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, MPI-SP & IMDEA Software Institute
CHARLIE JACOMME, CISPA Helmholtz Center for Information Security
STEVE KREMER, LORIA, Inria Nancy-Grand Est & CNRS & Université de Lorraine

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. We then devise a general way to draw links between the universal probabilistic problems and widely studied problems on linear recurrence sequences. Finally we study several variants of the equivalence problem, including a problem we call majority, motivated by differential privacy. We also define and provide some insights about program indistinguishability, proving that it is decidable for programs always returning 0 or 1.

CCS Concepts: • **Security and privacy → Logic and verification**; • **Theory of computation → Program reasoning**.

Additional Key Words and Phrases: program equivalence, probabilistic programs, finite fields, decidability and complexity

ACM Reference Format:
Gilles Barthe, Charlie Jacomme, and Steve Kremer. 2021. Universal equivalence and majority of probabilistic programs over finite fields. *ACM Trans. Comput. Logic* 1, 1 (August 2021), 41 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Program equivalence is one of the most fundamental tools in the theory of programming languages and arguably the most important example of a 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

Authors' addresses: Gilles Barthe, MPI-SP & IMDEA Software Institute; Charlie Jacomme, CISPA Helmholtz Center for Information Security; Steve Kremer, LORIA, Inria Nancy-Grand Est & CNRS & Université de Lorraine.

1529-3785/2021/8-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

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 $[[P_1]]^{\mathbb{F}_{q^k}}, [[P_2]]^{\mathbb{F}_{q^k}} : \mathbb{F}_{q^k}^m \to \mathsf{Distr}(\mathbb{F}_{q^k}^n)$.

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

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

  $\mathbb{F}_{q^\infty}$-equivalence requires the property to hold on all extensions of a field, i.e.,

$$P_1 \approx_{\mathbb{F}_{q^\infty}} P_2 \quad \text{iff} \quad \forall k.\ P_1 \approx_{\mathbb{F}_{q^k}} P_2$$

- $\mathbb{F}_{q^k}$*-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

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

  This property allows us to use r to bound the distance between the two distributions. $\mathbb{F}_{q^k}$*-0-majority* (denoted $P_1 \prec^r_{\mathbb{F}_{q^k}} P_2$) is a variant of majority, where we only consider the output $b = 0^n$, rather than quantifying over all outputs. $\mathbb{F}_{q^\infty}$*-0-majority* requires the property to hold on all extensions of a field, i.e.,

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

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

Example 1.1.

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

are 2- but not 2^2-*equivalent*, and hence not $\mathbb{F}_2^\infty$-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

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

are $\mathbb{F}_{q^\infty}$-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.
- δ-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, that is a program with a specific input variable that can be instantiated by the return value of another program, 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 some expression Z built over the variables of the program? Although independence is not naturally expressed as a relational property, it has been shown in [5] 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 $\mathbb{F}_{q^k}$-*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 from 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 [19], see Related Work below.

The second, and main contribution, is the study of universal equivalence, $\mathbb{F}_{q^\infty}$-equivalence for short, and universal (0-)majority, $\mathbb{F}_{q^\infty}$-(0-)majority for short. First, we show that the $\mathbb{F}_{q^\infty}$-*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 characterizes the number of zeros of a tuple of polynomials in all extensions of a finite field. Lauder and Wan [24] notably propose an algorithm to compute such functions, whose complexity is however exponential. Interestingly, this local zeta function also provides us with a way to reduce our problems to problems over Linear Recurrence Sequences (LRS)[2]: properties of the local zeta function imply that the sequence of number of zeros of a tuple of polynomials over each extension of a finite field is a LRS.

[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.
[2] An LRS is a sequence of integers satisfying a linear recurrence relation.

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

Fig. 1. Summary of results related to equivalence

Based on this result, our proof proceeds in two 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, or equivalently that two LRS are equal. 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 simpler methods fail or only provide sufficient conditions. Our decidability result significantly generalizes prior work on universal equivalence [4], 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 [31] and by Vereshchagin [41] for LRS of order ≤ 4 and later by Ouaknine and Worrell [36] 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 long standing open problems in Diophantine approximation. In the general case, the best known lower bound for the positivity problem is NP-hardness [35].

Unfortunately, the order of the linear recurrence 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. We remark that the LRS obtained from the majority problem is *simple*, and we can thus decide a close problem, which is ultimate positivity (is the LRS always positive after some point), that was proven decidable in [37]. Using the results from [23], we observe that the reduction extends to a more general form of universal majority problem.

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.

As side contributions, we provide some first tentative attempts for verifying program indistinguishability, that can be seen as an approximate universal equivalence. We define and prove the decidability of the LRS negligibility problem, but leave the question of program indistinguishability open. We however obtain the decidability of program indistinguishability for programs that only return 0 or 1. Finally, we also prove that enriching the programming language with loops makes the universal equivalence problem undecidable over finite fields.

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

Preliminary version. A preliminary version of this work appeared in [7]. In comparison, the decidability proof of universal equivalence has been improved and generalized, thanks to the link to LRS. We also introduce the indistinguishability problem for programs and show its decidability for programs that return a boolean. Finally, we also provide in this paper the proof that universal equivalence is undecidable when extending the programming language with loops.

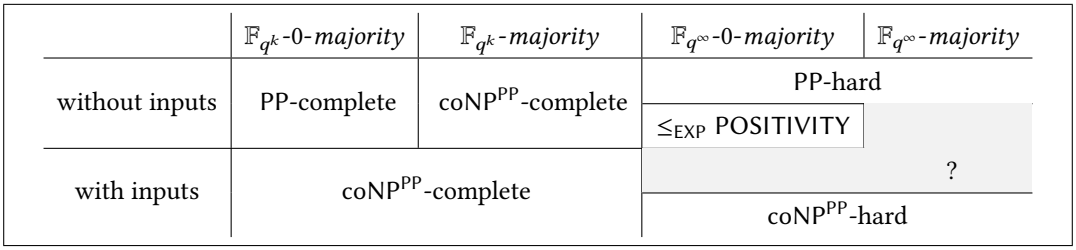

Fig. 2. Summary of results related to majority

We remark that independently, the complexity of universal equivalence has been improved by [13]. The authors improve the complexity of the local zeta function computation w.r.t. the number of polynomials, and this directly yields an improvement to our results.

Related work

Universal equivalence. The case of linear programs is studied in [4]. The authors propose a decision procedure for universal equivalence based on the classic XOR-lemma [15]. We give an alternative decision procedure and analyze its complexity.

The case of linear programs with random oracles is considered in [11]. 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 [22]. 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 4.4, we show that universal equivalence reduces to universal simulatability. As [22] 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 [33] 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* [25] 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* [6] 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 [18] 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 [3] 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 [30].

Gaboardi, Nissim and Purser [19] 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 [14] also study the complexity of approximating differential privacy, but in the case of Markov Chains.

Theory of fields. A celebrated result by Ax [2] shows that the theory of finite fields is decidable. In a recent development based on Ax's result, Johnson [21] 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.

2.1 Syntax and informal semantics

We define in Figure 3 the syntax for simple probabilistic programs (without loops nor recursion). 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 $P \in \mathbb{F}_q[x_1, \ldots, x_m]$ 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 $\vec{r} = r_1, \ldots, r_m$, the expression $r_1, \ldots, r_m \xleftarrow{\$} \{X \in \mathbb{F}^m \mid b\}$ 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 $r_1, \ldots, r_m$. 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.

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

Fig. 3. Program syntax

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 $\mathcal{P}_{\mathbb{F}_q}(I, R)$ 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 2.1. 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 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 .

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

Example 2.2. We also illustrate the suitability of this language on a simple security protocol. Shamir's secret sharing scheme [38] is a (t, n) threshold scheme: the protocol splits a secret s between n participants in such a way that s can be computed from any t shares; however no information about s can be obtained with less shares.

The idea of the protocol is to construct a polynomial

$$f(x) = s + a_1 x + \cdots + a_{t-1} x^{t-1}$$

where coefficients a_i are randomly sampled in a finite field. The shares distributed to the participants are distinct pairs $(p_j, f(p_j))$ where $p_j \neq 0$ are elements of the field for $1 \leq j \leq n$. Given at least t such pairs it is easy to recompute f by interpolation and recover $s = f(0)$.

