on } \mathbb{F}_{q^k} \end{cases} \\ [(e_1, \ldots, e_n)]^{q^k}_{\vec{v}} & = & \begin{cases} \perp \text{ if } [e_i]^{q^k}_{\vec{v}} = \perp \text{ for some } i \\ ([e_1]^{q^k}_{\vec{v}}, \ldots, [e_n]^{q^k}_{\vec{v}}) \text{ else} \end{cases} \end{array}$$

Intuitively, the set of executions corresponding to non failure executions represent the set of possible executions of the program. We next define probabilistic semantics by sampling uniformly the valuations of the random variables while conditioning on the fact that the program does not fail.

Probabilistic semantics. For any n, the set of distributions over $\mathbb{F}^n_q$ is denoted by $\mathsf{Distr}(\mathbb{F}^n_q)$. For a program $P \in \mathcal{P}_q(I, R)$ with $|P| = n$, and $|I| = m$, we define its semantics to be a function from inputs to a distribution over the outputs:

$$[[P]]^{q^k} : \mathbb{F}^m_{q^k} \to \mathsf{Distr}(\mathbb{F}^n_{q^k})$$

We assume that programs inside $P \in \mathcal{P}_q(I, R)$ do not fail all the time, i.e., for any possible input and any program its probability of failure is strictly less than 1. For program P, input $\vec{i} \in \mathbb{F}^m_{q^k}$ and output $\vec{o} \in \mathbb{F}^n_{q^k}$ we set $\mathbb{P}\{\vec{x} = \vec{o} \mid \vec{x} \xleftarrow{\$} [[P]]^{q^k}(\vec{i})\}$ to

$$\frac{\mathbb{P}\{[P]^{q^k}_{\vec{i},\vec{r}} = \vec{o} \mid \vec{r} \xleftarrow{\$} \mathbb{F}^{|R|}_{q^k}\}}{\mathbb{P}\{[P]^{q^k}_{\vec{i},\vec{r}} \neq \perp \mid \vec{r} \xleftarrow{\$} \mathbb{F}^{|R|}_{q^k}\}}$$

Note that the normalization by conditioning on non-failing programs is well defined as we supposed that programs do not always fail.

3 The fixed case

We start by studying the complexity of several problems over a given finite field. In this case we only manipulate finite objects, and hence all problems are obviously decidable, by explicitly computing the distributions. We however provide precise complexity results and show that these problems have complexities in the counting hierarchy [32]. We also define the universal variant and state some results that are common to both variants of the problems.

3.1 Conditional equivalence

In this section, we prove that for any $k \in \mathbb{N}$, the q^k-*equivalence* problem is $\mathsf{coNP}^{\mathsf{C_=P}}$-complete. To this end, we introduce a technical generalization of the equivalence problem, that we call q^k-*conditional equivalence*, and we proceed in four steps, showing that:

1. without loss of generality, we can consider programs without inputs; (Lemma 4)
2. verifying if the conditioned distributions of two inputless programs coincide on a fixed point is in $\mathsf{C_=P}$; (Lemma 5)
3. verifying if the conditioned distribution of inputless programs coincide on all points is in $\mathsf{coNP}^{\mathsf{C_=P}}$; (Corollary 6)
4. and finally, even equivalence for programs over $\mathbb{F}_2$ is $\mathsf{coNP}^{\mathsf{C_=P}}$-hard. (Lemma 7)

3.1.1 Defining conditional equivalence

q^k*-conditional equivalence* is a generalization of equivalence, where we require programs to be equivalent when the distributions are conditioned by some other program being equal to zero. Conditional equivalence is a technical generalisation, that is interesting because it is self-reducible when removing for instance the conditionals.

Definition 3 (q^k*-conditional equivalence*)**.** Let $P_1, Q_1 \in \mathcal{P}_q(I, R)$ and $P_2, Q_2 \in \overline{\mathcal{P}}_q(I, R)$ with $|P_1| = |Q_1| = n$. We denote $P_1 \mid P_2 \approx_{q^k} Q_1 \mid Q_2$, if:

$$\forall \vec{i} \in \mathbb{F}_{q^k}^{|I|}.\ \forall \vec{c} \in \mathbb{F}_{q^k}^{n}.\ [[(P_1, P_2)]]_{\vec{i}}^{q^k}(\vec{c}, \vec{0}) = [[(Q_1, Q_2)]]_{\vec{i}}^{q^k}(\vec{c}, \vec{0})$$

The universal version q^∞*-conditional equivalence* is defined similarly to q^∞*-equivalence* , i.e.,

$$P_1 \mid P_2 \approx_{q^\infty} Q_1 \mid Q_2 \text{ iff } \forall k \in \mathbb{N}.\ P_1 \mid P_2 \approx_{q^k} Q_1 \mid Q_2.$$

Note that conditional equivalence is a direct generalization of equivalence, as for $P, Q \in \mathcal{P}_q(I, R)$, $P \approx_{q^k} Q$ if and only if $P \mid 0 \approx_{q^k} Q \mid 0$.

We also remark that equivalence over $\mathbb{Z}$ is undecidable, which is a consequence of Hilbert's 10th problem, as a polynomial over randomly sampled variables will be equivalent to zero if and only if it does not have any solutions.

We first define precisely the decision problems associated to our questions, for $k \in \mathbb{N} \cup \{\infty\}$:

q^k*-conditional equivalence*
INPUT: $P_1, Q_1 \in \mathcal{P}_q(I, R)$, $P_2, Q_2 \in \overline{\mathcal{P}}_q(I, R)$ QUESTION: $P_1 \mid P_2 \approx_{q^k} Q_1 \mid Q_2$?

The decision problem for q^k*-equivalence* simply corresponds to q^k*-conditional equivalence* with P_2 and Q_2 being equal to 0. In the following we will show that both problems are interreducible, and that q^k*-equivalence* and q^k*-conditional equivalence* are both $\mathsf{coNP}^{\mathsf{C_=P}}$-complete.

3.1.2 Complexity results for conditional equivalence

Recall that $\mathsf{C_=P}$-complete is the set of decision problems solvable by a NP Turing Machine whose number of accepting paths is equal to the number of rejecting paths. halfSAT is the natural $\mathsf{C_=P}$-complete problemdefined as follows.

halfSAT
INPUT: CNF boolean formula ϕ QUESTION: Is ϕ true for exactly half of its valuations?

$\mathsf{coNP}^{\mathsf{C_=P}}$ is the set of decision problems whose complement can be solved by a NP Turing Machine with access to an oracle deciding problems in $\mathsf{C_=P}$. The canonical $\mathsf{coNP}^{\mathsf{C_=P}}$ problem is (using the results from [31, Sec. 4] and [22]) :

A−halfSAT
INPUT: CNF boolean formula $\phi(X, Y)$ QUESTION: For all valuations of X, is $\phi(X, Y)$ true for exactly half of the valuations of Y?

Also, recall that conditional equivalence is a direct generalization of equivalence. We thus trivially have, for any $k \in \mathbb{N} \cup \{\infty\}$, that q^k*-equivalence* reduces in polynomial time to q^k*-conditional equivalence.*

We first study the complexity of deciding if the distributions of two programs are equal on a specific point. To do so, we remark that it is not necessary to consider inputs when considering equivalence or conditional equivalence. The intuition is that inputs can be seen as random values, that must be synchronized on both sides. This synchronization is achieved by explicitly adding these random variables to the output, forcing them to have the same value on both side. The following Lemma is a generalization to conditional equivalence of a Lemma from [4].

Lemma 4. *For any $k \in \mathbb{N} \cup \{\infty\}$, q^k-*conditional equivalence *reduces to q^k-*conditional equivalence *restricted to programs without inputs in polynomial time.*

Omitted proofs can be found in Appendix A. As we can without loss of generality ignore the inputs, we study the complexity of deciding if the distributions of two inputless programs coincide on a specific point. To this end, we build a Turing Machine, such that it will accept half of the time if and only if the programs given as input have the same probability to be equal to some given value. Essentially, it is based on the fact that over $\mathbb{F}_2$, if $r = 0$ then P else $(Q+1) \approx_2 r$ if and only if $P \approx_2 Q$.

Lemma 5. *Let $P_1, Q_1 \in \mathcal{P}_q(\emptyset, R)$ and $P_2, Q_2 \in \overline{\mathcal{P}}_q(\emptyset, R)$ with $|P_1| = |Q_1| = n$. For any $\vec{c} \in \mathbb{F}_{q^k}^n$, we can decide in* $\mathsf{C_=P}$ *if:*

$$[[(P_1, P_2)]]^{q^k}(\vec{c}, \vec{0}) = [[Q_1, Q_2]]^{q^k}(\vec{c}, \vec{0})$$

Proof. As a shortcut, for $P \in \mathcal{P}_q(\emptyset, R)$ (a program without inputs) and $\vec{o} \in \mathbb{F}_{q^k}^{|P|} \times \{\perp\}$, we denote by $\tilde{P}^{\vec{o}}$, the probability that P evaluates to $\vec{o}$. Let $P_1, Q_1 \in \mathcal{P}_q(\emptyset, R)$, $P_2, Q_2 \in \overline{\mathcal{P}}_q(\emptyset, R)$ with $|P_1| = |Q_1| = n$. For any $c \in \mathbb{F}_q^n$, let us consider the probabilistic polynomial time Turing Machine M which on input $P_1, P_2, Q_1, Q_2, \vec{c}$ is defined by:

$x \xleftarrow{\$} \{0,1\}; \vec{r} \xleftarrow{\$} \mathbb{F}_q^{|R|}; \vec{r'} \xleftarrow{\$} \mathbb{F}_q^{|R|};$
if $x = 0$ then
  if $\neg(P_1(\vec{r}) = \vec{c} \wedge P_2(\vec{r}) = \vec{0} \wedge Q_1(\vec{r'}) \neq \perp)$ then
    ACCEPT
  else REJECT
else
  if$(Q_1(\vec{r}) = \vec{c} \wedge Q_2(\vec{r}) = \vec{0} \wedge P_1(\vec{r'}) \neq \perp)$ then
    ACCEPT
  else REJECT

Let $P = (P_1, P_2)$ and $Q = (Q_1, Q_2)$. The probability that M accepts is, by case disjunction on the value of x:

$$\tfrac{1}{2}(1 - \tilde{P}^{(\vec{c},\vec{0})}(1 - \tilde{Q_1}^{\perp})) + \tfrac{1}{2}(\tilde{Q}^{(\vec{c},\vec{0})}(1 - \tilde{P_1}^{\perp})) = \tfrac{1}{2} + \frac{\tilde{Q}^{(\vec{c},\vec{0})}(1-\tilde{P_1}^{\perp}) - \tilde{P}^{(\vec{c},\vec{0})}(1-\tilde{Q_1}^{\perp})}{2}$$

And thus:

$$\begin{aligned}
[[P]]^{q^k}(\vec{c}, \vec{0}) = [[Q]]^{q^k}(\vec{c}, \vec{0}) &\Leftrightarrow \frac{\mathbb{P}\{[P]^{q^k}_{\vec{i},\vec{r}} = (\vec{c},\vec{0}) \mid \vec{r} \xleftarrow{\$} \mathbb{F}_q^{|R|}\}}{\mathbb{P}\{[P]^{q^k}_{\vec{i},\vec{r}} \neq \perp \mid \vec{r} \xleftarrow{\$} \mathbb{F}_q^{|R|}\}} = \frac{\mathbb{P}\{[Q]^{q^k}_{\vec{i},\vec{r}} = (c,\vec{0}) \mid \vec{r} \xleftarrow{\$} \mathbb{F}_q^{R}\}}{\mathbb{P}\{[Q]^{q^k}_{\vec{i},\vec{r}} \neq \perp \mid \vec{r} \xleftarrow{\$} \mathbb{F}_q^{R}\}} \\
&\Leftrightarrow \frac{\tilde{P}^{(\vec{c},\vec{0})}}{1-\tilde{P_1}^{\perp}} = \frac{\tilde{Q}^{(\vec{c},\vec{0})}}{1-\tilde{Q_1}^{\perp}} \\
&\Leftrightarrow \tilde{Q}^{(\vec{c},\vec{0})}(1 - \tilde{P_1}^{\perp}) - \tilde{P}^{(\vec{c},\vec{0})}(1 - \tilde{Q_1}^{\perp}) = 0 \\
&\Leftrightarrow M \text{ accepts exactly half of the time}
\end{aligned}$$

□

As $\mathsf{C_=P}$ is closed under finite intersection [31], we can decide in $\mathsf{C_=P}$ if two distributions over a set of fixed size are equal, by testing the equality over all points. When we only consider inputless programs of fixed arity, the set of points to test is constant, and the equivalence problem is in $\mathsf{C_=P}$ (see Corollary 38 for details). However, when we extend to inputs, or to programs of variable arity, we need to be able to check for all possible points if the distribution are equal over this point. (Note that our encoding that allows to only consider inputless programs increases the arity.) Checking all possible points is typically in coNP. We thus obtain that:

Corollary 6. *q^k-*equivalence *and q^k-*conditional equivalence *are in* $\mathsf{coNP}^{\mathsf{C_=P}}$ *for any $k \in \mathbb{N}$.*

To conclude completeness for both q^k-*equivalence* and q^k-*conditional equivalence*, it is sufficient to show the hardness of 2-*equivalence*, which we do by reducing A−halfSAT. We simply transform a CNF boolean formula into a polynomial over $\mathbb{F}_2$, such that the polynomial is uniform if the formula is in A−halfSAT. This is a purely technical operation (see Lemma 39).