Taking the attacker's point of view we write the program that, given $p_1, \ldots, p_{t-1}$ points in the finite field, returns $f(t_1), \ldots, f(t_{p-1})$. (We also note that the program is independent of n, as we are interested in showing the security when the attacker may query $t-1$ shares.)

$$\begin{array}{l}
\texttt{t-n-SSS}(p_1, \ldots, p_{t-1}) = \\
\quad s \xleftarrow{\$} \mathbb{F}; \qquad\qquad \text{// sample random secret s} \\
\quad a_1, \ldots, a_{t-1} \xleftarrow{\$} \mathbb{F}^{t-1}; \qquad \text{// sample } t-1 \text{ random coefficients} \\
\quad \mathsf{if}\ \bigwedge_{1 \leq i \leq t-1} p_i \neq 0\ \mathsf{then} \\
\quad\quad \mathsf{return}\ (f(p_1), \ldots, f(p_{t-1}))
\end{array}$$

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 remainder of the paper. To define this core language, we add an explicit failure instruction `fail`, similarly to [8]. It allows us to get rid of conditioning in random samples and observe instructions. Looking ahead, and denoting by $[\![P]\!]^{\mathbb{F}_{q^k}}$ the semantics of the program P inside $\mathbb{F}_{q^k}$, we will have that

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

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 $(e_1, \ldots, e_n)$ defined as follows.

			simplified expressions
e	::=		
	|	P	polynomial
	|	`fail`	failure
	|	$\mathsf{if}\ b\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2$	conditional branching

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 $(0, \ldots, 0)$. We denote by $\overline{\mathcal{P}}_{\mathbb{F}_q}(I, R)$ 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 $(\mathsf{if}\ b\ \mathsf{then}\ x := t_1\ \mathsf{else}\ x := t_2; P)$, i.e. sequential composition after a conditional, implies a propagation of 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 $\mathsf{let}\ x = t\ \mathsf{in}$ 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 $e \in \mathcal{P}_{\mathbb{F}_q}(I,R)$ and $\vec{v} \in \mathbb{F}_{q^k}^{|I \cup R|}$, we define a natural evaluation of e, denoted $[e]_{\vec{v}}^{\mathbb{F}_{q^k}}$, which is a value inside $\mathbb{F}_{q^k}^{|P|} \times \{\texttt{fail}\}$:

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

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

$$[\![P]\!]^{\mathbb{F}_{q^k}} : \mathbb{F}_{q^k}^m \to \mathrm{Distr}(\mathbb{F}_{q^k}^n)$$

For ease of writing, given $\vec{i} \in \mathbb{F}_{q^k}^m$, we will write $[\![P]\!]^{\mathbb{F}_{q^k}}(\vec{i})$ as $[\![P]\!]_{\vec{i}}^{\mathbb{F}_{q^k}}$: thus, given $\vec{o} \in \mathbb{F}_{q^k}^n$ in the output domain, $[\![P]\!]_{\vec{i}}^{\mathbb{F}_{q^k}}(\vec{o})$ denotes the probability that the program returns the output $\vec{o}$ on input $\vec{i}$.

We assume that programs inside $P \in \mathcal{P}_{\mathbb{F}_q}(I,R)$ 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 $\vec{i} \in \mathbb{F}_{q^k}^m$ and output $\vec{o} \in \mathbb{F}_{q^k}^n$ we define

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

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

Example 2.3. Going back to Shamir's secret sharing example of Example 2.2, we can express the property corresponding to it being secure.

The usual way to express the security of this scheme is to say that the return value of `t-n-SSS` along with the secret s follows the same distribution as `t-n-SSS` along with another independent

random value. Essentially, the attacker is unable to distinguish the actual secret from some other independent randomness. We can express this with the following program, parametrized by $b \in \{0,1\}$ and where $f(x) = s_0 + a_1 x + \cdots + a_{t-1}x^{t-1}$, i.e., f is computed using s_0 and is independent of b.

$$
\begin{array}{l}
\texttt{t-n-SSS}_b(p_1, \ldots, p_{t-1}) = \\
\quad s_0, s_1 \xleftarrow{\$} \mathbb{F}^2; \qquad \text{// sample random secrets } s_0 \text{ and } s_1 \\
\quad a_1, \ldots, a_{t-1} \xleftarrow{\$} \mathbb{F}^{t-1}; \qquad \text{// sample } t-1 \text{ random coefficients} \\
\quad \text{if } \bigwedge_{1 \le i \le t-1} p_i \neq 0 \text{ then} \\
\quad\quad \text{return } (s_b, f(p_1), \ldots, f(p_{t-1}))
\end{array}
$$

$\texttt{t-n-SSS}_0$ is the program that leaks $t-1$ shares of the secret along with the real secret s_0, while $\texttt{t-n-SSS}_1$ leaks $t-1$ shares and a fresh independent random value. The security of the scheme requires that the return distribution of the two programs are equivalent.

Formally, the (t,n) secret sharing scheme is secure in the finite field $\mathbb{F}_q$ if, given $p_1, \ldots, p_{t-1} \in \mathbb{F}_q$, we have that:

$$[\![\texttt{t-n-SSS}_0]\!]^{\mathbb{F}_q} = [\![\texttt{t-n-SSS}_1]\!]^{\mathbb{F}_q}$$

This is of course an instance of the previously defined $\mathbb{F}_q$-*equivalence*, where we are asking that the two programs are equivalent, i.e., $\texttt{t-n-SSS}_0 \approx_{\mathbb{F}_q} \texttt{t-n-SSS}_1$.

$\mathbb{F}_{q^\infty}$-equivalence allows us to express the fact that the scheme is secure over all extensions $\mathbb{F}_{q^k}$, requiring that $\texttt{t-n-SSS}_0 \approx_{\mathbb{F}_{q^\infty}} \texttt{t-n-SSS}_1$.

3 COMPLEXITY IN THE FINITE 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 [40]. 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 first study the complexity of deciding whether two probabilistic programs produce exactly the same output distribution. More precisely, we prove that for any $k \in \mathbb{N}$, the $\mathbb{F}_{q^k}$-*equivalence* problem is $\mathsf{coNP}^{\mathsf{C_=P}}$-complete. To this end, we introduce a technical generalization of the equivalence problem, that we call $\mathbb{F}_{q^k}$-*conditional equivalence*, and we proceed in four steps, showing that:

(1) without loss of generality, we can consider programs without inputs; (Lemma 3.2)
(2) verifying if the conditioned distributions of two inputless programs coincide on a fixed point is in $\mathsf{C_=P}$; (Lemma 3.3)
(3) verifying if the conditioned distribution of inputless programs coincide on all points is in $\mathsf{coNP}^{\mathsf{C_=P}}$; (Corollary 3.4)
(4) and finally, even equivalence for programs over $\mathbb{F}_2$ is $\mathsf{coNP}^{\mathsf{C_=P}}$-hard. (Lemma 3.5)

3.1.1 Defining conditional equivalence. The $\mathbb{F}_{q^k}$-*conditional equivalence* problem 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. This intuitively means that we only check if the two programs produce the same output distribution when we only look at the subset of possible executions that satisfy the side-condition. We introduce this generalization for technical reasons, as it simplifies the multiple reductions. Notably, conditional equivalence over programs with

conditionals reduces to conditional equivalence over programs without conditionals, which is not the case for the unconditional equivalence.

Definition 3.1 ($\mathbb{F}_{q^k}$-conditional equivalence). Let $P_1, Q_1 \in \mathcal{P}_{\mathbb{F}_q}(I,R)$ and $P_2, Q_2 \in \overline{\mathcal{P}}_{\mathbb{F}_q}(I,R)$ with $|P_1| = |Q_1| = n$. We denote $P_1 \mid P_2 \approx_{\mathbb{F}_{q^k}} Q_1 \mid Q_2$, if:

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

We forbid P_2 and Q_2 to be branching, as straight-line programs suffice for all our proofs, and generalizing would needlessly complexify the decidability proofs. The universal version *$\mathbb{F}_{q^\infty}$-conditional equivalence* is defined similarly to *$\mathbb{F}_{q^\infty}$-equivalence* , i.e.,

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

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

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: $P_1, Q_1 \in \mathcal{P}_{\mathbb{F}_q}(I,R), P_2, Q_2 \in \overline{\mathcal{P}}_{\mathbb{F}_q}(I,R)$ QUESTION: $P_1 \mid P_2 \approx_{q^k} Q_1 \mid Q_2$?

The decision problem for *q^k-equivalence* simply corresponds to *$\mathbb{F}_{q^k}$-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 *$\mathbb{F}_{q^k}$-equivalence* and *$\mathbb{F}_{q^k}$-conditional equivalence* are both $\mathsf{coNP}^{\mathsf{C_=P}}$-complete.

3.1.2 Complexity results for conditional equivalence. We first introduce some complexity background, before considering the complexity of our problem. Notably, 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 problem defined 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 [39, Sec. 4] and [28]) :

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?

Equipped with those complexity definitions, 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 [5].

Lemma 3.2. *For any $k \in \mathbb{N} \cup \{\infty\}$, $\mathbb{F}_{q^k}$-conditional equivalence reduces to $\mathbb{F}_{q^k}$-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 3.3. *Let $P_1, Q_1 \in \mathcal{P}_{\mathbb{F}_q}(\emptyset, R)$ and $P_2, Q_2 \in \overline{\mathcal{P}}_{\mathbb{F}_q}(\emptyset, R)$ with $|P_1| = |Q_1| = n$. For any $\vec{c} \in \mathbb{F}_{q^k}^n$, we can decide in $\mathsf{C_{=}P}$ if:*

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

Proof. As a shortcut, for $P \in \mathcal{P}_{\mathbb{F}_q}(\emptyset, R)$ (a program without inputs) and $\vec{o} \in \mathbb{F}_{q^k}^{|P|} \times \{\texttt{fail}\}$, we denote by $\tilde{P}^{\vec{o}}$, the probability that P evaluates to $\vec{o}$. Let $P_1, Q_1 \in \mathcal{P}_{\mathbb{F}_q}(\emptyset, R)$, $P_2, Q_2 \in \overline{\mathcal{P}}_{\mathbb{F}_q}(\emptyset, R)$ with $|P_1| = |Q_1| = n$. For any $c \in \mathbb{F}_q^n$, let us consider the non deterministic 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 $\neg(P_1(\vec{r}) = \vec{c} \wedge P_2(\vec{r}) = \vec{0} \wedge Q_1(\vec{r'}) \neq \texttt{fail})$ then
    ACCEPT
  else REJECT
else
  if$(Q_1(\vec{r}) = \vec{c} \wedge Q_2(\vec{r}) = \vec{0} \wedge P_1(\vec{r'}) \neq \texttt{fail})$ then
    ACCEPT
  else REJECT

This machine indeed runs in polynomial time, as evaluating a polynomial over a fixed finite field can be done in polynomial time (note that it is important here that q is not an input of the problem).

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

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

And thus:

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

□

As $\mathsf{C}_{=}\mathsf{P}$ is closed under finite intersection [39], we can decide in $\mathsf{C}_{=}\mathsf{P}$ 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 $\mathsf{C}_{=}\mathsf{P}$ (see Corollary A.2 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. Also, recall that conditional equivalence is a direct generalization of equivalence. We thus trivially have, for any $k \in \mathbb{N} \cup \{\infty\}$, that $\mathbb{F}_{q^k}$*-equivalence* reduces in polynomial time to $\mathbb{F}_{q^k}$*-conditional equivalence*. We thus obtain that:

Corollary 3.4. $\mathbb{F}_{q^k}$-equivalence *and* $\mathbb{F}_{q^k}$-conditional equivalence *are in* $\mathsf{coNP}^{\mathsf{C}_{=}\mathsf{P}}$ *for any* $k \in \mathbb{N}$.

To conclude completeness for both $\mathbb{F}_{q^k}$*-equivalence* and $\mathbb{F}_{q^k}$*-conditional equivalence*, it is sufficient to show the hardness of $\mathbb{F}_2$*-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 A.3).

Lemma 3.5. $\mathbb{F}_2$-equivalence *is* $\mathsf{coNP}^{\mathsf{C}_{=}\mathsf{P}}$*-hard.*

3.2 Independence

We show here that equivalence and (conditional) independence 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. This implies that one cannot infer anything about the distribution of one of the programs given the distribution of the other one.

Definition 3.6 ($\mathbb{F}_{q^k}$-conditional independence*).* Let $P_1, \dots, P_n \in \mathcal{P}_{\mathbb{F}_q}(I, R)$. Given $Y \subset R$, we say that $P_1, \dots, P_n$ are independent conditioned by Y, denoted $\perp^{Y}_{\mathbb{F}_{q^k}} (P_1, \dots, P_n)$, if:

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

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

Example 3.7. 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 $\mathcal{P}_{\mathbb{F}_2}(\{i_1, i_2\}, \{r\})$, we have that $\perp_{\mathbb{F}_2} (i_1(i_2 + r), i_2)$, which means that $i_1(i_2 + r)$ leaks no information about i_2. However, $\not\perp_{\mathbb{F}_2} (i_1(i_2 + r), i_1)$.

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

q^k*-conditional independence*
INPUT: $P_1, \dots, P_n \in \mathcal{P}_{\mathbb{F}_q}(I, R)$, $Y \subset R$
QUESTION: $\perp^{Y}_{q^k} (P_1, \dots, P_n)$?

The universal version, $\mathbb{F}_{q^\infty}$*-conditional independence*, is defined as expected. We now prove that $\mathbb{F}_{q^k}$*-conditional independence* is also $\mathsf{coNP}^{\mathsf{C}_{=}\mathsf{P}}$-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 3.8. *Let $P_1, \dots, P_n$ be programs over $\mathcal{P}_{\mathbb{F}_{q^k}}(I, R)$, and $Y \subset R$.*

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

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

To reduce independence to equivalence, given $P_1, \dots, P_n$, we make a copy P'_i for each program P_i such that all P'_i use disjoint random variables. Therefore all P'_i are independent from one another by construction. We also have that P_i and P'_i have the same distribution. In fact, $(P_1, \dots, P_n)$ has the same distribution as $(P'_1, \dots, P'_n)$ if and only if the P_i are independent.

This translates into the following Lemma.

Lemma 3.9. *Let $P_1, \dots, P_n$ be programs over $\mathcal{P}_{\mathbb{F}_{q^k}}(I, \{r_1, \dots, r_m\})$*

$$\perp_{\mathbb{F}_{q^k}} (P_1, \dots, P_n) \Leftrightarrow (P_1, \dots, P_n) \approx_{\mathbb{F}_{q^k}} (P_1\sigma_1, \dots, P_n\sigma_n)$$

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

Example 3.10. We can illustrate this reduction from independence to equivalence by going back to Shamir's secrect sharing protocol (Examples 2.2 and 2.3). In Example 2.3, security of the protocol was expressed as a program equivalence: the program in which we leak the secret is equivalent to the program leaking a fresh independent random value.

An alternative formalization of the protocol's security is to require that the distribution of the secret is independent from the distribution of the $t-1$ shares:

$$\perp_{\mathbb{F}_{q^k}} (s, \texttt{t-n-SSS})$$

where s is the program that simply returns s and `t-n-SSS` is the program defined in Example 2.2.

Applyin Lemma 3.9, we easily obtain that

$$\perp_{\mathbb{F}_{q^k}} (s, \texttt{t-n-SSS}) \Leftrightarrow \texttt{t-n-SSS}_0 \approx_{\mathbb{F}_{q^k}} \texttt{t-n-SSS}_1.$$

The two previous Lemmas directly yield the following corollary.

Corollary 3.11. $\mathbb{F}_{q^k}$-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 $\perp^{\emptyset}_{\mathbb{F}_2} (P + r, r)$ 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 3.12. $\mathbb{F}_{q^k}$-conditional independence *is* $\mathsf{coNP}^{\mathsf{C_=P}}$*-complete.*

3.3 Majority

The majority problem allows to bound the probability of an event. In this section, our goal is to prove that it is $\mathsf{coNP}^{\mathsf{PP}}$-complete. To this end, we study the complexity of $\mathbb{F}_{q^k}$-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 $\mathbb{F}_{q^k}$-*majority*, which can be decided using $\mathbb{F}_{q^k}$ times $\mathbb{F}_{q^k}$-0-*majority*.

3.3.1 The majority problem. $\mathbb{F}_{q^k}$-*majority* asks if, given two programs, the quotient of their distribution is bounded on all points by some rational r. This can be used either to bound the probability of some event encoded inside a program, or to show that the distributions of two programs are multiplicatively close. $\mathbb{F}_{q^k}$-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 δ-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 $\mathbb{F}_{q^k}$-*majority*, with $k \in \mathbb{N} \cup \{\infty\}$:

q^k-*majority*
INPUT: $P, Q \in \mathcal{P}_{\mathbb{F}_q}(I, R)$, $r \in \mathbb{Q}$ 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. Before tackling the complexity of the majority problem, we once more recall a few complexity definitions. 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 $\mathsf{C}_=\mathsf{P}$. 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 [28]:

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 $\mathbb{F}_{q^k}$-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 $P \prec^{\frac{r}{s}}_{\mathbb{F}_{q^k}} Q \Leftrightarrow (P, T_r) \prec_{\mathbb{F}_{q^k}} (Q, T_s)$, if T_j is a machine which is equal to zero with probability $\frac{1}{j}$.

Lemma 3.13. *For any $k \in \mathbb{N}$, $\mathbb{F}_{q^k}$-0-majority reduces in polynomial time to $\mathbb{F}_{q^k}$-0-majority with $r = 1$.*

The proof showing that $\mathbb{F}_{q^k}$-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 3.14. *For any $k \in \mathbb{N}$, $\mathbb{F}_{q^k}$-0-majority restricted to inputless programs is PP-complete.*

Finally, as PP is closed under finite intersection, we also get that $\mathbb{F}_{q^k}$-*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 $\mathbb{F}_{q^k}$-*majority* to $\mathbb{F}_{q^k}$-0-*majority*. Indeed, for $P, Q \in \mathcal{P}_{\mathbb{F}_q}(I, R)$ and $c \in \mathbb{F}_q^{|P|}$, with a fresh $x \in I$:

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

We show that $\mathbb{F}_{q^k}$-*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 3.15. $\mathbb{F}_{q^k}$-majority *is* $\mathsf{coNP}^{\mathsf{PP}}$ *complete.*

4 THE UNIVERSAL CASE

We first give some general insights on universal equivalence showing important differences with the case of a fixed field. We then derive a general way to study the universal properties by reducing them to Linear Recurrence Sequences problems. This allows us to provide our main decidability result for universal equivalence, 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, that follow easily from the reduction to LRS: independence and 0-majority. For independence and equivalence, the universal problem is in 2-EXP.

4.1 General Remarks

In this Section we try to provide some insights on the difficulty of deciding $\mathbb{F}_q^\infty$-*equivalence*. First of all, we note that equivalence and universal equivalence do *not* coincide.

Example 4.1. The program $x^2 + x$ (with x a random variable) 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 [5]. We denote by $\mathsf{bij}^{\mathbb{F}_q^m}$ the set of permutations 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., [34]), and can be applied as a substitution. On straight-line programs, i.e. polynomials, the characterization can then be stated as follows, where $=_{\mathbb{F}_q}$ denotes equality between polynomials modulo the rule of the field (i.e., $X^q = X$).

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

[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.