Lemma 7. 2-equivalence *is* $\mathsf{coNP}^{\mathsf{C_=P}}$*-hard.*

3.2 Independence

We show here that equivalence and (conditional) indepence have the same complexity. Conditional independence asks if for any fixed value of some variables Y, the programs are independent, i.e if the product of their distributions is equal to the distribution of their product.

Definition 8 (q^k*-conditional independence*)**.** Let $P_1, \dots, P_n \in \mathcal{P}_q(I, R)$. Given $Y \subset R$, we say that $P_1, \dots, P_n$ are independent conditioned by Y, denoted $\perp^Y_{q^k} (P_1, \dots, P_n)$, if:

$$\forall \vec{i} \in \mathbb{F}^{|I|}_{q^k}. \forall \vec{i'} \in \mathbb{F}^{|Y|}_q. [[(P_1, \dots, P_n)]]^{q^k}_{\vec{i},\vec{i'}} = ([[P_1]]^{q^k}_{\vec{i},\vec{i'}}, \dots, [[P_n]]^{q^k}_{\vec{i},\vec{i'}})$$

We write $\perp_{q^k} (P_1, \dots, P_n)$ for $\perp^{\emptyset}_{q^k} (P_1, \dots, P_n)$, which simply denotes independence of the programs.

Example 9. Independence implies that the distribution of one of the programs does not provide any information about the distribution of the other one. In particular, considering programs in $\mathcal{P}_2(\{i_1, i_2\}, \{r\})$, we have that $\perp_2 (i_1(i_2 + r), i_2)$, which means that $i_1(i_2 + r)$ leaks no information about i_2. However, $\not\perp_2 (i_1(i_2 + r), i_1)$.

We define the decision problem associated to independence, for $k \in \mathbb{N} \cup \{\infty\}$:

q^k*-conditional independence*
INPUT: $P \in \mathcal{P}_q(I, R)$, $Y \subset R$ QUESTION: $\perp^Y_{q^k} P$?

The universal version, q^∞-*conditional independence*, is defined as expected. We now prove that q^k-*conditional independence* is also $\mathsf{coNP}^{\mathsf{C_=P}}$-complete in two steps: first, we reduce conditional independence to indenpence, and, second, we reduce independence to equivalence.

To reduce to non conditional independence, we show that we can replace the conditionned random variable by some fresh input variable.

Lemma 10. *Let* $P_1, \dots, P_n$ *be programs over* $\mathcal{P}_{q^k}(I, R)$*, and* $Y \subset R$*.*

$$\perp^Y_{q^k} (P_1, \dots, P_n) \Leftrightarrow \perp_{q^k} (P_1\sigma, \dots, P_n\sigma)$$

where $\sigma : Y \longrightarrow I_Y$ *is the substitution that replaces each variable in* Y *by a fresh input variable in* I_Y*.*

To reduce independence to equivalence, the idea is that if n programs (as a tuple) are equivalent to a copy of the n programs where they all sample independently there randomness, they are independent. This translates into the following Lemma.

Lemma 11. *Let* $P_1, \dots, P_n$ *be programs over* $\mathcal{P}_{q^k}(I, \{r_1, \dots, r_m\})$

$$\perp_{q^k} (P_1, \dots, P_n) \Leftrightarrow (P_1, \dots, P_n) \approx_q (P_1\sigma 1, \dots, P_n\sigma_n)$$

where σ_i *is the substitution that to any* r_j *associates a fresh random variable* r^i_j*.*

The two previous Lemmas directly yield the following corollary.

Corollary 12. q^k-conditional independence *is in* $\mathsf{coNP}^{\mathsf{C_=P}}$*.*

It remains to show the hardness of conditional independence. The key idea is that for any program P and fresh random r, we have that $\perp^{\emptyset}_2 (P+r, r)$ if and only if P follows the uniform distribution. Intuitively, P perfectly masks the dependance in r only if it is a uniform value. Then, we reduce uniformity to independence, and as we previously reduced A−halfSAT to uniformity, we conclude.

Theorem 13. q^k-conditional independence *is* $\mathsf{coNP}^{\mathsf{C_=P}}$*-complete.*

3.3 Majority

The goal of this section is to show that the majority problem is $\mathsf{coNP}^{\mathsf{PP}}$-complete. To this end, we study the complexity of q^k-0-*majority*, showing:

- PP-completeness for inputless programs;
- $\mathsf{coNP}^{\mathsf{PP}}$-completeness in general.

The proof in both cases uses similar ideas as for equivalence. Note that we actually use the same Turing Machine for the Membership. As both complexity classes are closed under finite intersection, it yields the complexity of q^k-*majority*, which can be decided using q^k times q^k-0-*majority*.

3.3.1 The majority problem

q^k-*majority* asks if, given two programs, the quotient of their distribution is bounded on all points by some rational r. q^k-0-*majority* is a subcase, where we only ask if the quotient of their distribution is bounded on a single point. This problem allows to estimate the distance between two distributions. It is close to the differential privacy question, which asks, when $\delta = 0$, if the quotient of two distributions is bounded over all points by some e^ϵ.

We observe that the majority problem is harder than equivalence, as majority for $r = 1$ implies equivalence. An important difference between equivalence and majority is that the presence of inputs actually changes the complexity of the majority problem.

Let us define the decision problem associated to q^k-*majority*, with $k \in \mathbb{N} \cup \{\infty\}$:

q^k-*majority*
INPUT: $P, Q \in \mathcal{P}_q(I, R)$, $r \in \mathbb{Q}$ QUESTION: $P \prec^r_{q^k} Q$?

We consider that r is given in input as two integers written in unary. Essentially, this is because if one whishes to encode any r, it requires an exponential blow up, but in practice, we tend to use some particular rationals such as $r = q^l$, for which there is no exponential blow up.

3.3.2 Complexity results for the majority problem

We recall that PP is the set of languages accepted by a probabilistic polynomial-time Turing Machine with an error probability of less than $1/2$ for each instance, i.e., a word in the language is accepted with probability at least $1/2$, and a word not in the language is accepted with probability less than $1/2$. Alternatively, one can define PP as the set of languages accepted by a non-deterministic Turing Machine where the acceptance condition is that a majority of paths are accepting. Notably, PP contains both NP and coNP, as well as $\mathsf{C_{=}P}$. Also, PP is closed uner finite intersection. A natural PP-complete problem is MAJSAT: is a boolean CNF formula satisfied for at least half of its valuations:

MAJSAT
INPUT: CNF boolean formula ϕ QUESTION: Is ϕ true for at least half of its valuations?

$\mathsf{coNP}^{\mathsf{PP}}$ is the class of of problems whose complement is decided by a NP Turing Machine with access to an oracle deciding problems in PP. The classical $\mathsf{NP}^{\mathsf{PP}}$ problem is E−MAJSAT [22] :

E−MAJSAT
INPUT: CNF boolean formula $\phi(X, Y)$ QUESTION: Is there a valuation of X such that, $\phi(X, Y)$ is true for at least half of the valuation of Y?

Its complement, A−MINSAT is then the classical $\mathsf{coNP}^{\mathsf{PP}}$ problem.

To obtain the complexity of q^k-0-*majority* over inputless programs, we notice that the Turing Machine we used to obtain the complexity of the equivalence problem are easily adapted for our purpose. Indeed, it accepted half of the time if the two distributions were equal on a single point, but it actually accepts with probability greater than half only if the value of the first distribution is greater than the second one on the given point.

The only difficulty is that we are comparing with a rational. We thus briefly show how one can assume without loss of generality that $r = 1$ (in which case we omit r from the notation). The idea is , given $r, s \in \mathbb{N}$, that $P \prec_{q^k}^{\frac{r}{s}} Q \Leftrightarrow (P, T_r) \prec_{q^k} (Q, T_s)$, if T_j is a machine which is equal to zero with probability $\frac{1}{j}$.

Lemma 14. *For any $k \in \mathbb{N}$, q^k-*0-majority *reduces in polynomial time to q^k-*0-majority *with $r = 1$.*

The proof showing that q^k-0-*majority* is in PP is similar to proving that testing if two distributions are equal over a point is in $\mathsf{C_=P}$. We prove PP-completeness by deriving the hardness from MAJSAT.

Lemma 15. *For any $k \in \mathbb{N}$, q^k-*0-majority *restricted to inputless programs is* PP*-complete.*

Finally, as PP is closed under finite intersection, we also get that q^k-*majority* over inputless programs with a fixed arity is PP-complete.

Let us now turn to the general version, for programs with inputs. By using some fresh inputs variables, let us remark that one can easily reduce q^k-*majority* to q^k-0-*majority*. Indeed, for $P, Q \in \mathcal{P}_q(I, R)$ and $c \in \mathbb{F}_q^{|P|}$, with a fresh $x \in I$:

$$\forall \vec{i} \in \mathbb{F}_{q^k}^{|I|}.\ [[P]]_{\vec{i}}^{q^k}(c) \leq r[[Q]]_{\vec{i}}^{q^k}(c) \Leftrightarrow (P - x) \prec_{q^k}^{r} (Q - x)$$

We show that q^k-*majority* is $\mathsf{coNP}^{\mathsf{PP}}$ complete, and thus is most likely[3] harder than its version without inputs. The membership and hardness proofs are similar to the equivalence problem when going from $\mathsf{C_=P}$ to $\mathsf{coNP}^{\mathsf{C_=P}}$.

Lemma 16. q^k-majority *is* $\mathsf{coNP}^{\mathsf{PP}}$ *complete.*

4 The universal case

In this section we first give some general insights on universal equivalence showing important differences with the case of a fixed field. Then we provide our main decidability result, first for arithmetic programs, then arithmetic programs enriched with conditionals, and finally for general programs. We continue by studying two other problems in the universal case: simulatability and 0-majority.

4.1 General remarks

In this section we try to provide some insights on the difficulty of deciding q^∞-*equivalence*. First of all, we note that equivalence and universal equivalence do *not* coincide.

Example 17. The program $x^2 + x$ and the program 0 are equivalent over $\mathbb{F}_2$ (they are then both equal to zero), but not over $\mathbb{F}_4$.

In the case of a given finite field, equivalence can be characterized by the existence of a bijection, see for instance [4]. We denote by $\mathsf{bij}^{\mathbb{F}_q^m}$ the set of bijections over $\mathbb{F}_q^m$. Any element $\sigma \in \mathsf{bij}^{\mathbb{F}_q^m}$ can be expressed as a tuple of polynomials (see e.g. [28]), and can be applied as a substitution. The characterization can then be stated as follows, where we denote by $=_{\mathbb{F}_q}$ equality between polynomials modulo the rule of the field (i.e., $X^q = X$).

$$P \approx_q Q \Leftrightarrow \exists \sigma \in \mathsf{bij}^{\mathbb{F}_q^m}, P =_{\mathbb{F}_q} Q\sigma$$

However, there are universally equivalent programs such that there does *not* exist a universal σ suitable for all extensions.