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

Example 4.2. Consider $P = xy + yz + zx$ where all variables are randomly sampled. With $\sigma : (x, y, z) \mapsto (x, y + x, z + x)$, we get that $P \approx_{\mathbb{F}_{2^\infty}} x^2 + yz$. Now, $x \mapsto x^2$ is a permutation over all $\mathbb{F}_{2^k}$, so we also have $P \approx_{\mathbb{F}_{2^\infty}} x + yz$ and finally $P \approx_{\mathbb{F}_{2^\infty}} x$.

But here, a permutation 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 permutation 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 $\mathbb{F}_q$*-equivalence* and $\mathbb{F}_{q^\infty}$*-equivalence* are equivalent. Intuitively, the permutation allowing to obtain the equality between two linear programs is also a permutation valid for all extensions of the finite field, as the permutation 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 4.3. $\mathbb{F}_{q^\infty}$-equivalence *restricted to linear programs is in* PTIME.

Moreover, building on results from [29] 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 exceptional polynomials, permutation polynomials over $\mathbb{F}_q[x]$ that are permutations over infinitely many $\mathbb{F}_{q^k}[x]$.

A univariate polynomial that is uniform is then an exceptional polynomial of $\mathbb{F}_q[x]$. They have been fully characterized [32, p237]. The multivariate case appears unsolved, but an efficient algorithm for this case would provide new insights about our problems.

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

$$\sigma \in \bigcap_k \mathrm{bij}^{\mathbb{F}_{q^k}^m} \Rightarrow P \approx_{\mathbb{F}_{q^\infty}} P\sigma$$

Notably, any linear permutation in $\mathrm{bij}^{\mathbb{F}_q^m}$ is also in $\bigcap_k \mathrm{bij}^{\mathbb{F}_{q^k}^m}$. Leveraging some mathematical results classifying the permutations over $\mathbb{F}_{q^k}^m$, we can also provide some insights about functions that are permutations over all extensions of a finite field.

We derive two Lemmas that provide an easy way to generate permutations that are permutations over all extensions of a finite field, and can thus serve as a witness for a universal equivalence.

We first use Theorem 3.2 of [29] to classify what are the permutations over $\mathbb{F}_{q^k}^m$. For a finite field $\mathbb{F}$, $\mathrm{bij}^{\mathbb{F}^n}$ denotes the set of permutations over $\mathbb{F}^n$; and $\mathcal{E}(T(\mathbb{F}, n))$ denotes the set of permutations obtained through

- permutations: $(x_1, \ldots, x_n) \mapsto (x_{\pi(1)}, \ldots, x_{\pi(n)})$),
- scalar multiplications: for any $a \in \mathbb{F}^*$, $(x_1, \ldots, x_n) \mapsto (ax_1, \ldots, x_n)$),
- and the invertible transformations of the form, for any $P \in \mathbb{F}[x_2, \ldots, x_n]$,

$$(x_1, \ldots, x_n) \mapsto (x_1 + P(x_2, \ldots, x_n), \ldots, x_n)$$

$\mathcal{E}(T(\mathbb{F}, n))$ is called the set of the tame automorphisms.

Theorem 4.4 (2.3 of [29]). *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 4.5. *For any prime* $p > 2$, *integers* $k \geq 1$ *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 4.4, we have that for all prime p not equal to 2:

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

Thus, f can be written as a composition of substitutions, scalar multiplications and linear transformations. All those operations are directly permutations 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 4.6. *For any* $k > 1$ *and* $n > 1$, *for any function* f*:*

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

Proof. For any m, we denote by $\mathcal{F}(T(\mathbb{F}_{2^m}, n))$ 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 [26, p. 351] that x^n is a permutation of $\mathbb{F}_q$ if n and $q-1$ are coprime. We have that for any k, 2 and $2^k - 1$ are coprime, and then, we have $\mathcal{F}(T(\mathbb{F}_{2^{2(2k+1)}}, n)) = \mathsf{bij}^{\mathbb{F}_{2^{2(2k+1)}}}$

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

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

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

□

4.2 From Arithmetic Programs without Inputs to LRS

In this Section, we consider the case of arithmetic programs without inputs, $P, Q \in \overline{\mathcal{P}}_{\mathbb{F}_q}(\emptyset, R)$. In this sub-case, P and Q are simply tuples of polynomials over a finite field. For such programs, we show how we can use results about Linear Recurrence Sequences (LRS) to reason about our problems. To this end, we leverage the local zeta Riemann functions. We thus first introduce the necessary background about the local zeta functions and LRS, before discussing how it applies to our programs.

Local zeta Riemann functions. We 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

$$Z(P, T) = \exp\left(\sum_{k \geq 1} \frac{|N_k(P)|}{k} T^k\right)$$

where $N_k(P) = \{\vec{x} \in \mathbb{F}^n_{q^k} \mid \bigwedge_{1 \leq i \leq m} P_i(\vec{x}) = 0\}$.

Remark that given $P \in \overline{\mathcal{P}}_{\mathbb{F}_q}(\emptyset, R)$,

$$[\![P]\!]^{\mathbb{F}_{q^k}}(0) = \frac{|N_k(P)|}{q^{k\times|R|}}$$

The remarkable fact is that for a given P, $Z(P,T)$ is a rational function in T whose polynomial representation can be computed. Equivalently, this allows us to test the equality of local-zeta functions. We briefly survey the results that lead to the proof of the previous fact.

Weil's conjecture [42] states several fundamental properties of local zeta Riemann functions over algebraic varieties. Dwork [16] 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 [9] 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 defining Z; since these values can be computed by brute force, this yields an algorithm for computing the polynomials R and S by interpolation (remark that we are only discussing decidability here, not efficient computation).

We remark that Weil's conjecture actually only applies to non-singular projective varieties, which would be an issue for our applications. However, as outlined by [24], Dwork's proof can be used to obtain the stronger result of the rationality of the local zeta function for any algebraic variety. Furthermore, [24] presents one of the algorithm with the best known complexity, and we will use this algorithm's complexity to upper bound the complexity of our problems.

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 $\mathbb{F}_{q^k}$.

Linear recurrence sequences. We recall that the Linear Recurrence Sequence denoted by $\langle u_k \rangle$ is an infinite sequence of reals $u_1, u_2, \ldots$ such that there exist real constants $a_1, \ldots, a_n$ such that for all $k \geq 0$,

$$u_{k+n} = a_1 u_{k+n-1} + \cdots + a_n u_k$$

For an extensive background, we refer the reader to [17], we recall here there main features. The order of a LRS $\langle u_k \rangle$ is the smallest positive n such that the equation above holds. The recurrence relation can be associated to a polynomial, called the characteristic polynomial. We then say that a LRS is simple if its characteristic polynomial does not have any repeated roots. As a sufficient condition (see e.g. [37]), a LRS of order n is simple if there exist algebraic constants $\gamma_1, \ldots, \gamma_n$ and non-zero algebraic constants $c_1, \ldots ..., c_n$ such that, for all $k \geq 0$:

$$u_k = \sum_{1 \leq i \leq n} c_i \gamma_i^k$$

Remark that given two simple LRS of order n, it is enough to test the equality of the first $n+1$ terms to obtain equality of the two LRS. Some other problems related to our study are:

- the positivity problem: for all $k \in \mathbb{N}$, does it holds that $u_k \geq 0$? It is only known to be decidable for LRS of order smaller than 5, and smaller than 9 in the case of simple LRS.
- the ultimate positivity problem: does there exists K such that for all $k > K$, $u_k \geq 0$? It is decidable for simple LRS but its decidability in the general case is open.

From programs to LRS. Summing up the results from Dwork, Bombieri and Deligne, [12] allows us to characterize $\langle N_k(P) \rangle$ as a simple LRS. Concretely, this means that we can find the equation that allows to compute the probability that P returns 0 given the size of the finite field. Indeed,

given a tuple P of m polynomials in n variables with maximal degree a, there exist integers a_1, a_2 such that $a_1 + a_2 \leq (4a+9)^{n+m}$ and algebraic complex numbers $\alpha_1, \ldots, \alpha_{a_1}, \beta_1, \ldots, \beta_{a_2}$ such that for any $k \geq 1$:

$$N_k(P) = \sum_{j=1}^{a_2} \beta_j^k - \sum_{j=1}^{a_1} \alpha_j^k$$

We thus have that $\langle N_k(P) \rangle$ is a simple LRS. Remark that given P, computing the LRS corresponding to $N_k(P)$ or computing $Z(P)$ is equivalent (recall that the derivative of the logarithm of $Z(P)$ is the formal power series corresponding to the LRS $\langle N_k(P) \rangle$, which is sometimes called the Poincaré serie), and the reductions given in this paper from problems on programs to problems on LRS are thus exponential (the cost of computing the local-zeta function). Based on the previous discussions, we obtain the following Corollary:

Corollary 4.7. *Let $P_1, \ldots, P_k \in \overline{\mathcal{P}}_{\mathbb{F}_q}(\emptyset, R)$, any linear combination of the $\{N_k(P_i)\}_{1 \leq i \leq k}$ is a LRS. So is any linear combination of the $\{[\![P_i]\!]^{\mathbb{F}_{q^k}}(0)\}_{1 \leq i \leq k}$.*

LRS, which have been widely studied, provide a uniform way to reason about our relational properties:

- first encode the relational property as a property of some linear combinations of $\{[\![P_i]\!]^{\mathbb{F}_{q^k}}(0)\}_{1 \leq i \leq k}$;
- then reason about the corresponding properties of the simple LRS.

This directly implies that, given $P, Q \in \overline{\mathcal{P}}_{\mathbb{F}_q}(\emptyset, R)$, one can decide if:

- $\exists K,\ \forall k > K.\ [\![P]\!]^{\mathbb{F}_{q^k}}(0) \geq 0$. This is because ultimate positivity is decidable for simple LRS [37]. This implies decidability of a variant of the q^k-0-majority, that we may call ultimate q^k-0-majority: $\exists K,\ \forall k > K.\ [\![P]\!]^{\mathbb{F}_{q^k}}(0) \geq [\![Q]\!]^{\mathbb{F}_{q^k}}(0)$.
- $\forall k \geq 0.\ [\![P]\!]^{\mathbb{F}_{q^k}}(0) = [\![Q]\!]^{\mathbb{F}_{q^k}}(0)$. This is because we can decide if two LRS are equal. Remark that this is only a reformulation of testing if $Z(P) = Z(Q)$. Hence, testing if two programs have the same probability to return 0 over all finite fields is decidable.

Furthermore, for arithmetic programs without inputs, we have reduced $\mathbb{F}_{q^\infty}$-0-*majority* to the positivity problem for LRS, as the question is if for all k, $[\![P]\!]^{\mathbb{F}_{q^k}}(0) - [\![Q]\!]^{\mathbb{F}_{q^k}}(0) \geq 0$ where the left hand side is a LRS.

4.3 Decidability of Universal Equivalence

We show decidability of $\mathbb{F}_{q^\infty}$-*equivalence*, leveraging tools from algebraic geometry, showing that[4]:

(1) $\mathbb{F}_{q^\infty}$-*conditional equivalence* is decidable for arithmetic programs; (Lemma 4.9)
(2) it is also decidable for programs with conditionals; (Lemma 4.11)
(3) it is finally decidable for programs with conditioning, e.g., failures. (Lemma 4.12)

Direct consequences of the local zeta function. Notice that, given two programs P and Q, the local zeta function directly allows us to conclude if they are equal to some value with the same probability for all extensions of the base field. Moreover, thanks to Theorem 3 of [23], the local zeta functions are also defined and rational (and thus computable once again through Bombieri's bound) for sets defined by arbitrary first order formulas. That is, rather than $N_k(P)$ being defined as the number of times such that $P = 0$ when evaluting inside $\mathbb{F}_{q^k}$, we can define $N_k(P)$ as the number of times that a first-order formula $\phi(P)$ over finite fields holds when evaluated inside $\mathbb{F}_{q^k}$. Transposing this to our programs, we get the following corollary.

[4]The following reductions do not hold for equivalence, it is the reason why we considered conditional equivalence. It works as equivalence trivially reduces to conditional equivalence.

Corollary 4.8. *Let ϕ and ψ be two first order formulas 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 that can be expressed as first order formulas over a 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 by a first order formula.

General results. We first show that $\mathbb{F}_{q^\infty}$*-equivalence* is decidable for arithmetic programs, i.e. programs without conditionals or conditioning. Recall that using the local zeta function, testing whether $Z(P) = Z(Q)$, directly allows us to conclude that P and Q have the same probability of returning $\vec{0}$ over all extensions of the field. This directly generalizes to an arbitrary output value $\vec{c}$ by testing if $Z(P + \vec{c}) = Z(Q + \vec{c})$. But for equivalence, we must be able to check this for all possible return value, i.e., check if $\forall \vec{c}.\ Z(P + \vec{c}) = Z(Q + \vec{c})$. Therefore, we express the output distributions as vectors, and show that the two-norm of the distances between the distributions is a LRS, that we can then compute.

Given an enumeration $1 \leq j \leq q^{kn}$ of the elements c_j of $\mathbb{F}_{q^k}^n$, for any programs $P_1, P_2 \in \overline{\mathcal{P}}_{\mathbb{F}_q}(\emptyset, R)$ where $|P_1| = n$, we denote by $\overrightarrow{P_1, P_2}^k = ([[P_1, P_2]]^{\mathbb{F}_{q^k}}(\vec{c_1}, \vec{0}), \ldots, [[P_1, P_2]]^{\mathbb{F}_{q^k}}(\vec{c_{q^{kn}}}, \vec{0}))$, that completely characterizes the distribution of P_1 conditioned by $P_2 = 0$. Notice that when $|R| = m$, we have:

$$q^{km}\overrightarrow{P_1, P_2}^k = (N_k(P_1 + \vec{c_1}, P_2), \ldots, N_k(P_1 + \vec{c_{q^{kn}}}, P_2))$$

The core of the reduction to LRS is that the squared norm-two of $\overrightarrow{P_1, P_2}^k - \overrightarrow{Q_1, Q_2}^k$ is a LRS. Indeed, recall that:

$$\|\overrightarrow{P_1, P_2}^k - \overrightarrow{Q_1, Q_2}^k\|_2^2 = \sum_{c \in \mathbb{F}_{q^k}^n} (N_k(P_1 + \vec{c}, P_2) - N_k(Q_2 + \vec{c}, Q_2))^2$$

So we have that

$$P_1 \mid P_2 \approx_{\mathbb{F}_{q^\infty}} Q_1 \mid Q_2 \iff \forall k \in \mathbb{N}.\ \|\overrightarrow{P_1, P_2}^k - \overrightarrow{Q_1, Q_2}^k\|_2^2 = 0$$

This allows us to directly conclude decidability, as we can decide if the corresponding LRS is always zero. Remark that it is crucial to reduce to linear sequences compared to arbitrary ones. Otherwise we would be unable to find a bound on the number of elements of the sequence required to define the sequence, and we would not know how to decide equality to zero.

We now provide the core Lemma, where we provide a way to express $\|\overrightarrow{P_1, P_2}^k - \overrightarrow{Q_1, Q_2}^k\|_2^2$ as a combination of local-zeta functions over well chosen polynomials, and thus see $\|\overrightarrow{P_1, P_2}^k - \overrightarrow{Q_1, Q_2}^k\|_2^2$ as a LRS.

Lemma 4.9. *Let P_1, P_2, Q_1, Q_2 in $\overline{\mathcal{P}}_{\mathbb{F}_{q^k}}(\emptyset, R)$. We have that $\|\overrightarrow{P_1, P_2}^k - \overrightarrow{Q_1, Q_2}^k\|_2^2$ is a LRS.*

Proof. Using the classical inner product $\vec{x} \cdot \vec{y} = \sum_i x_i y_i$, for any k and programs $U, V, U', H' \in \overline{\mathcal{P}}_{\mathbb{F}_{q^k}}(\emptyset, R)$, we have, when σ is a mapping from variables in R to fresh variables in R' and $|R| = m$:

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

So now,

$$\begin{aligned} &N_k(P_1 - P_1\sigma, P_2, P_2\sigma) - 2N_k(P_1 - Q_1\sigma, P_2, Q_2\sigma) + N_k(Q_1 - Q_1\sigma, Q_1, Q_1\sigma) \\ &\quad = q^{2km} \times (\overrightarrow{P_1, P_2}^k \cdot \overrightarrow{P_1, P_2}^k - 2\overrightarrow{P_1, P_2}^k \cdot \overrightarrow{Q_1, Q_2}^k + \overrightarrow{Q_1, Q_2}^k \cdot \overrightarrow{Q_1, Q_2}^k) \end{aligned}$$

In other terms:

$$\begin{aligned} &N_k(P_1 - P_1\sigma, P_2, P_2\sigma) - 2N_k(P_1 - Q_1\sigma, P_2, Q_2\sigma) + N_k(Q_1 - Q_1\sigma, Q_1, Q_1\sigma) \\ &\quad = q^{2km} \times \|\overrightarrow{P_1, P_2}^k - \overrightarrow{Q_1, Q_2}^k\|_2^2 \end{aligned}$$

Corollary 4.7 finally allows us to conclude. □

We can now conclude decidability of $\mathbb{F}_{q^\infty}$-*equivalence* for arithmetic programs, as we can decide if the corresponding LRS is always zero. Computing the LRS is in fact equivalent to computing the associated local zeta functions, and thus check if the following is equal to 0:

$$Z(P_1 - P_1\sigma, P_2, P_2\sigma) - 2Z(P_1 - Q_1\sigma, P_2, Q_2\sigma) + Z(Q_1 - Q_1\sigma, Q_1, Q_1\sigma)$$

Using the complexity for the computation of the local zeta function provided by [24, Corollary 2] we obtain the following corollary.

Corollary 4.10. $\mathbb{F}_{q^\infty}$-conditional equivalence *and* $\mathbb{F}_{q^\infty}$-equivalence *restricted to arithmetic programs are in* EXP.