[3] As $\mathsf{PH} \subset \mathsf{coNP}^{\mathsf{PP}}$, $\mathsf{PP} = \mathsf{coNP}^{\mathsf{PP}}$ would imply $\mathsf{PH} \subset \mathsf{PP}$ which is commonly believed to be false.

Example 18. Consider, $P = xy + yx + zx$, with $\sigma : (x, y, z) \mapsto (x, y + x, z + x)$, we get that $P \approx_{2^\infty} x^2 + yz$. Now, $x \mapsto x^2$ is a bijection over all $\mathbb{F}_{2^k}$, so we also have $P \approx_{2^\infty} x + yz$ and finally $P \approx_{2^\infty} x$.

But here, a bijection between $x^2 + yz$ and x must use the inverse of x^2 whose expression depends on the size of the field. Thus, there isn't a universal polynomial σ which is a bijection such that on all $\mathbb{F}_{2^k}$, $P =_{\mathbb{F}_{2^k}} Q \circ \sigma$.

Nevertheless, we can note that for linear programs this characterization allows us to show that *q-equivalence* and *q^∞-equivalence* are equivalent. Intuitively, the bijection allowing to obtain the equality between two linear programs is also a bijection valid for all extensions of the finite field, as the bijection is linear, and is thus a witness of equivalence over all extensions. For linear programs, there exists a polynomial time decision procedure for equivalence, and hence for universal equivalence.

Lemma 19. q^∞-equivalence *restricted to linear programs is in* PTIME.

Moreover, building on results from [23] on Tame automorphisms, we can use the above characterization to design a sufficient condition which implies universal equivalence for general programs. Even though not complete this sufficient condition may be useful to verify universal equivalence more efficiently in practice.

A Sufficient Condition

In the univariate case, our notion is also strongly linked to what mathematicians calls exceptional polynomials, permutation polynomials over $\mathbb{F}_q[x]$ that are permutations over infinitely many $\mathbb{F}_{q^k}[x]$.

A univariate polynomial which is uniform is then an exceptional polynomial of $\mathbb{F}_q[x]$. They have been fully characterized [26, p237]. However, the multivariate case appears unsolved.

With the previous characterization, we can however easily obtain the following condition, for any function σ:

$$\sigma \in \bigcap_k \mathsf{bij}^{\mathbb{F}^m_{q^k}} \Rightarrow P \approx_{q^\infty} P\sigma$$

To better understand this condition, we now provide some results provinding some insights about functions that are bijections over all extensions of a finite field.

We first use Theorem 3.2 of [23] to classify what are the bijections over $\mathbb{F}^m_{q^k}$. For a finite field $\mathbb{F}$, $\mathsf{bij}^{\mathbb{F}^n}$ denotes the set of bijections over $\mathbb{F}^n$, and $\mathcal{E}(T(\mathbb{F}, n))$ denotes the set of bijections obtained through permutations, scalar multiplications (for any $a \in \mathbb{F}^*$, $(x_1, \dots, x_n) \mapsto (ax_1, \dots, x_n)$) and linear transformations (for any $P \in k[x_2, \dots, x_n]$, $(x_1, \dots, x_n) \mapsto (x_1 + P(x_2, \dots, x_n), \dots, x_n)$) , which are called the tame automorphisms.

Theorem 20 (2.3 of [23])**.** *We have:*

- *if* $n = 1$*, and* $\mathbb{F} = \mathbb{F}_2$ *or* $\mathbb{F}_3$*, then* $\mathcal{E}(T(\mathbb{F}, n)) = \mathsf{bij}^{\mathbb{F}^n}$*,*
- *if* $n \geq 2$ *and* $\mathbb{F} \neq \mathbb{F}_{2^m}$ *for* $m > 1$ *,* $\mathcal{E}(T(\mathbb{F}, n)) = \mathsf{bij}^{\mathbb{F}^n}$*,*
- *else,* $\mathcal{E}(T(\mathbb{F}, n)) \neq \mathsf{bij}^{\mathbb{F}^n}$*.*

This allows us to obtain that:

Lemma 21. *For any* $k \geq 1$ *and* $n > 1$*, for any function* f*:*

$$f \in \mathsf{bij}^{\mathbb{F}^n_{p^k}} \Rightarrow \forall k' > k. f \in \mathsf{bij}^{\mathbb{F}^n_{p^{k'}}}$$

Proof. Let $f \in \mathsf{bij}^{\mathbb{F}_{p^k}}$. With Theorem 20, we have that for all prime p:

$$\mathcal{E}(T(\mathbb{F}_{p^k}, n)) = \mathcal{B}(\mathbb{F}^n_{p^k})$$

Thus, f can be written as a composition of substitutions, scalar multiplications linear transformations. All those operations are directly bijections over any $\mathbb{F}^n_{p^k}$, we thus conclude:

$$\forall k' > k. f \in \mathsf{bij}^{\mathbb{F}^n_{p^{k'}}}$$

□

The case $p = 2$ must be handled differently:

Lemma 22. *For any $k > 1$ and $n > 1$, for any function f:*

$$f \in \mathrm{bij}^{\mathbb{F}^n_{2^{2(2k+1)}}} \Rightarrow \forall k' > 2(2k+1). f \in \mathrm{bij}^{\mathbb{F}^n_{2^{k'}}}$$

Proof. For any m, we denote $\mathcal{F}(T(\mathbb{F}_{2^m}, n))$ the set generated by $\mathcal{E}(T(\mathbb{F}_{2^m}, n))$ and the permutation $\sigma = (X_1, \ldots, X_n) \mapsto (X_1^2, \ldots, X_n)$. It is shown in [20, p. 351] that x^n is a bijection in $\mathbb{F}_q$ if n and $q-1$ are coprime. We have that for any k, 2 and $2^k - 1$ are coprime, and then, we have that σ is a bijection over all $\mathbb{F}^n_{2^m}$.

If σ is of signature -1, has $\mathcal{E}(T(\mathbb{F}_{2^m}, n))$ contains all elements with a positive signature of $\mathcal{B}(\mathbb{F}_{2^m})$, then we have $\mathcal{F}(T(\mathbb{F}_{2^m}, n)) = \mathcal{B}(\mathbb{F}_{2^m})$.

We have $\sigma^m = Id$, so σ is made of m cycles, and σ only leaves 0 and 1 invariants. Thus, σ has $V = \frac{2^m - 2}{m}$ cycles. For the m cycles to be of sign -1, m must be pair, i.e $m = 2l$. Now, if l is odd, i.e $m = 2(2k+1)$, $V = \frac{2^{m-1}-1}{2k+1}$ is odd, and then σ is of sign -1.

Thus, we have proven that for any k: $\mathcal{F}(T(\mathbb{F}_{2^{2(2k+1)}}, n)) = \mathcal{B}(\mathbb{F}_{2^{2(2k+1)}})$

Let us fix k and let $f \in \mathrm{bij}^{\mathbb{F}_{2^{2(2k+1)}}}$.

Thus, f can be written as a composition of substitutions, scalar multiplications, linear transformations and σ. Recall that σ is a bijection over all $\mathbb{F}^n_{2^k}$, and the others trivially are. We thus conclude:

$$\forall k' > 2(2k+1). f \in \mathrm{bij}^{\mathbb{F}^n_{2^{k'}}}$$

□

Those two lemmas provides an easy way to generate bijections which are bijections over all extension of the finite field, and can thus serve as a witness for a universal equivalence.

4.2 Decidability of universal equivalence

We show decidability of q^∞-*equivalence*, leveraging tools from algebraic geometry, showing that:

1. q^∞-*conditional equivalence* is decidable for arithmetic programs; (Lemma 24)
2. it is also decidable for programs with conditionals; (Lemma 26)
3. it is finally decidable for programs with conditioning, e.g. failures. (Lemma 27)

We first recall the definition and relevant properties of local zeta Riemann functions. For a tuple P of polynomials $P_1, \ldots, P_m \in \mathbb{F}_q[X_1, \ldots, X_n]$, the local zeta Riemann function over T is the formal series

$$Z(P, T) = \exp\left(\sum_{k \in \mathbb{N}^*} \frac{|N_k(P)|}{k} T^k\right)$$

where $N_k(P) = \{\vec{x} \in \mathbb{F}^n_{q^k} \mid \bigwedge_{1 \leq i \leq m} P_i(\vec{x}) = 0\}$. Weil's conjecture [34] states several fundamental properties of local zeta Riemann functions over algebraic varieties. Dwork [11] proves part of Weil's conjecture stating that the local zeta Riemann functions over algebraic varieties is a rational function with integer coefficients—recall that $Z(T)$ is a rational function iff there exist polynomials $R(T)$ and $S(T)$ such that $Z(T) = R(T)/S(T)$. Bombieri [7] shows that the sum of the degrees of R and S is upper bounded by $4(d+9)^{n+1}$, where d is the total degree of $(P_1, \ldots, P_m)$. It follows that the values of N_k for $k \leq 4(d+9)^{n+1}$ suffice for computing Z; since these values can be computed by brute force, this yields an algorithm for computing Z. We will by abuse of notations write $Z(P)$ instead of $Z(P, T)$ for the local zeta function of P. $Z(P)$ completely characterizes the number of times P is equal to zero on all the different extensions. For instance, $Z(P) = Z(Q)$ allows us to conclude that P and Q always evaluate to zero for the same number of valuations, and this over any $\mathbb{F}_{q^k}$. As Z can effectively be computed [18], we can use it to decide q^∞-*equivalence.*

Notice that, given two programs P and Q, the local zeta function directly allow us to conclude if they are equal to some value with the same probability for all extensions of the base field. Moreover, thanks to [17], the computability of the local zeta function can be extended from counting the number of points such that $P = 0$ for a tuple of polynomials, to counting the number of points such that ϕ holds, where ϕ is an arbitrary first order formula over finite fields.

Corollary 23. *Let ϕ and ψ be two first order formulae built over atoms of the form $P = 0$ with $P \in \mathbb{F}_q[X]$, and with free variables $F \subset X$. One can decide if for all $k \in \mathbb{N}$:*

$$\left|\{\vec{f} \in \mathbb{F}_{q^k}^{|F|} \mid \phi(\vec{f}) = 1\}\right| = \left|\{\vec{f} \in \mathbb{F}_{q^k}^{|F|} \mid \psi(\vec{f}) = 1\}\right|$$

Thus, for any two events which can be expressed as a first order formula over finite field one can verify if they happen with the same probability over all extensions of the base field. Remark that this cannot be used to decide universal equivalence, as equivalence cannot be expressed inside a first order formula.

We first show that, thanks to the local zeta functions, q^∞*-equivalence* is decidable for arithemic programs, i.e programs without conditionals or conditioning.

Lemma 24. *Let $P_1, P_2, Q_1, Q_2 \in \overline{\mathcal{P}}_q(\emptyset, R)$.*

$$\begin{array}{ccl} P_1 \mid P_2 & & Z((Q_1 - Q_1\sigma, Q_2, Q_2\sigma)) \\ \approx_{q^\infty} & \Leftrightarrow & = Z((P_1 - Q_1\sigma, P_2, Q_2\sigma)) \\ Q_1 \mid Q_2 & & = Z((P_1 - P_1\sigma, P_2, P_2\sigma)) \end{array}$$

where $\sigma : R \mapsto R'$ maps each variable to a fresh one.