Removing the conditionals. We now wish to remove conditionals, in order to reduce equivalence for programs with conditionals to arithmetic programs (which are simply tuples of polynomials). To remove the conditionals, the first idea is to use a classical encoding in finite fields:

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

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, for any program Q and output $\vec{o}$:

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

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

Lemma 4.11. *For any* $k \in \mathbb{N} \cup \{\infty\}$, $\mathbb{F}_{q^k}$-conditional equivalence *restricted to programs without failures reduces in exponential time to* $\mathbb{F}_{q^k}$-conditional equivalence *restricted to arithmetic programs.*

Removing failures. Recall that failures define the probabilistic semantics through normalization. For instance, for a program (if $b = 0$ then P_1 else $\texttt{fail}, P_2$) where P_1 and P_2 do not fail and b is a polynomial, for any $\vec{o}$, we have:

$$[\![(\text{if } b = 0 \text{ then } P_1 \text{ else } \texttt{fail}, P_2)]\!]^{\mathbb{F}_{q^k}}(\vec{o}, \vec{0}) = \frac{\mathbb{P}\{P_1 = \vec{o} \wedge P_2 = \vec{0} \wedge b = 0\}}{\mathbb{P}\{b = 0\}}$$

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 in conditional equivalence some probabilities, as $[\![P]\!]^{\mathbb{F}_{q^k}}(\vec{o}) \times \mathbb{P}\{b = 0\} = [\![P, b]\!]^{\mathbb{F}_{q^k}}(\vec{o}, 0)$ 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:

$$\begin{aligned}\text{if } b = 0 \text{ then } P_1 \text{ else } \texttt{fail} \mid P_2 \approx_{\mathbb{F}_{q^k}} Q \mid 0 &\Leftrightarrow \forall \vec{o}. [\![(\text{if } b \text{ then } P_1 \text{ else } \texttt{fail}, P_2)]\!]^{\mathbb{F}_{q^k}}(\vec{o}, \vec{0}) = [\![Q]\!]^{\mathbb{F}_{q^k}}(\vec{o}) \\ &\Leftrightarrow \forall \vec{o}. \mathbb{P}\{P_1 = \vec{o} \wedge P_2 = \vec{0} \wedge b = 0\} = \mathbb{P}\{b = 0\}[\![Q]\!]^{\mathbb{F}_{q^k}}(\vec{o}, \vec{0}) \\ &\Leftrightarrow \forall \vec{o}. [\![P_1, P_2, b]\!]^{\mathbb{F}_{q^k}}(\vec{o}, \vec{0}) = \mathbb{P}\{b = 0\}[\![Q]\!]^{\mathbb{F}_{q^k}}(\vec{o})\end{aligned}$$

Universal equivalence. Using those techniques, we obtain:

Lemma 4.12. *For any $k \in \mathbb{N} \cup \{\infty\}$, $\mathbb{F}_{q^k}$-*conditional equivalence *reduces to $\mathbb{F}_{q^k}$-*conditional equivalence *restricted to programs without failures in exponential time.*

The previous Lemmas allows us to conclude.

Theorem 4.13. *$\mathbb{F}_{q^\infty}$-*equivalence *and $\mathbb{F}_{q^\infty}$-*conditional equivalence *are in* 2-EXP.

We remark here that the improvement in the complexity of computing the local zeta function w.r.t. the number of polynomials, proposed in [13], improves this bound and allows to obtain an EXP bound.

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

Lemma 4.14. *$\mathbb{F}_q$-*equivalence *reduces in polynomial time to $\mathbb{F}_{q^\infty}$-*equivalence. *$\mathbb{F}_{q^\infty}$-*equivalence *is thus* $\text{coNP}^{\text{C}_=\text{P}}$*-hard.*

Remark that improving this lower bound is unlikely to allow to obtain new lower-bounds for LRS decision problems. Indeed, as we go through an exponential time reduction to be able to get the LRS corresponding to a given program, we cannot transfer lower-bounds under exponential time to the LRS side.

Independence. Using once again Lemmas 3.8 and 3.9, we obtain the same complexity results for the independence problem.

Corollary 4.15. *$\mathbb{F}_{q^\infty}$-*conditional independence *is in* 2-EXP.

Universal zero-majority without inputs. For arithmetic programs, we have reduced *$\mathbb{F}_{q^\infty}$-0-majority* to the positivity problem for LRS. The generalization to general programs with conditionings and branchings is similar to the reductions for universal equivalence. We thus obtain the following result.

Theorem 4.16. *$\mathbb{F}_{q^\infty}$-*0-majority *for inputless programs reduces in exponential time to the positivity problem for simple LRS.*

The reduction can also be applied with the generalization of [23], 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.

We also remark that similarly to $\mathbb{F}_{q^\infty}$-*equivalence*, the complexity of the problem strongly comes from the presence of multiplications. Indeed, in the linear case, majority implies equivalence and we obtain the following.

Lemma 4.17. $\mathbb{F}_{q^\infty}$-*0*-majority *restricted to linear programs is in* PTIME.

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

Lemma 4.18. $\mathbb{F}_{2^\infty}$-0-majority *is* PP-*hard.*

Compared to equivalence, we do not have a way to reduce majority or 0-majority programs without inputs. Thus, we are not able to generalize the reduction to the positivity problem for those cases.

4.4 Bounded Universal Simulatability

Simulation-based proofs [27] are a 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 between 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 properly define this notion inside our framework. We were however unable to provide a general decidabiltiy result. We thus consider a simpler task, where the size of the simulator is bounded. In this case, we are able to derive decidability from our previous result.

Given a program C, we denote by $\deg(C)$ the maximum degree of a program, 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). Finally, given programs P, Q such that Q contains an input variable i, we denote by $Q[{}^{P}/_{i}]$ the program Q where i has been replaced by the return value of P.

Definition 4.19. [Bounded (universal) simulatability] Let $P, Q \in \mathcal{P}_{\mathbb{F}_q}(I, R)$, R' such that $|R| = |R|'$ and $l \in \mathbb{N}$. We write $P \sqsubseteq^{l}_{\mathbb{F}_{q^k}} Q$, if there exists $S \in \mathcal{P}_{\mathbb{F}_q}(\{i_1, \ldots, i_n\}, R'\})$ such that $\deg(S) \leq l$, and

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

The associated decision problem is:

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

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 A.6. From the bound on the number of contexts and the decidability of universal equivalence, one can derive the decidability of bounded simulatability.

Corollary 4.20. l,$\mathbb{F}_{q^k}$-simulatability *is decidable.*

As a lower bound, we prove that $l, \mathbb{F}_{q^k}$-*simulatability* is as hard as universal equivalence:

Lemma 4.21. *For any* $l \in \mathbb{N}$, $k \in \mathbb{N} \cup \{\infty\}$, $\mathbb{F}_{q^k}$-equivalence *reduces in polynomial time to* $l, \mathbb{F}_{q^k}$-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.

5 PROGRAM INDISTINGUISHABILITY

Computational indistinguishability is a classic notion in cryptography used to model the security properties. It allows one to specify that two programs parameterized by a security parameter, in our case the size of the field, should behave very similarly when the parameter grows, and thus have indistinguishable behaviours. To reason about this notion in our setting, we study and define program indistinguishability and the LRS negligibility problem. We make some first steps towards the decidability of program indistinguishability by proving the decidability of the LRS negligibility problem, but leave the decidability of program indistinguishability open, except for binary programs, that only return a boolean. In the special case of binary programs, we show the decidability of program indistinguishability, using the same reduction to LRS as for the case of universal equivalence.

A variant of equivalence that is of interest for security proofs is indistinguishability. Intuitively, it means that the statistical distance between two programs is negligible w.r.t., some security parameter.

Definition 5.1. We say that two programs P, Q are indistinguishable, denoted by $P \sim Q$, if for all $d \in \mathbb{N}$ there exists K_d such that:

$$\forall k > K_d. \sum_{\vec{o} \in \mathbb{F}_{q^k}^n} |[\![P]\!]^{\mathbb{F}_{q^k}}(\vec{o}) - [\![Q]\!]^{\mathbb{F}_{q^k}}(\vec{o})| \leq \frac{1}{k^d}$$

Or equivalently:

$$P \sim Q \Leftrightarrow \forall d \in \mathbb{N}, \exists K_d, \forall k > K_d. \, \|\vec{P}^k - \vec{Q}^k\|_1 \leq \frac{1}{k^d}$$

Indistinguishability of programs is of course implied by equivalence, but the converse is not true. Consider for instance the program that always outputs 0, $P :=$ return 0, and the program $Q := x \xleftarrow{\$} \mathbb{F}$, if $x = 0$ then return 1 else 0.

This is very close to the definition of computational indistinguishability. A widely known fact (see e.g. [20]) is that the negligibility of the statistical distance implies the computational indistinguishability of the two programs: no attacker can guess with which of two programs they interact. In other terms, abusively denoting programs as protocols outputting some value, we have $P \sim Q \Rightarrow P \perp Q$.

We provide some first insight about the program-indistinguishability problem by showing that the corresponding LRS problem is decidable (this relies heavily on the techniques of [37] and on some of its notations, that we do not recall here).

Classically, a positive function $f : k \mapsto f(k)$ is negligible if:

$$\forall d, \exists K_d, \forall k > K_d. \, f(k) \leq \frac{1}{k^d}$$

Notably, for any $x < 1$, $k \mapsto x^k$ is negligible.

Definition 5.2. A simple LRS $\langle u_k \rangle$ is negligible if:

$$\forall d, \exists K_d, \forall k > K_d. \, |u_k| \leq \frac{1}{k^d}$$

Before giving the Theorem about the negligibility of a LRS, we recall some of their notions. A simple LRS $\langle u_k \rangle$ can be expressed under the form

$$u_k = \sum_{1 \leq i \leq n} c_i \gamma_i^k$$

where $\gamma_1, \ldots, \gamma_n, c_1, \ldots ..., c_n$ are non-zero algebraic complex constants. The γ_is are called the roots of the LRS, as they correspond to the roots of its characteristic polynomial (cf. Section 4.2). We now derive a sufficient and necessary condition over the maximal modulus of the roots of a LRS for it to be negligible.