Proof. We assimilate $P_1 P_2, Q_1, Q_2 \in \overline{\mathcal{P}}_q(\emptyset, R)$ of size m with polynomials, denoting $P(X)$ the value of P given $X \in \mathbb{F}_{q^k}^m$. Given an enumeration $1 \leq j \leq s$ of the elements c_j of $\mathbb{F}_{q^k}^n$, for any programs T, T', we let

$$(T, T')_i^k = \left|\{X \in \mathbb{F}_{q^k}^m \mid T(X) = c_i \wedge T'(X) = \vec{0}\}\right|$$

Then, if we denote $\overrightarrow{(T, T')^k} = (T, T')_1^k, \ldots, (T, T')_s^k$, that characterizes the distribution of $T|T'$,

$$\begin{array}{cl} & P_1 \mid P_2 \approx_{q^\infty} Q_1 \mid Q_2 \\ \Leftrightarrow & \forall k \in \mathbb{N}.\ \overrightarrow{(P_1, P_2)^k} = \overrightarrow{(Q_1, Q_2)^k} \end{array}$$

Using the classical inner product $\vec{x} \cdot \vec{y} = \sum_i x_i y_i$, for any k and programs $U, V, U', H' \in \overline{\mathcal{P}}_q$, we have:

$$\begin{array}{rl} N_k((U - V\sigma, U', V')) & = \left|\left\{X, X' \in \mathbb{F}_{q^k}^m \mid \ U(X) = V(X') \wedge (U'(X), V'(X)) = \vec{0}\ \right\}\right| \\ & = \sum_{c \in \mathbb{F}_{q^k}^n} \left|\{X \in \mathbb{F}_{q^k}^m \mid U(X) = c \wedge U'(X) = \vec{0}\}\right| \left|\{X \in \mathbb{F}_{q^k}^m \mid V(X) = c \wedge V'(X) = \vec{0}\}\right| \\ & = \sum_i (U, U')_i^k (V, V')_i^k \\ & = \overrightarrow{(U, U')^k} \cdot \overrightarrow{(V, V')^k} \end{array}$$

Using scalar operations, we have that:

$$\begin{array}{cl} & N_k(Q_1 - Q_1\sigma, Q_2, Q_2\sigma) = N_k(P_1 - Q_1\sigma, P_2, Q_2\sigma) = N_k(P_1 - P_1\sigma, P_2, P_2\sigma) \\ \Leftrightarrow & \overrightarrow{(Q_1, Q_2)^k} \cdot \overrightarrow{(Q_1, Q_2)^k} = \overrightarrow{(P_1, P_2)^k} \cdot \overrightarrow{Q^k} = \overrightarrow{(P_1, P_2)^k} \cdot \overrightarrow{(P_1, P_2)^k} \\ \Leftrightarrow & (\overrightarrow{(P_1, P_2)^k} - \overrightarrow{Q^k}) \cdot (\overrightarrow{(P_1, P_2)^k} - \overrightarrow{(Q_1, Q_2)^k}) = \vec{0} \\ \Leftrightarrow & \overrightarrow{(P_1, P_2)^k} = \overrightarrow{(Q_1, Q_2)^k} \end{array}$$

Hence,

$$\forall k \in \mathbb{N}. \forall c \in \mathbb{F}_{q^k}^n.\ \left|\{X \in \mathbb{F}_{q^k}^m \mid P_1(X) = c \wedge P_2(X) = \vec{0}\}\right| = \left|\{X \in \mathbb{F}_{q^k}^m \mid Q_1(X) = c \wedge Q_2(X) = \vec{0}\}\right|$$
$$\Leftrightarrow$$
$$\forall k \in \mathbb{N}. N_k((Q_1 - Q_1\sigma, Q_2, Q_2\sigma)) = N_k((P_1 - Q_1\sigma, P_2, Q_2\sigma)) = N_k((P_1 - P_1\sigma, P_2, P_2\sigma))$$

This concludes the proof, as for all U, V,

$$Z(U) = Z(V) \Leftrightarrow \forall k.\ N_k(U) = N_k(V)$$

□

In ??, we provide a variant of this result for the specific case of verifying if a program follows the uniform distribution over all extensions, where only one computation of a local zeta function is required.

Using the complexity for the computation of the local zeta function provided by [18, Corollary 2] we obtain the following corollary.

Corollary 25. *q^∞-*equivalence *and q^∞-*conditional equivalence *restricted to arithmetic programs are in* EXP.

We now wish to remove conditionals, in order to reduce equivalence for programs with conditional to arithmetic programs (which are simply tuples of polynomials). To remove the conditionals, the first idea is to use a classical encoding inside finite fields: $[[\text{if } B \neq 0 \text{ then } P_1^t \text{ else } P_1^f]]^{q^k} = [[P_1^f + B^{q^k-1}(P_1^t - P_1^f)]]^{q^k}$. This works nicely as B^{q^k-1} is equal to 0 if $B = 0$, else to 1. However, for the universal case, we need to have an encoding which does not depend on the size of the field, i.e., it must be independent of k. The key idea is that for any variable t and polynomial B:

$$(B(Bt-1) = 0 \wedge t(Bt-1) = 0) \Leftrightarrow t = B^{q^k-2}$$

And thus, we can for instance write, with some program Q and $\vec{c}$:

$$\begin{aligned}[[\text{if } B \neq 0 \text{ then } P_1^t \text{ else } P_1^f]]^{q^k}(\vec{c}) = [[Q]]^{q^k}(\vec{c}) &\Leftrightarrow [[P_1^f + B^{q^k-1}(P_1^t - P_1^f)]]^{q^k}(\vec{c}) = [[Q]]^{q^k}(\vec{c}) \\ &\Leftrightarrow [[P_1^f + Bt(P_1^t - P_1^f), (B(Bt-1), t(Bt-1))]]^{q^k}(\vec{c}, \vec{0}) = [[Q]]^{q^k}(\vec{c})\end{aligned}$$

An induction on the number of conditionals yields our second lemma.

Lemma 26. *For any $k \in \mathbb{N} \cup \{\infty\}$, q^k-*conditional equivalence *restricted to programs without failures reduces in exponential time to q^k-*conditional equivalence *restricted to arithmetic programs.*

Recall that failures define the probablisitic semantics by normalization. And for instance, for some program (if $b = 0$ then P_1 else $\bot, P_2$) where P_1 and P_2 do not fail and b is a polynomial, for any $\vec{c}$, we have:

$$[[(\text{if } b = 0 \text{ then } P_1 \text{ else } \bot, P_2)]]^{q^k}(\vec{c}, \vec{0}) = \frac{\mathbb{P}\{P_1 = \vec{c} \wedge P_2 = \vec{0} \wedge b = 0\}}{\mathbb{P}\{\neg(b = 0)\}}$$

Handling this division by itself would be difficult if we wanted to compute the distribution. However, in our setting, we are comparing the equality of two distributions, so we can simply multiply on both side by the denominator, and try to express once again all factors as an instance of conditional equivalence. We will be able to push inside conditional equivalence some probabilities, as $[[P]]^{q^k}(\vec{c}) \times \mathbb{P}\{b = 0\} = [[P, b]]^{q^k}(\vec{c}, 0)$ when all variables in b do not appear in P.

As an illustration of how to remove the failures, with some program Q, we have:

$$\begin{aligned}\text{if } b = 0 \text{ then } P_1 \text{ else } \bot \mid P_2 \approx_{q^k} Q \mid 0 &\Leftrightarrow \forall \vec{c}. [[(\text{if } b \text{ then } P_1 \text{ else } \bot, P_2)]]^{q^k}(\vec{c}, \vec{0}) = [[Q]]^{q^k}(\vec{c}) \\ &\Leftrightarrow \forall \vec{c}. \mathbb{P}\{P_1 = \vec{c} \wedge P_2 = \vec{0} \wedge b = 0\} = \mathbb{P}\{\neg(b = 0)\}[[Q]]^{q^k}(\vec{c}, \vec{0}) \\ &\Leftrightarrow \forall \vec{c}. [[P_1, P_2, b]]^{q^k}(\vec{c}, \vec{0}) = \mathbb{P}\{\neg(b = 0)\}[[Q]]^{q^k}(\vec{c})\end{aligned}$$

To reduce to an instance of conditional equivalence, the issue is that we need to express as an equality the disequality $b \neq 0$. With some fresh variable t, multiplying by $\mathbb{P}\{\neg(b = 0)\}$ or conditioning on $tb - 1 = 0$ is equivalent, as b has an inverse if and only if it is different from zero. We can thus have:

$$\begin{aligned}\text{if } b = 0 \text{ then } P_1 \text{ else } \bot \mid P_2 \approx_{q^k} Q \mid 0 &\Leftrightarrow \forall \vec{c}. [[P_1, P_2, b]]^{q^k}(\vec{c}, \vec{0}) = \mathbb{P}\{\neg(b = 0)\}[[Q]]^{q^k}(\vec{c}) \\ &\Leftrightarrow \forall \vec{c}. [[P_1, P_2, b]]^{q^k}(\vec{c}, \vec{0}) = [[Q, tb - 1]]^{q^k}(\vec{c}, 0) \\ &\Leftrightarrow P_1 \mid P_2, b \approx_{q^k} Q \mid tb - 1\end{aligned}$$

Using those techniques, we obtain:

Lemma 27. *For any $k \in \mathbb{N} \cup \{\infty\}$,* q^k-conditional equivalence *reduces to* q^k-conditional equivalence *restricted to programs without failures in exponential time.*

The previous Lemmas allows us to conclude.

Theorem 28. q^∞-equivalence *and* q^∞-conditional equivalence *are in* 2-EXP.

And using once again Lemmas 10 and 11, we obtain the same complexity results for the independence problem.

Corollary 29. q^∞-conditional independence *is in* 2-EXP.

Moreover, we can also extend the lower bound obtained for *q-equivalence*.

Lemma 30. q-equivalence *reduces in polynomial time to* q^∞-equivalence.

4.3 Bounded Universal Simulatability

Simulation-based proofs [21] are one main cornerstone of cryptography. Informally, simulation-based proofs consider a real and an ideal world, and require showing the existence of a simulator, such that no adversary can distinguish the composition of the simulator and of the ideal world from the real world. This can be modelled in our context by requiring the existence of a program S (the simulator) such that "plugging in" the ideal world into S is equivalent to the real world. In this section, we consider a simpler task, where the size of the simulator is bounded. Given a program C, we denote $\deg(C)$ the maximum degree of a progam, i.e the maximum degree of any polynomial appearing in C (the degree of a polynomial is the maximum over the sum of the degrees of each monomial).

Definition 31. [Bounded (universal) simulatability] Let $P, Q \in \mathcal{P}_q(I, R)$, R' such that $\sharp R = \sharp R'$ and $l \in \mathbb{N}$. We denote $P \sqsubseteq^l_{q^{[\infty]}} Q$, if there exists $S \in \mathcal{P}_q(\{i_1, \dots, i_n\}, R'\})$ such that $\deg(S) \leq l$, and

$$S[^Q/_{\vec{i}}] \approx_{q^{[\infty]}} P$$

The associated decision problem is:

l,q-*simulatability*
INPUT: $P, Q \in \mathcal{P}_q(I, R)$ QUESTION: $P \sqsubseteq^l_{q^\infty} Q$?

Thanks to the bound on the degree coming from l, we can easily obtain a bound on the number of such possible contexts. This is shown in Lemma 42. From the bound on the number of contexts and the decidability of universal equivalence, one can derive the decidability of bounded simulatability.

Theorem 32. *l,q*-simulatability *is decidable.*

As a lower bound, we prove that *l, q-simulatability* is as hard as universal equivalence:

Lemma 33. *For any $l \in \mathbb{N}$, $k \in \mathbb{N} \cup \{\infty\}$,* q^k-equivalence *reduces in polynomial time to* l, q^k-simulatability.

We conclude this section by noting that our notion of bounded simulatability is more restricted than the general paradigm of simulation-based proofs but could be a good starting point for automating simulation-based proofs.

4.4 Universal zero-majority without inputs

For arbitrary programs, we reduce q^∞-0-*majority* to the POSITIVITY problem. We recall that a Linear Recurrence Sequence (LRS) is an infinite sequence of reals $u = u_1, u_2, \dots$ such that there exist real constants $a_1, \dots, a_k$ such that for all $n \geq 0$,

$$u_{n+k} = a_1 u_{n+k-1} + \cdots + a_k u_n$$

The order of a LRS $u = u_1, u_2, \ldots$ is the smallest k such that the equation above holds. A LRS $u = u_1, u_2, \ldots$ is positive if $u_n \geq 0$ for every $n \in \mathbb{N}$. The positivity problem consists in deciding whether a LRS is positive.

We use the fact that from a local zeta function, which is rational, we can obtain a Linear Recurrence Sequence. Then, by considering the POSITIVITY of the LRS obtained by susbtracting two local zeta function, we actually check if the coefficients of the first one are always greater than the second one. We remark that the complexity of the problem strongly relies on the presence of multiplications, as for q^∞*-equivalence*. Indeed, in the linear case, majority implies equivalence and we obtain the following.

Lemma 34. q^∞*-0-*majority *restricted to linear programs is in* PTIME.