We first perform the proof for a non-degenerate LRS, where a LRS is said to be non-degenerate if it does not have two distinct characteristic roots whose quotient is a root of unity.

Theorem 5.3. *Let $\langle u_k \rangle$ be a non-degenerate simple and rational LRS with roots $\gamma_1, \ldots, \gamma_n$ and $M = \mathtt{max}_{1 \leq i \leq n}\{|\gamma_i|\}$ be the maximal modulus of the roots. $\langle u_k \rangle$ is negligible if and only if $M < 1$.*

Proof. We perform a case analysis on the maximal modulus M of the roots.

Maximal modulus $M < 1$. Let $C = \mathtt{max}_i(\{c_i\})$. We have by the triangle inequality that $\forall k.\ |u_k| \leq nCM^k$. If M is smaller than one, $k \mapsto nCM^k$ is a negligible function, and thus the LRS is negligible.

Maximal modulus $M > 1$. This follows directly from [1] that states that any non-degenerate LRS with a root of maximal modulus greater than 1 diverges. It is the non negligible.

Maximal modulus $M = 1$. Without loss of generality, we assume that the LRS is not always zero (else it is trivially negligible). We will use Braverman's Lemma [10], which we recall here (it is also called the Complex Units Lemma in [1]).

Lemma 5.4 (Braverman's Lemma [10]). *Let $z_1, \ldots, z_m \in \{z \in \mathbb{C} \mid |z| = 1\} \setminus 1$ be distinct complex numbers and let $\alpha_1, \ldots, \alpha_m \in \mathbb{C} \setminus \{0\}$. Set $u_n := \sum_{k=1}^{m} \alpha_k z_k^n$. Then there exists $c < 0$ such that for infinitely many n, $Re(u_n) < c$.*

Here, $Re(x)$ denotes the real part of the complex number. Remark that by applying the lemma to both u_n and $-u_n$, we can actually get a constant $c > 0$ such that for infinitely many n, $Re(u_n) < -c$ and $Re(u_n) > c$.

To apply this lemma to the general form of $\langle u_k \rangle$, we must distinguish whether 1 is a root or not of the LRS. We denote by $\Lambda = \{\gamma_i\}$ the set of roots of $\langle u_k \rangle$ and we consider $\Lambda_{max} = \{x \in \Lambda \mid |x| = 1\}$ the set of roots of modulus 1.

Assume that $1 \notin \Lambda_{max}$. We can write u_k as $\sum_{\gamma_j \in \Lambda_{max}} c_j \gamma_j^k + r_k$, where $r_k = o(1)$ (r_k converges to zero) and for all j, $\gamma_j \neq 1$. By applying Braverman's Lemma to $\sum_{\gamma_j \in \Lambda_{max}} c_j \gamma_j^k$, we get $c > 0$ such that infinitely often, $Re(\sum_{\gamma_j \in \Lambda_{max}} c_j \gamma_j^k) > c$ (and also infinitely often $Re(\sum_{\gamma_j \in \Lambda_{max}} c_j \gamma_j^k) < -c$. We then have that $|\sum_{\gamma_j \in \Lambda_{max}} c_j \gamma_j^k|$ goes above c infinitely often. And thus, as $r_k = o(1)$, there exists some $\epsilon < c$ such that infinitely often we have $|u_k| \geq c - \epsilon > 0$. The LRS is then non-negligible.

Assume that $1 \in \Lambda_{max}$. Then, there exists an index j such that γ_j equals 1. If we consider $v_k = u_k - c_j$, we know that all the roots of v_k are distinct from 1 and have a modulus smaller or equal to 1. We perform a sub-case analysis, depending on whether v_k contains other roots of modulus 1.

- If v_k does not have any other roots with of modulus 1, all its other roots are of modulus strictly smaller than 1. Then, $|v_k|$ converges to 0, and $|u_k|$ converges to $|c_j|$ and is thus non-negligible.
- Otherwise, we do the exact same step over v_k as we did over u_k in the case $1 \notin \Lambda_{max}$, by applying Braverman's Lemma to the part of v_k containing the roots of modulus 1. We then have $c > 0$ such that infinitely often, $Re(u_k - c_j) > c$ and $Re(u_k - c_j) < -c$. Assume by contradiction that $|u_k|$ is negligible. It would then converge to zero, and we would need to have both $Re(-c_j) > c$ and $Re(-c_j) < -c$, which is a contradiction. This concludes the proof.

□

We get as a corollary that it also decidable in the degenerate case.

Corollary 5.5. *Let $\langle u_k \rangle$ be a simple rational LRS. It is possible to decide if $\langle u_k \rangle$ is negligible.*

We use [37, Proposition 2.2] that states that:

Lemma 5.6 (Proposition 2.2 [37]). *Let $\langle u_k \rangle$ be an LRS over $\mathbb{Q}$. There is a constant s such that each subsequence $\langle u_{sk+l} \rangle$ is non-degenerate for $0 \leq l < s$*

Now, remark that given such a s, $\langle u_k \rangle$ is negligible if and only if all the subsequences $\langle u_{sk+l} \rangle$ are negligible for $0 \leq l < s$. Indeed, if any of the subsequence is non-negligible, the LRS is also non-negligible and if all the subsequences are negligible, so is the LRS.

Thus, we can decide if $\langle u_k \rangle$ is negligible as we can decide if all its non-degenerate subsequences are negligible with Theorem 5.3.

Now, we can show that the negligibility of $\|\vec{P}^k - \vec{Q}^k\|_2^2$ is decidable, as thanks to Lemma 4.9 it is a simple LRS. This provides a necessary condition and a weak sufficient condition for $P \sim Q$, as for any k, when m is the output arity of P and Q, we have:

$$\|\vec{P}^k - \vec{Q}^k\|_2 \leq \|\vec{P}^k - \vec{Q}^k\|_1 \leq q^{km/2} \|\vec{P}^k - \vec{Q}^k\|_2$$

(this inequality is the one corresponding to the equivalence of the norm one and norm two in Euclidian spaces, for vectors of length q^{km})

Now, if we consider programs that always return either 0 or 1, we have that $\|\vec{P}^k - \vec{Q}^k\|_1 = \|\vec{P}^k - \vec{Q}^k\|_2^2$. This observation allows to obtain the following corollary, combining Lemma 4.9 and the fact that the negligibility of a simple LRS is decidable.

Corollary 5.7. *Program indistinguishability restricted to programs that always output either 0 or 1 is decidable.*

Open question. Unfortunately, we leave the decidability in the general case as an open question. We recall that

$$\|\vec{P}^k - \vec{Q}^k\|_1 = \sum_{\vec{c} \in \mathbb{F}_{q^k}^m} |N_k(P + \vec{c}) - N_k(Q + \vec{c})|$$

Expressing this quantity as a LRS, and for instance as function of some local zeta function would then solve the question. The reduction would also work if any polynomial over $\|\vec{P}^k - \vec{Q}^k\|_1$ can be seen as a LRS, as any function is negligible if and only if any polynomial in this function is negligible.

6 UNDECIDABILITY WITH LOOPS

We have proved the decidability of universal equivalence for programs without loops. We now prove that this result cannot be generalised, by proving the undecidability of universal equivalence for programs with loops over finite fields. This is done by reduction from the halting problems of Minsky machines with two counters.

Assuming the same guards b as in the conditionals (Figure 3), we add the while b do c construct to our language. The associated semantics is natural and not detailed (note that we also extend the semantics of variables to be used in loops). The semantics of a program that does not terminate is given a specific value $\perp^*$. Then, the uniform equivalence problem of this enriched language is undecidable. We reduce the halting problem for two counter Minsky Machines.

A Minsky Machine, or counter machine, is a 3 tuple (C, L, I) where

- $C = \{c_1, \ldots, c_l\}$ is a set of counters;
- $L = \{l_1, \ldots, l_m\}$ is an ordered set of labels;
- and $I = \{i_1, \ldots, i_m\}$ is an ordered set of instructions.

For each instruction i_j, l_j is the associated label, used for jumps. Instructions are of the form:

$$\begin{aligned} i := \;& \mathsf{incr}(c_k); \mathsf{JUMP}(l_j) \\ & \mid \mathsf{decr}(c_k); \mathsf{JUMP}(l_j) \\ & \mid \mathsf{if}\ c_k = 0\ \mathsf{then}\ \mathsf{JUMP}(l_s)\ \mathsf{else}\ \mathsf{JUMP}(l_t) \\ & \mid \mathsf{HALT} \end{aligned}$$

A configuration of the machine is given as a couple $(n_1, \ldots, n_l), i$ where $(n_1, \ldots, n_l) \in \mathbb{N}^l$ and $i \in I$. Intuitively, the configuration gives explicitly a value for all the counters of the machine, and stores in a dedicated register the current instruction to be executed. The one step reduction of a machine M is denoted by $\rightarrow_M$, defined by:

$$\begin{aligned} & (n_1, \ldots, n_l), (\mathsf{incr}(c_k); \mathsf{JUMP}(l_j)) \rightarrow_M (n_1, \ldots, n_k + 1, \ldots, n_l), i_j \\ & (n_1, \ldots, n_l), (\mathsf{decr}(c_k); \mathsf{JUMP}(l_j)) \rightarrow_M (n_1, \ldots, n_k - 1, \ldots, n_l), i_j \ (\text{when } n_k > 0) \\ & (n_1, \ldots, n_l), (\mathsf{if}\ c_k = 0\ \mathsf{then}\ \mathsf{JUMP}(l_s)\ \mathsf{else}\ \mathsf{JUMP}(l_t)) \rightarrow_M (n_1, \ldots, n_l), i_s \ (\text{when } n_k = 0) \\ & (n_1, \ldots, n_l), (\mathsf{if}\ c_k = 0\ \mathsf{then}\ \mathsf{JUMP}(l_s)\ \mathsf{else}\ \mathsf{JUMP}(l_t)) \rightarrow_M (n_1, \ldots, n_l), i_t \ (\text{when } n_k \neq 0) \end{aligned}$$

We denote by $\rightarrow_M^*$ its transitive closure.