The general case has yet to be proven decidable.

Theorem 35. q^∞-0-majority *for inputless programs reduces in exponential time to* POSITIVITY.

Proof. Let $P, Q \in \overline{\mathcal{P}}_q(\emptyset, R)$. We assume without loss of generality that we only have to consider arithmetic programs, using the same simplifications for observe and conditionals as we did for universal equivalence.

Recall that for any P, the local zeta function $Z(P)$ (over indeterminate T) is rational thanks to Weil's conjecture [11]. Moreover, with $N_k(P) = \left|\{X \in \mathbb{F}_{q^k}^m \mid P(X) = 0\}\right|$, we have that:

$$\frac{d}{dT} log(Z(P)) = \frac{Z'(P)}{Z(P)} = \sum_k N_k(P) T^k$$

Let us call $\tilde{Z}(P) = \sum_k N_k(P)T^k$, which is also a rational function as Z is (and so is Z'). As the coefficients of $\tilde{Z}(P) - \tilde{Z}(Q)$ are $N_k(P) - N_k(Q)$, we have that $\forall k, N_k(P) \geq N_k(Q)$ if and only if $\tilde{Z}(P) - \tilde{Z}(Q)$ only has positive coefficients. It is well known that the coefficients of the Taylor serie of a rational function form a LRS (see e.g. [12]). This means that the coefficients of $\tilde{Z}(P) - \tilde{Z}(Q)$ form an LRS, which we denote z^{PQ}. Finally:

$$\begin{aligned} Q <_q^\infty P &\Leftrightarrow \forall k, N_k(Q) \leq N_k(P) \\ &\Leftrightarrow \tilde{Z}(P) - \tilde{Z}(Q) \text{ only has positive coefficients} \\ &\Leftrightarrow \forall n, z_n^{PQ} \geq 0 \end{aligned}$$

□

This reduction can also be applied with the generalization of [17], and thus, for any two events about programs over finite fields, one can, given an oracle for the POSITIVITY problem, decide if the probability of the first event is greater than the second one for all extensions of the base field.

Similarly to the equivalence case, we can derive some hardness from the non universal case, but we do not obtain any completeness result.

Lemma 36. 2^∞-0-majority *is* PP*-hard.*

5 Conclusion

We have introduced universal equivalence and majority problems and studied their complexity and decidability. Our work could notably be used as a building block to design a decidable logic for universal probabilistic program verification. It leaves several questions of interest open:

- the exact complexity of universal equivalence is open. It is even unknown whether the universal problem strictly harder than the non-universal one;
- the decidability of universal majority is open. The decidability of POSITIVITY would yield decidability of universal 0-majority and equivalently, undecidability of universal majority would also solve negatively the POSITIVITY problem;
- the decidability of universal approximate equivalence is open. Approximate equivalence asks whether the statistical distance between the distributions of two programs is negligible in k. This notion has direct applications in provable security.

References

[1] James Ax. The elementary theory of finite fields. *Annals of Mathematics*, 88(2):239–271, 1968.

[2] Gilles Barthe, Rohit Chadha, Vishal Jagannath, A. Prasad Sistla, and Mahesh Viswanathan. Automated methods for checking differential privacy. *CoRR*, abs/1910.04137, 2019.

[3] Gilles Barthe, Marion Daubignard, Bruce Kapron, Yassine Lakhnech, and Vincent Laporte. On the equality of probabilistic terms. In *16th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR'10)*, volume 6355 of *Lecture Notes in Computer Science*, pages 46–63. Springer, 2010.

[4] Gilles Barthe, Benjamin Grégoire, Charlie Jacomme, Steve Kremer, and Pierre-Yves Strub. Symbolic methods in computational cryptography proofs. In *32nd IEEE Computer Security Foundations Symposium (CSF'19)*, pages 136–151. IEEE Computer Society, June 2019.

[5] Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. Formal certification of code-based cryptographic proofs. *ACM SIGPLAN Notices*, 44(1):90–101, 2009.

[6] Benjamin Bichsel, Timon Gehr, and Martin Vechev. Fine-grained semantics for probabilistic programs. In *27th European Symposium on Programming (ESOP'18)*, volume 10801 of *Lecture Notes in Computer Science*, pages 145–185. Springer, 2018.

[7] Enrico Bombieri. On exponential sums in finite fields. *American Journal of Mathematics*, 88(1):71–105, 1966.

[8] Brent Carmer and Mike Rosulek. Linicrypt: a model for practical cryptography. In *36th Annual International Cryptology Conference (CRYPTO'16)*, volume 9816 of *Lecture Notes in Computer Science*, pages 416–445. Springer, 2016.

[9] Dmitry Chistikov, Andrzej S Murawski, and David Purser. Asymmetric distances for approximate differential privacy. In *30th International Conference on Concurrency Theory (CONCUR 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[10] Benny Chor, Oded Goldreich, Johan Håstad, Joel Friedman, Steven Rudich, and Roman Smolensky. The bit extraction problem of t-resilient functions (preliminary version). In *26th Annual Symposium on Foundations of Computer Science (FOCS'85)*, pages 396–407. IEEE Computer Society, 1985.

[11] Bernard Dwork. On the rationality of the zeta function of an algebraic variety. *American Journal of Mathematics*, 82(3):631–648, 1960.

[12] Graham Everest, Alf van der Poorten, Igor Shparlinski, and Thomas Ward. Exponential functions, linear recurrence sequences, and their applications, 2002.

[13] Matthew Fredrikson and Somesh Jha. Satisfiability modulo counting: A new approach for analyzing privacy properties. In *Joint Meeting of the 23rd Annual Conference on Computer Science Logic (CSL) and the 29th ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10. ACM, 2014.

[14] Marco Gaboardi, Kobbi Nissim, and David Purser. The complexity of verifying circuits as differentially private. *CoRR*, abs/1911.03272, 2019.

[15] William Andrew Johnson. *Fun with fields.* PhD thesis, UC Berkeley, 2016.

[16] Charanjit S Jutla and Arnab Roy. Decision procedures for simulatability. In *17th European Symposium on Research in Computer Security (ESORICS'12)*, volume 7459 of *Lecture Notes in Computer Science*, pages 573–590. Springer, 2012.

[17] Catarina Kiefe. Sets definable over finite fields: their zeta-functions. *Transactions of the American Mathematical Society*, 223:45–59, 1976.

[18] Alan GB Lauder and Daqing Wan. Counting points on varieties over finite fields of small characteristic. *arXiv preprint math/0612147*, 2006.

[19] Axel Legay, Andrzej S Murawski, Joël Ouaknine, and James Worrell. On automated verification of probabilistic programs. In *14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08)*, volume 4963 of *Lecture Notes in Computer Science*, pages 173–187. Springer, 2008.

[20] Rudolf Lidl and Harald Niederreiter. *Finite fields*. Addison-Wesley, 1983.

[21] Yehuda Lindell. How to simulate it - A tutorial on the simulation proof technique. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 277–346. Springer International Publishing, 2017.

[22] Michael L Littman, Judy Goldsmith, and Martin Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998.

[23] Stefan Maubach. The automorphism group over finite fields. 2001.

[24] Scott McCallum and Volker Weispfenning. Deciding polynomial-transcendental problems. *Journal of Symbolic Computation*, 47(1):16–31, 2012.

[25] Maurice Mignotte, Tarlok Nath Shorey, and Robert Tijdeman. The distance between terms of an algebraic recurrence sequence. *Journal für die reine und angewandte Mathematik*, (349):63–76, 1984.

[26] Gary L Mullen and Daniel Panario. *Handbook of finite fields*. Chapman and Hall/CRC, 2013.

[27] Andrzej S. Murawski and Joël Ouaknine. On probabilistic program equivalence and refinement. In Martín Abadi and Luca de Alfaro, editors, *CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings*, volume 3653 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 2005.

[28] Tobias Nipkow. Unification in primal algebras, their powers and their varieties. *J. ACM*, 37(4):742–776, October 1990.

[29] Joël Ouaknine and James Worrell. Decision problems for linear recurrence sequences. In *6th International Workshop on Reachability Problems (RP'12)*, volume 7550 of *Lecture Notes in Computer Science*, pages 21–28. Springer, 2012.

[30] Joël Ouaknine and James Worrell. Positivity problems for low-order linear recurrence sequences. In *25th ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, pages 366–379. Society for Industrial and Applied Mathematics, 2014.

[31] Jacobo Toràn. An oracle characterization of the counting hierarchy. In *3rd Annual Structure in Complexity Theory Conference*, pages 213–223, 1988.

[32] Jacobo Torán. Complexity classes defined by counting quantifiers. *Journal of the ACM*, 38:753–774, 1991.

[33] NK Vereshchagin. The problem of appearance of a zero in a linear recurrence sequence. *Mat. Zametki*, 38(2):609–615, 1985.

[34] André Weil. Numbers of solutions of equations in finite fields. *Bulletin of the AMS*, 1949.

A Proofs

The following Lemma is essentialy a generalization to conditional equivalence of a Lemma from [4].

Lemma 37. *Let* $P_1, Q_1 \in \mathcal{P}_q(I,R)$, $P_2, Q_2 \in \overline{\mathcal{P}}_q(I,R)$. *When* $\sigma : I \to R_I$ *is the substitution that replaces each variable in* I *by a fresh random variable in* R_I, *we have:*

$$P_1 \mid P_2 \approx_{q^k} Q_1 \mid Q_2 \Leftrightarrow (P_1\sigma, R_I) \mid P_2\sigma \approx_{q^k} (Q_1\sigma, R_I) \mid Q_2\sigma$$

Proof. Let $P_1, Q_1 \in \mathcal{P}_q(I,R)$,$P_2, Q_2 \in \overline{\mathcal{P}}_q(I,R)$, we have:

$$\begin{aligned}
&P_1 \mid P_2 \approx_{q^k} Q_1 \mid Q_2 \\
&\Leftrightarrow \forall \vec{i} \in \mathbb{F}_{q^k}^{|I|}.\forall c \in \mathbb{F}_{q^k}^{n} [[(P_1, P_2)]]_{\vec{i}}^{q^k}(c, \vec{0}) = [[(Q_1, Q_2)]]_{\vec{i}}^{q^k}(c, \vec{0}) \\
&\Leftrightarrow \forall t \in \mathbb{F}_{q^k}^{|\{I\}|}.\forall c \in \mathbb{F}_{q^k}^{n} [[(P_1\sigma, P_2\sigma, R_I)]]_{\vec{i}}^{q^k}(c, \vec{0}, t) = [[(Q_1\sigma, Q_2\sigma, R_I)]]_{\vec{i}}^{q^k}(c, \vec{0}, t) \\
&\Leftrightarrow \forall c' \in \mathbb{F}_{q^k}^{n+|\{I\}|} [[((P_1\sigma, R_I), P_2\sigma)]]_{\vec{i}}^{q^k}(c', \vec{0}) = [[((Q_1\sigma, R_I), Q_2\sigma)]]_{\vec{i}}^{q^k}(c', \vec{0}) \\
&\Leftrightarrow (P_1\sigma, R_I) \mid P_2\sigma \approx_{q^k} (Q_1\sigma, R_I) \mid Q_2\sigma
\end{aligned}$$

□

Corollary 38. *For any* $k \in \mathbb{N}$, q^k-equivalence *and* q^k-conditional equivalence *restricted to programs of fixed arity and without inputs are in* $\mathsf{C_=P}$.

Proof. We only have to consider q^k*-conditional equivalence* as it is harder than q^k*-equivalence*. Let $P_1, Q_1 \in \mathcal{P}_q(\emptyset, R)$,$P_2, Q_2 \in \overline{\mathcal{P}}_q(\emptyset, R)$ with $|P_1| = |Q_1| = 1$.

For some c, we have that verifying if

$$[[P_1, P_2]]^{q^k}(c, \vec{0}) = [[Q_1, Q_2]]^{q^k}(c, \vec{0})$$

is in $\mathsf{C_=P}$. As $\mathsf{C_=P}$ is closed under finite intersection [31], we can decide in $\mathsf{C_=P}$ if:

$$\bigwedge_{c \in \mathbb{F}_q} [[P_1, P_2]]^{q^k}(c, \vec{0}) = [[Q_1, Q_2]]^{q^k}(c, \vec{0})$$

This is exactly the definition of conditional equivalence, and thus it concludes the proof. □

Corollary 6. q^k-equivalence *and* q^k-conditional equivalence *are in* $\mathsf{coNP}^{\mathsf{C_=P}}$ *for any* $k \in \mathbb{N}$.

Proof. First, we only have to consider $\mathsf{C{-}EQUIV}_q$ as it is a generalization of equivalence. Next, we only have to consider $\mathsf{C{-}EQUIV}_q$ restricted to program without inputs with Lemma 4. Let $P_1, Q_1 \in \mathcal{P}_q(\emptyset, R)$,$P_2, Q_2 \in \overline{\mathcal{P}}_q(\emptyset, R)$ with $|P_1| = |Q_1| = n$.

Now,

$$P_1 \mid P_2 \approx_{q^k} Q_1 \mid Q_2 \Leftrightarrow \forall c \in \mathbb{F}_{q^k}^{n} [[P_1, P_2]]^{q^k}(c, \vec{0}) = [[Q_1, Q_2]]^{q^k}(c, \vec{0})$$

For some $c \in \mathbb{F}_{q^k}^{n}$, we have that deciding if

$$[[P_1, P_2]]^{q^k}(c, \vec{0}) = [[Q_1, Q_2]]^{q^k}(c, \vec{0})$$

is in $\mathsf{C_=P}$.

The decision problem is then directly inside $\mathsf{coNP}^{\mathsf{C_=P}}$. □

Lemma 7. 2-equivalence *is* $\mathsf{coNP}^{\mathsf{C_=P}}$*-hard.*

Proof. Given a CNF formula $\phi(I,R)$ over two sets of variables and $(\vee, \wedge)$ we set $P = \phi' \in \mathcal{P}_2(I,R)$ obtained according to Lemma 39. Given a fresh random variable r:

$$\begin{aligned}
P(I,R) \approx_2 r &\Leftrightarrow \text{for all valuations of } I, \phi \text{ is true for half of the valuations of } R \\
&\Leftrightarrow \phi(I,R) \in \mathsf{A{-}halfSAT}
\end{aligned}$$

□

Lemma 10. *Let $P_1, \ldots, P_n$ be programs over $\mathcal{P}_{q^k}(I, R)$, and $Y \subset R$.*

$$\perp^{Y}_{q^k} (P_1, \ldots, P_n) \Leftrightarrow \perp_{q^k} (P_1\sigma, \ldots, P_n\sigma)$$

where $\sigma : Y \rightarrow I_Y$ is the substitution that replaces each variable in Y by a fresh input variable in I_Y.

Proof.

$$\begin{aligned}
\perp^{Y}_{q^k} (P_1, \ldots, P_n) &\Leftrightarrow \forall \vec{i} \in \mathbb{F}^{|X|}_{q^k}, \forall \vec{i'} \in \mathbb{F}^{|Y|}_{q^k}.\ [[P_1, \ldots, P_n]]^{q^k}_{\vec{i},\vec{i'}} = ([[P_1]]^{q^k}_{\vec{i},\vec{i'}}, \ldots, [[P_n]]^{q^k}_{\vec{i},\vec{i'}}) \\
&\Leftrightarrow \forall \vec{i} \in \mathbb{F}^{|I \uplus I_Y|}_{q^k}.\ [[(P_1, \ldots, P_n)\sigma]]^{q^k}_{\vec{i}} = ([[P_1\sigma]]^{q^k}_{\vec{i}}, \ldots, [[P_n\sigma]]^{q^k}_{\vec{i}}) \\
&\Leftrightarrow \perp_{q^k} (P_1\sigma, \ldots, P_n\sigma)
\end{aligned}$$

□

Lemma 11. *Let $P_1, \ldots, P_n$ be programs over $\mathcal{P}_{q^k}(I, \{r_1, \ldots, r_m\})$*

$$\perp_{q^k} (P_1, \ldots, P_n) \Leftrightarrow (P_1, \ldots, P_n) \approx_q (P_1\sigma 1, \ldots, P_n\sigma_n)$$

where σ_i is the substitution that to any r_j associates a fresh random variable r^i_j.

Proof.

$$\begin{aligned}
\perp_{q^k} (P_1, \ldots, P_n) &\Leftrightarrow \forall \vec{i} \in \mathbb{F}^{|X|}_{q^k}.\ [[P_1, \ldots, P_n]]^{q^k}_{\vec{i}} = ([[e_1]]^{q^k}_{\vec{i}}, \ldots, [[P_n]]^{q^k}_{\vec{i}}) \\
&\Leftrightarrow \forall \vec{i} \in \mathbb{F}^{|X|}_{q^k}.\ [[P_1, \ldots, P_n]]^{q^k}_{\vec{i}} = ([[P_1\sigma_1]]^{q^k}_{\vec{i}}, \ldots, [[P_n\sigma_n]]^{q^k}_{\vec{i}}) \\
&\Leftrightarrow \forall \vec{i} \in \mathbb{F}^{|X|}_{q^k}.\ [[P_1, \ldots, P_n]]^{q^k}_{\vec{i}} = [[(P_1\sigma_1, \ldots, P_n\sigma_n)]]^{q^k}_{\vec{i}} \\
&\Leftrightarrow (P_1, \ldots, P) \approx_{q^k} (P_1\sigma_1, \ldots, P_n\sigma_n)
\end{aligned}$$

Indeed, for any $\vec{i} \in \mathbb{F}^{|X|}_{q^k}$ we have that $[[P_i]]^{q^k}_{\vec{i}} = [[P_i\sigma_i]]^{q^k}_{\vec{i}}$ as we are only performing renaming. Moreover, $P_1\sigma_1, \ldots, P_n\sigma_n$ do not share any variable, and thus trivially verify:

$$([[P_1\sigma_1]]^{q^k}_{\vec{i}}, \ldots, [[P_n\sigma_n]]^{q^k}_{\vec{i}}) = [[(P_1\sigma_1, \ldots, P_n\sigma_n)]]^{q^k}_{\vec{i}}$$

□

Theorem 13. q^k-conditional independence *is* $\mathsf{coNP}^{\mathsf{C_{=}P}}$*-complete.*

Proof. Only the hardness remains. Given a CNF formula $\phi(I, R)$ over two sets of variables and $(\vee, \wedge)$ we set $P = \phi' \in \mathcal{P}_2(I, R)$ obtained according to Lemma 39. With r a fresh random variable, recall that:

$$\begin{aligned}
P \approx_2 r &\Leftrightarrow \text{for all valuation of } I, \phi \text{ is true for half of the valuation of } R \\
&\Leftrightarrow \phi(I, R) \in \mathsf{A{-}halfSAT}
\end{aligned}$$

But, with x a fresh deterministic variable and r' a fresh random variable:

$$\begin{aligned}
P \approx_2 r &\Leftrightarrow P + x \approx_2 r + x \\
&\Leftrightarrow P + x \approx_2 r \\
&\Leftrightarrow (P + r', r') \approx_2 (r, r') \\
&\Leftrightarrow \perp^{\emptyset}_2 (P + r', r')
\end{aligned}$$

And thus, we conclude with:

$$\perp^{\emptyset}_2 (P + r, r) \Leftrightarrow \phi(I, R) \in \mathsf{A{-}halfSAT}$$

□

Lemma 14. *For any $k \in \mathbb{N}$,* q^k-0-majority *reduces in polynomial time to* q^k-0-majority *with $r = 1$.*

Proof. Indeed, for any n, let us denote D_n any subset of $\mathbb{F}_q^m$, where $m = |\frac{n}{q}|$, such that $|\{D_n\}| = n$. If we denote d_n a fixed element of D_m, let T_n be the program:

$$
\begin{array}{l}
r_1, \ldots, r_m \xleftarrow{\$} \{x \in \mathbb{F}_q^m \mid \bigvee_{d \in D_n} x = d\} \\
\text{if } (r_1, \ldots, r_m) = d_n \text{ then} \\
\quad \text{return } \vec{0} \\
\text{else} \\
\quad \text{return } \vec{1}
\end{array}
$$

Notice that by construction $[[T_n]]^{q^k}(\vec{0}) = \frac{1}{n}$. This is only the most naive version of this encoding, simpler polynomials can be found for many specific cases. And finally, for any $r, s \in \mathbb{N}$, assuming the probabilities are non zero, we have:

$$
\begin{aligned}
\forall \vec{i} \in \mathbb{F}_{q^k}^{|I|}.\ \frac{[[P]]_{\vec{i}}^{q^k}(\vec{0})}{[[Q]]_{\vec{i}}^{q^k}(\vec{0})} \leq \frac{r}{s} &\Leftrightarrow \forall \vec{i} \in \mathbb{F}_{q^k}^{|I|}.\ \frac{[[P]]_{\vec{i}}^{q^k}(\vec{0})}{r} \leq \frac{[[Q]]_{\vec{i}}^{q^k}(\vec{0})}{s} \\
&\Leftrightarrow \forall \vec{i} \in \mathbb{F}_{q^k}^{|I|}.\ [[P]]_{\vec{i}}^{q^k}(\vec{0})[[T_r]]_{\vec{i}}^{q^k}(\vec{0}) \leq [[Q]]_{\vec{i}}^{q^k}(\vec{0})[[T_s]]_{\vec{i}}^{q^k}(\vec{0}) \\
&\Leftrightarrow \forall \vec{i} \in \mathbb{F}_{q^k}^{|I|}.\ [[(P, T_r)]]_{\vec{i}}^{q^k}(\vec{0}) \leq [[(Q, T_s)]]_{\vec{i}}^{q^k}(\vec{0}) \\
&\Leftrightarrow (P, T_r) \prec_{q^k} (Q, T_s)
\end{aligned}
$$

□

Lemma 15. *For any $k \in \mathbb{N}$, q^k-0-majority restricted to inputless programs is* PP*-complete.*

Proof. Membership
Let $P, Q \in \mathcal{P}_q(\emptyset, R)$. Let us reuse the polynomial time Turing Machine M defined in Lemma 5. Given P_1, P_2, Q_1, Q_2 and $\vec{c}$, it was such that:

$$[[P_1, P_2]]^{q^k}(\vec{c}, \vec{0}) = [[Q_1, Q_2]]^{q^k}(\vec{c}, \vec{0}) \Leftrightarrow M \text{ accepts exactly half of the time}$$

Now, by replacing equals by $>$ signs in the proof, we directly have that:

$$[[P_1, P_2]]^{q^k}(\vec{c}, \vec{0}) \leq [[Q_1, Q_2]]^{q^k}(\vec{c}, \vec{0}) \Leftrightarrow M \text{ accepts at least half of the time}$$

Thus, we do have:

$$
\begin{aligned}
P \prec_{q^k} Q &\Leftrightarrow [[P, 0]]^{q^k}(\vec{0}, 0) \leq [[Q]]^{q^k}(0, \vec{0}) \\
&\Leftrightarrow M \text{ accepts at least half of the time on input } (P, 0, Q, 0, \vec{0})
\end{aligned}
$$

Hardness
We show PP-hardness by reduction from MAJSAT. Given a CNF formula $\phi(R)$ over two sets of variables and $(\vee, \wedge)$ we set $P = \phi' \in \mathcal{P}_2(R)$ obtained according to Lemma 39. We then have:

$$
\begin{aligned}
\phi \in \text{MAJSAT} &\Leftrightarrow \left|\{X \in \mathbb{F}_2^m \mid P(X) = \vec{0}\}\right| \leq 2^{m-1} \\
&\Leftrightarrow \left|\{X \in \mathbb{F}_2^m \mid P(X) = \vec{0}\}\right| \leq \left|\{X \in \mathbb{F}_2^m \mid x_1 = 0\}\right| \\
&\Leftrightarrow P \prec_2 x_1
\end{aligned}
$$

□

Lemma 16. *q^k-majority is* coNP$^{\text{PP}}$ *complete.*

Proof. Hardness Let ϕ a CNF formula built over two sets of variables I and R. We use the same construction as in Lemma 15 to obtain a polynomial $P \in \mathcal{P}_2(I, R)$ whose truth value is equivalent of ϕ.

We have, for some variable r inside R:

$$\phi \in \text{A–MINSAT} \Leftrightarrow r \prec_2 P$$

Membership

Let $P, Q \in \mathcal{P}_{q^k}(I, R)$. We slightly modify M from Lemma 15, so that it takes as extra argument a valuation for the variables in I, and every evaluation of P or Q is made according to the valuation.

Then, we directly have:

$$P \prec_q Q \Leftrightarrow \forall \vec{i} \in \mathbb{F}_p^{|I|}, M \text{ accepts with probability greater than half on input } \vec{i}$$

This problem is then directly inside $\mathsf{coNP}^{\mathsf{PP}}$. □

Lemma 19. q^∞-equivalence *restricted to linear programs is in* PTIME.

Proof. Without loss of generality, we only consider programs without input variables (Lemma 4).

Given a set of variables R, we assume that there is an ordering over the variables in R. We say that an expression is in normal form if it is of one of the following form: 0 or 1, or e, or $1 \oplus e$, where e is built from variables and $\oplus$ (but no constants), and variables appear at most once in increasing order.

Every linear expression can easily be put in normal form, using the commutativity of $\oplus$, and the normal form is indeed unique thanks to the ordering on variables.

We now assume that all polynomials are in normal form.

Given $P_1, \ldots, P_n \in \overline{\mathcal{P}}_q(\emptyset, R)$ without multiplications, we iterate over each P_i, where, after initializing a set S to the emptyset:

- if $\mathsf{vars}(P_i) \cap S \neq \emptyset$, let $r = \min(\mathsf{vars}(P_i) \cap S)$ and:
  - replace P_i by r;
  - set $S := S \cup \{r\}$;
  - for each $j \geq i$, replace P_j by $P_j[P_i \oplus r/r]$.
- else, continue.

This produces a normal form for any tuple $(P_1, \ldots, P_n)$, where each P_i is either a fresh random variable (not appearing in the previous Ps), or a linear combination of the previous $P_1, \ldots, P_{i-1}$.

Finally, two programs are universally equivalent if and only if they have the same normal form (up to α-renaming). Indeed, if they have the same normal form, they are trivially universally equivalent. Now, if they do not have the same normal form, there exists some i such that P_i and Q_i are two different expressions, and this imply non equivalence.

This basic decision prodcedures gives us a $\mathcal{O}(n \times |R|)$ complexity. Indeed, we treat each polynomial P_i or Q_i only once, first to apply the currenly known substitutions, and then to transform it into a fresh random if required. Applying the currently known susbtitutions may take up to $|R|$ loops, hence the considered complexity. □

Corollary 25. q^∞-equivalence *and* q^∞-conditional equivalence *restricted to arithmetic programs are in* EXP.

Proof. [18, Corollary 2] provides a precise complexity for the evaluation of $Z(P)$. They provide an algorithm to compute $Z(P)$ for which there exist an explicit polynomial R such that it runs in time $R(p^m k^m d^{m^2} 2^n))$, where d is the sum of the degrees of the P^i. It is then polynomial in the degrees of the polynomials and the size of the finite fields, but exponential in the number of variables. In our, case, we need to compute three times Z, on polynomials depending over $2m$ variables (has we duplicate variables), which gives us en exponential in the size of our arithmetic programs.

□

Lemma 39. *Given a CNF formula $\phi(I, R)$ over two sets of variables and $(\vee, \wedge)$, we can produce in polynomial time a program $P \in \mathcal{P}_2(I, R)$ equivalent to ϕ.*

Proof. Given a CNF formula $\phi(I, R)$ over two sets of variables and $(\vee, \wedge)$ we transform ϕ into an equivalent formula ϕ' over $I \uplus R$ and $\oplus, \wedge$ in polynomial time w.r.t the size of the formula. Indeed, given a clause of ϕ of the form $x \vee y \vee z$, we have that $x \vee y \vee z = (x \oplus y \oplus xy) \vee z = (x \oplus y \oplus xy) \oplus z \oplus (x \oplus y \oplus xy)z = x \oplus y \oplus xy \oplus z \oplus xz \oplus yz \oplus xyz = x \oplus y \oplus z \oplus xy \oplus yz \oplus xz \oplus xyz$. With this transformation, we have $|\phi'| \leq 5 \times |\phi|$.

And then, $P = \phi' \in \mathcal{P}_2(I, R)$ is a program equivalent to ϕ. □

Lemma 40. *Let* $P, Q \in \overline{\mathcal{P}}_2(\emptyset, R)$ *without any multiplication.*

$$P \approx_2 Q \Leftrightarrow P \approx_{2^\infty} Q$$

Proof.
$\Leftarrow$ Trivial direction.
$\Rightarrow$ As outlined in [3], one can decide if $P \approx_2 Q$ by constructing a bijection represented by only linear terms (thanks to the weak primality of $\mathbb{F}_2$ restricted to addition). We thus have a bijection σ without multiplication such that $P = Q\sigma$. σ is then a bijection over all $\mathbb{F}_{2^k}$, and we do have $P \approx_{2^\infty} Q$.

□

Lemma 41. *Let b be a propositional formula built over built over atoms of the form $B = 0$ or $B \neq 0$ with $B \in \mathbb{F}_q[X]$. There exists $X' \supset X$ and polynomials $B_1, \ldots, B_n \in \mathbb{F}_q[X']$ so that:*

$$\left|\{X \in \mathbb{F}_{q^k}^m \mid b\}\right| = \left|\{X' \in \mathbb{F}_{q^k}^m \mid \bigwedge_{1 \leq i \leq n} B_i = 0\}\right|$$

Those polynomials can be computed in exponential time.

Proof. We prove by induction of the formula that for any formula b, there exists polynomials $B_1, \ldots, B_n$ so that:

$$\left|\{X \in \mathbb{F}_{q^k}^m \mid b\}\right| = \left|\{X' \in \mathbb{F}_{q^k}^m \mid \bigwedge_{1 \leq i \leq n} B_i = 0\}\right|$$

We will assume that the formula are in conjunctive normal form, hence the exponential time. $b := B = 0$
Direct, with $X' = X$ and $B_1 = B$.
$b := B' \neq 0$ For any k and c we have that:

$$\left|\{X \in \mathbb{F}_{q^k}^m \mid B \neq 0\}\right| = \left|\{X \in \mathbb{F}_{q^k}^m, t \in \mathbb{F}_{q^k} \mid tB - 1 = 0\}\right|$$

Indeed, B is different from zero if and only if it is invertible, and thus if and only if there exist a single value t such that $tB = 1$.
$b := \bigvee_{1 \leq i \leq l} B_i = 0$

$$\left|\{X \in \mathbb{F}_{q^k}^m \mid \bigvee_{1 \leq i \leq l} B_i = 0\}\right| = \left|\{X \in \mathbb{F}_{q^k}^m \mid (\prod_{1 \leq i \leq l} B_i) = 0\}\right|$$

$b := \bigwedge_{1 \leq i \leq k} b_i$ By induction hypothesis on each b_i we get $B_1^i, \ldots, B_{n_i}^i$ so that all of them verify:

$$\left|\{X \in \mathbb{F}_{q^k}^m \mid b\}\right| = \left|\{X \in \mathbb{F}_{q^k}^m, t \in \mathbb{F}_{q^k} \mid \bigwedge_{1 \leq i \leq k} \bigwedge_{1 \leq j \leq n_i} B_j^i = 0\}\right|$$

$b := b_1 \vee b_2$ By induction hypothesis on b_1 we get $B_1, \ldots, B_n$, and on b_2 $B_1', \ldots, B_n'$, which satisfies

$$\left|\{X \in \mathbb{F}_{q^k}^m \mid b\}\right| = \left|\{X \in \mathbb{F}_{q^k}^m, t \in \mathbb{F}_{q^k} \mid \bigwedge_{1 \leq i \leq n} B_i = 0 \vee \bigwedge_{1 \leq i \leq n} B_i' = 0\}\right|$$

□

Lemma 26. *For any $k \in \mathbb{N} \cup \{\infty\}$, q^k-*conditional equivalence *restricted to programs without failures reduces in exponential time to q^k-*conditional equivalence *restricted to arithmetic programs.*

Proof. Let $P_1, Q_1 \in \mathcal{P}_q(\emptyset, R)$,$P_2, Q_2 \in \overline{\mathcal{P}}_q(\emptyset, R)$, without failures.

We reason by induction on the total number n of conditional branching inside P_1 and Q_1. By basic transformations of the conditionals, we can assume that all conditions are of the form $B \neq 0$ (one can easily encode negations, conjunction and disjunction using conditionals branching).

$\underline{n = 0}$ If there are no conditionals branching, the result is trivial.

$\underline{n > 1}$ We consider one of the inner most branching inside P_1, i.e $P_1 := C[\text{if } B \neq 0 \text{ then } P_1^t \text{ else } P_1^f]$ for some context C, and P_1^t, P_1^f arithmetic programs.

For a fixed k, we have a classical encoding of the if then else inside polynomials (cf CSF19):

$$[\![\text{if } B \neq 0 \text{ then } P_1^t \text{ else } P_1^f]\!]^{q^k} = [\![P_1^f + B^{q^k-1}(P_1^t - P_1^f)]\!]^{q^k}$$

We then have that:

$$C[\text{if } B \neq 0 \text{ then } P_1^t \text{ else } P_1^f] \mid P_2 \approx_{q^k} Q_1 \mid Q_2 \Leftrightarrow C[P_1^f + B^{q^k-1}(P_1^t - P_1^f)] \mid P_2 \approx_{q^k} Q_1 \mid Q_2$$

A difficulty of this encoding is that it depends on the k, so it cannot be lifted to universal conditional equivalence. However, we can remove this difficulty by using an extra variable t to encode the B^{q^k-1}.

With t a fresh variable, we denote

$$\text{ite}(B, P_1^t, P_1^f) = (P_1^f + tB(P_1^t - P_1^f), B(Bt-1), t(Bt-1))$$

Now, for any k and c we have that:

$$\begin{aligned}
&[\![(P_1^f + B^{q^k-1}(P_1^t - P_1^f), P_2)]\!]^{q^k}(c, \vec{0}) \\
&\quad = \left|\{X \in \mathbb{F}_{q^k}^m \mid P_1^f + B^{q^k-1}(P_1^t - P_1^f) = c \wedge P_2 = \vec{0}\}\right| \times \frac{1}{\left|\{\mathbb{F}_{q^k}^m\}\right|} \\
&\quad = \left|\{X \in \mathbb{F}_{q^k}^m, t \in \mathbb{F}_{q^k} \mid \text{ite}(B, P_1^t, P_1^f) = (c, \vec{0}) \wedge P_2 = \vec{0}\}\right| \times \frac{1}{\left|\{\mathbb{F}_{q^k}^m\}\right|}
\end{aligned}$$

Indeed, for any variable t and polynomial B:

$$(B(Bt-1) = 0 \wedge t(Bt-1) = 0) \Leftrightarrow t = B^{q^k-2}$$

Finally:

$$\begin{aligned}
&[\![(P_1^f + B^{q^k-1}(P_1^t - P_1^f), P_2)]\!]^{q^k}(c, \vec{0}) \\
&\quad = \left|\{X \in \mathbb{F}_{q^k}^m, t \in \mathbb{F}_{q^k} \mid \text{ite}(B, P_1^t, P_1^f) = (c, \vec{0}) \wedge P_2 = \vec{0}\}\right| \times \frac{1}{\left|\{\mathbb{F}_{q^k}^m\}\right| + \left|\{\mathbb{F}_q\}\right|} \\
&\quad = [\![(P_1^f + tB(P_1^t - P_1^f), B(Bt-1), t(Bt-1), P_2)]\!]^{q^k}(c, \vec{0})
\end{aligned}$$

Putting everything together, we get that:

$$\begin{aligned}
C[\text{if } B \neq 0 \text{ then } P_1^t \text{ else } P_1^f] \mid P_2 \approx_{q^k} Q_1 \mid Q_2 &\Leftrightarrow C[P_1^f + B^{q^k-1}(P_1^t - P_1^f)] \mid P_2 \approx_{q^k} Q_1 \mid Q_2 \\
&\Leftrightarrow C[P_1^f + tB(P_1^t - P_1^f)] \mid (B(Bt-1), t(Bt-1), P_2) \approx_{q^k} Q_1 \mid Q_2
\end{aligned}$$

And we finally have:

$$P_1 \mid P_2 \approx_{q^\infty} Q_1 \mid Q_2 \Leftrightarrow C[P_1^f + tB(P_1^t - P_1^f)] \mid (B(Bt-1), t(Bt-1), P_2) \approx_{q^\infty} Q_1 \mid Q_2$$

The conditional equivalence on the right-side contains strictly one less conditional, we thus conclude by induction hypothesis.

Conclusion We have shown by induction that we can remove all conditional branching. Each removal produces a new instance of polynomial size, and there is necessarily a polynomial number of conditional branching inside the programs. We thus reduces in exponential time $\mathsf{C{-}EQUIV}_{q^\infty}$ to $\mathsf{C{-}EQUIV}_{q^\infty}$ over programs without conditionals (recall that removing the failure cost an exponential). □

Lemma 27. *For any* $k \in \mathbb{N} \cup \{\infty\}$*,* q^k-conditional equivalence *reduces to* q^k-conditional equivalence *restricted to programs without failures in exponential time.*

Proof. Let $P_1, Q_1 \in \mathcal{P}_q(\emptyset, R)$,$P_2, Q_2 \in \overline{\mathcal{P}}_q(\emptyset, R)$.

Recall that observe are expressed using conditionals with a failure branch, and that sampling inside some specific set can be encoded using the observe primitive. Without loss of generality, we can consider that $\perp$ appears only once, as we can merge the conditions of the different failure branches inside a single one.

Then, P_1 is of the form $P_1 := \texttt{if } b \texttt{ then } P_1^t \texttt{ else } \perp$ for some program P_1^t which cannot fail.

Now, with Lemma 41, we have $R' \supset R$ and polynomials $B_1, \dots, B_n \in \mathbb{F}_q[R']$ so that:

$$\left|\{R \in \mathbb{F}_{q^k}^m \mid b\}\right| = \left|\{R' \in \mathbb{F}_{q^k}^m \mid \bigwedge_{1 \le i \le n} B_i = 0\}\right|$$

And then:

$$\begin{aligned}
&[[(\texttt{if } b \texttt{ then } P_1^t \texttt{ else } \perp, P_2)]]^{q^k}(\vec{c}, \vec{0}) \\
&\quad = \frac{\mathbb{P}\{P_1^t = \vec{c} \wedge P_2 = \vec{0} \wedge b|\}}{\mathbb{P}\{\neg b|\}} \\
&\quad = \left|\{R' \in \mathbb{F}_{q^k}^m \mid P_1^t = \vec{c} \wedge P_2 = \vec{0} \wedge \textstyle\bigwedge_{1 \le i \le n} B_i = 0\}\right| \times \frac{1}{\left|\{R' \in \mathbb{F}_{q^k}^m \mid \neg \bigwedge_{1 \le i \le n} B_i = 0\}\right|} \\
&\quad = \left|\{R' \in \mathbb{F}_{q^k}^m \mid P_1^t = \vec{c} \wedge P_2 = \vec{0} \wedge \textstyle\bigwedge_{1 \le i \le n} B_i = 0\}\right| \times \frac{1}{\left|\{R' \in \mathbb{F}_{q^k}^m, t_i \in \mathbb{F}_q \mid \prod_{1 \le i \le n} (t_i B_i - 1) = 0\}\right|}
\end{aligned}$$

This allows us to conclude, when σ maps random variables to fresh ones, that:

$$\texttt{if } b \texttt{ then } P_1^t \texttt{ else } \perp \mid P_2 \approx_{q^k} Q_1 \mid Q_2 \Leftrightarrow P_1^t \mid P_2, B_1, \dots, B_n \approx_{q^k} Q_1 \mid Q_2, \textstyle\prod_{1 \le i \le n} (t_i B_i \sigma - 1)$$

We thus removed the failure on the left side of the conditional equivalence. Proceeding similarly on the right side yield the expected result. □

Lemma 30. q-equivalence *reduces in polynomial time to* q^∞-equivalence.

Proof. Let $P, Q \in \mathcal{P}_q(\emptyset, \{r_1, \dots, r_m\})$. We directly have:

$$\begin{aligned}
P \approx_q Q \Leftrightarrow \quad & \left|\{X \in \mathbb{F}_q^m \mid P(X) = \vec{0}\}\right| = \left|\{X \in \mathbb{F}_q^m \mid Q(X) = \vec{0}\}\right| \\
\Leftrightarrow \quad & \texttt{if } \textstyle\bigwedge_{1 \le i \le m} (\bigvee_{c \in \mathbb{F}_q} r_i = c) \texttt{ then } P \texttt{ else } \vec{0} \\
& \qquad \approx_{2^\infty} \\
& \texttt{if } \textstyle\bigwedge_{1 \le i \le m} (\bigvee_{c \in \mathbb{F}_q} r_i = c) \texttt{ then } Q \texttt{ else } \vec{0}
\end{aligned}$$

□

Lemma 42. *Given* $l \in \mathbb{N}$*, with* $n = \sharp I + \sharp R$*,*

$$\left|\{C \in \mathcal{P}_q I, R \mid \deg(C) \le l\}\right| \le (q^{l^n})^{q^{l^n}}$$

Proof. There exists l^n possible monomials (choosing the degree of each variable). Choosing the coefficient in $\{0, \dots, q-1\}$ for each monomials yeilds that the number of polynomials is bounded by q^{l^n}. A program can, for each possible polynomial, performs a branching over it. There exists thus q^{l^n} possible conditions, which when true may yield a polynomial (q^{l^n} possible choices) or $\perp$. We finally obtain the expected result. □

Lemma 33. *For any* $l \in \mathbb{N}$*,* $k \in \mathbb{N} \cup \{\infty\}$*,* q^k-equivalence *reduces in polynomial time to* l, q^k-simulatability.

Proof. Let $P, Q \in \mathcal{P}_q(I, R)$. Given two fresh variable a and b, we set $I' = I \uplus \{a, b\}$. As previously, we lift additions and multiplications to tuples. Then,

$$P \approx_{q^k} Q \Leftrightarrow a + bP \sqsubseteq^1_{q^k} a + bQ$$

Indeed, if $P \approx_{q^k} Q$, then we trivially have $a + bP \sqsubseteq^1_{q^k} a + bQ$ with S as the identity. Let us assume that we have $S \in \mathcal{P}_q(\{i\}, R')$ such that $\deg(S) \leq l$ and $S[^{a+bQ}/_i] \approx_{q^k} a + bP$. We actually have the equivalence for any possible values we choose to give to a and b. For instance, with $b = 0$, we get that $S[^a/_i] \approx_{q^k} a$, which direclty implies that S is the identity. Thus, with $a = 0$ and $b = 1$, we have that $P \approx_{q^k} Q$. This concludes the proof. □

Lemma 34. q^∞*-0-*majority *restricted to linear programs is in* PTIME.

Proof. We show that for linear programs $P \prec^r_{q^k} Q$ implies that $P \approx_{q^k} Q$. Thus, universal majority is decidable, as universal equivalence is decidable for linear programs (and in PTIME).

Given $P_1, \dots, P_n \in \overline{\mathcal{P}}_q(\emptyset, R)$ without multiplications, let us consider once again the normal form for linear programs. In this normal form, each P_i is either a random r_i, or a linear combination of some r_j, with $j < i$. Let I_P be the set of indices i such that $P_i = r_i$. We denote $P = (P_1, \dots, Pn)$, and given $\vec{c} \in \mathbb{F}^n_{q^k}$, we have that $[[P]]^{q^k}(\vec{c}) = \begin{cases} \frac{1}{q^{k \times |I_P|}} & \text{if the linear constraints are satisfiable} \\ 0 & \text{else} \end{cases}$ Indeed, $\vec{c}$ imposes the values of each r_i for $i \in I$, and then for those values, either the other elements of the program coincides, and if they do not, the program is never equal to $\vec{c}$.

Let $P, Q \in \overline{\mathcal{P}}_q(\emptyset, R)$ without multiplications, we know that:

1. $\forall \vec{c} \in \mathbb{F}^n_{q^k}$, $[[P]]^{q^k}(\vec{c}) = \frac{1}{q^{k \times |I_P|}}$ or 0
2. $\forall \vec{c} \in \mathbb{F}^n_{q^k}$, $[[Q]]^{q^k}(\vec{c}) = \frac{1}{q^{k \times |I_Q|}}$ or 0
3. $\sum_{\vec{c} \in \in \mathbb{F}^n_{q^k}} [[P]]^{q^k}(\vec{c}) = \sum_{\vec{c} \in \in \mathbb{F}^n_{q^k}} [[Q]]^{q^k}(\vec{c})$

Now, let us assume that there exists *vecc* such that $[[P]]^{q^k}(\vec{c}) = 0$ and $[[Q]]^{q^k}(\vec{c}) \neq 0$. Then, for any r, we have $Q \not\prec^r_{q^k} P$. Moreover, if for all $\vec{c'} \neq \vec{c}$, $[[P]]^{q^k}(\vec{c'}) = 0$ or $[[Q]]^{q^k}(\vec{c'}) \neq 0$, it yields a contradiction with Hypothesis (3). Thus, there exists $\vec{c'}$ such that $[[P]]^{q^k}(\vec{c'}) \neq 0$ and $[[Q]]^{q^k}(\vec{c'}) = 0$. This also implies that for all r, $P \not\prec^r_{q^k} Q$.

Let us assume that forall k, $P \prec^r_{q^k} Q$. Then, by the previous developpment, we know that for all $\vec{c}$, $[[P]]^{q^k}(\vec{c}) \neq 0$ and $[[Q]]^{q^k}(\vec{c}) \neq 0$. If $|I_P| \neq |I_Q|$, it would yield a contradiction with Hypothesis (3). We thus conclude that $|I_P| = |I_Q|$, and based on Hypothesis (1) and (2), we have that forall $\vec{c}$, $[[P]]^{q^k}(\vec{c}) = [[P]]^{q^k}(\vec{c})$. We thus conclude that $P \approx_{q^k} Q$.

We have proven that $P \prec^r_{q^\infty} Q \Leftrightarrow P \approx_{q^\infty} Q$, when restricted to linear programs without multiplications. □

Lemma 36. 2^∞-0-majority *is* PP*-hard.*

Proof. We prove that 2-0-*majority* reduces to 2^∞-0-*majority* in polynomial time.

Let $P, Q \in \mathcal{P}_2(\emptyset, R)$.

$$
\begin{aligned}
P \prec_2 Q &\Leftrightarrow \left|\{X \in \mathbb{F}_2^m \mid P(X) = \vec{0}\}\right| \leq \left|\{X \in \mathbb{F}_2^m \mid q(X) = \vec{0}\}\right| \\
&\Leftrightarrow \forall k, \left|\{X \in \mathbb{F}_{2^k}^m \mid P(X) = \vec{0} \wedge X \in \mathbb{F}_2^m\}\right| \leq \left|\{X \in \mathbb{F}_{2^k}^m \mid Q(X) = \vec{0} \wedge X \in \mathbb{F}_2^m\}\right| \\
&\Leftrightarrow \forall k, \left|\{X \in \mathbb{F}_{2^k}^m \mid P(X) = \vec{0} \wedge x_1(x_1+1) = 0 \wedge \cdots \wedge x_m(x_m+1) = 0\}\right| \\
&\qquad\qquad \leq \left|\{X \in \mathbb{F}_{2^k}^m \mid Q(X) = \vec{0} \wedge x_1(x_1+1) = 0 \wedge \cdots \wedge x_m(x_m+1) = 0\}\right| \\
&\Leftrightarrow \forall k, \left|\{X \in \mathbb{F}_{2^k}^m \mid (P(X), x_1(x_1+1), \ldots, x_m(x_m+1)) = \vec{0}\}\right| \\
&\qquad\qquad \leq \left|\{X \in \mathbb{F}_{2^k}^m \mid (Q(X), x_1(x_1+1), \ldots, x_m(x_m+1)) = \vec{0}\}\right| \\
&\Leftrightarrow (P, x_1(x_1+1), \ldots, x_m(x_m+1)) \prec_2^\infty (Q, x_1(x_1+1), \ldots, x_m(x_m+1))
\end{aligned}
$$

□