The halting problem for two counter Minsky machines, i.e., a machine (C, L, I) with $|C| = 2$, is undecidable: given a machine M and an initial configuration C, r, one cannot decide if there exists a value C' of the counters such that $C, r \rightarrow_M^* C', \mathsf{HALT}$.

Theorem 6.1. $\mathbb{F}_{q^\infty}$-equivalence *is undecidable for programs with loops.*

Proof. Let $M = (C, r, L, I)$ be a two counter machine where $C = (\{c_1, c_2\}, L = \{l_1, \ldots, l_m\}$ and $I = \{i_1, \ldots, i_m\}$, and an initial configuration $(n_1, n_2), i_s$.

We build a program over $\mathbb{F}_q$ which emulates the counter machine execution, and which will be such that it never terminates in all interpretations if and only if M does not terminate. Then, the program will be universally equivalent to a program which never halts if and only if M does not terminate, which is the expected reduction.

We choose q to be the smallest prime number bigger than m, and we assume, without loss of generality, that the only halt instruction of M is l_1. We can then emulate r with a single variable, where the macro $\mathsf{JUMP}(l_i)$ is simply $r := i$.

If we denote by $[i]$ the encoding of an instruction i defined later, the core of the program is then:

$$\begin{aligned} & r := s \\ & \mathsf{while}\ r \neq 1\ \mathsf{do} \\ & \quad \mathsf{if}\ r = 2\ \mathsf{then} \\ & \quad\quad [i_2] \\ & \quad \ldots \\ & \quad \mathsf{if}\ r = m\ \mathsf{then} \\ & \quad\quad [i_m] \end{aligned}$$

We now provide encodings for each instruction. We first define a dummy non halting program $Loop := \mathsf{while}\ 0 = 0\ \mathsf{do}\ t \xleftarrow{\$} \mathbb{F}$ (the sampling of t is an alias for no operation).

To model the counters, we sample a pair of variables $x_1, x_2 \xleftarrow{\$} \{\mathbb{F} \mid x_1 x_2 = 1\}$, and a counter of value n is represented by x_1^n. In this representation incrementing the counter corresponds to multiplication by x_1, and decrementing is achieved by multiplication with x_2 (the inverse of x_1).

Assuming that we are given some variables x_1, x_2 and c_1, c_2, we define a function $[i]$ such that:

$$\begin{aligned} & [\mathsf{incr}(c_k); \mathsf{JUMP}(l_j)] && = c_k := c_k \times x_1; \mathsf{if}\ c_k = 1\ \mathsf{then}\ Loop\ \mathsf{else}\ r := j \\ & [\mathsf{decr}(c_k); \mathsf{JUMP}(l_j)] && = \mathsf{if}\ c_k = 1\ \mathsf{then}\ Loop\ \mathsf{else}\ c_k := c_k \times x_2; r := j \\ & [\mathsf{if}\ c_k = 0\ \mathsf{then}\ \mathsf{JUMP}(l_s)\ \mathsf{else}\ \mathsf{JUMP}(l_t)] && = \mathsf{if}\ c_k = 0\ \mathsf{then}\ r := s\ \mathsf{else}\ r := t \end{aligned}$$

The final program P is then:

$$
\begin{array}{l}
x_1, x_2 \xleftarrow{\$} \{x_1 x_2 = 1\} \\
c_1 := x_1^{n_1}; c_2 := x_1^{n_2}; r := s \\
\text{while } r \neq 1 \text{ do} \\
\quad \text{if } r = 2 \text{ then} \\
\quad\quad [i_2] \\
\quad \ldots \\
\quad \text{if } r = m \text{ then} \\
\quad\quad [i_m] \\
\text{return } 0
\end{array}
$$

To conclude the proof, we now prove that

$$P \approx_{\mathbb{F}_{q^\infty}} Loop \Leftrightarrow \text{M does not halt on input } (n_1, n_2), i_s$$

It is clear that without an overflow, i.e., when the multiplicative group generated by x_1 is big enough to avoid the case $c_k = 1$ in the encodings of incr, P perfectly simulates the behaviour of M, and terminates if and only if M terminates.

Let us assume that M does not halt on input $(n_1, n_2), i_s$. Given an interpretation $\mathbb{F}_{q^k}$, and a sampled value x_1, x_2, we have counters that can evolve in the cyclic multiplicative group generated by x_1, of some size $q^{k'}$. For any such k', either the simulation of M will create an overflow (increasing a counter over k'), and then P does not terminate. Else, there is a loop of instructions in M, which will be perfectly mimicked by P, which then does not terminate. Thus, for any interpretation and any random samplings, P does not terminate, and then $P \approx_{\mathbb{F}_{q^\infty}} Loop$.

Let us assume that M does halt on this input. We have an upper bound K on the values of the counter during the execution. Thus, there exists some k such that $q^k > K$, and there exists a random sampling of x_1 such that its generated multiplicative group is of size q^k. Then, the execution of P simulating P will not overflow, and P will terminate, going out of the while loop and returning 0. This execution is then a witness that $P \not\approx_{\mathbb{F}_{q^k}} Loop$, and thus $P \not\approx_{\mathbb{F}_{q^\infty}} Loop$

□

7 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 program indistinguishability is open, for programs that do not return a boolean. It asks if the statistical distance between the distributions of two programs is negligible in k. This would have direct applications in provable security.

ACKNOWLEDGMENTS

The authors would like to thank Joël Ouaknine for providing many interesting insights into LRS. The authors would also like to thank the anonymous referees for their feedback, that lead to a significant improvement of the paper.

REFERENCES

[1] Shaull Almagor, Brynmor Chapman, Mehran Hosseini, Joël Ouaknine, and James Worrell. 2018. Effective Divergence Analysis for Linear Recurrence Sequences. In *29th International Conference on Concurrency Theory (CONCUR 2018) (Leibniz International Proceedings in Informatics (LIPIcs))*, Sven Schewe and Lijun Zhang (Eds.), Vol. 118. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 42:1–42:15. https://doi.org/10.4230/LIPIcs.CONCUR.2018.42

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

[3] Gilles Barthe, Rohit Chadha, Vishal Jagannath, A Prasad Sistla, and Mahesh Viswanathan. 2020. Deciding differential privacy for programs with finite inputs and outputs. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*. 141–154.

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

[5] Gilles Barthe, Benjamin Grégoire, Charlie Jacomme, Steve Kremer, and Pierre-Yves Strub. 2019. Symbolic Methods in Computational Cryptography Proofs. In *32nd IEEE Computer Security Foundations Symposium (CSF'19)*. IEEE Computer Society, 136–151.

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

[7] Gilles Barthe, Charlie Jacomme, and Steve Kremer. 2020. Universal equivalence and majority on probabilistic programs over finite fields. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'20)*, Naoki Kobayashi (Ed.). ACM, Saarbrücken. To appear.

[8] Benjamin Bichsel, Timon Gehr, and Martin Vechev. 2018. Fine-grained Semantics for Probabilistic Programs. In *27th European Symposium on Programming (ESOP'18) (Lecture Notes in Computer Science)*, Vol. 10801. Springer, 145–185.

[9] Enrico Bombieri. 1978. On exponential sums in finite fields, II. *Inventiones mathematicae* 47, 1 (1978), 29–39.

[10] Mark Braverman. 2006. Termination of integer linear programs. In *International Conference on Computer Aided Verification*. Springer, 372–385.

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

[12] Antoine Chambert-Loir. 2006. Compter (rapidement) le nombre de solutions d'équations dans les corps finis.

[13] Qi Cheng, J. Maurice Rojas, and Daqing Wan. 2020. Computing zeta functions of large polynomial systems over finite fields. arXiv:2007.13214 [math.NT]

[14] Dmitry Chistikov, Andrzej S Murawski, and David Purser. 2019. Asymmetric Distances for Approximate Differential Privacy. In *30th International Conference on Concurrency Theory (CONCUR 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[15] Benny Chor, Oded Goldreich, Johan Håstad, Joel Friedman, Steven Rudich, and Roman Smolensky. 1985. The Bit Extraction Problem of t-Resilient Functions (Preliminary Version). In *26th Annual Symposium on Foundations of Computer Science (FOCS'85)*. IEEE Computer Society, 396–407.

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

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

[18] Matthew Fredrikson and Somesh Jha. 2014. 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)*. ACM, 1–10.

[19] Marco Gaboardi, Kobbi Nissim, and David Purser. 2020. The Complexity of Verifying Loop-Free Programs as Differentially Private. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020) (Leibniz International Proceedings in Informatics (LIPIcs))*, Artur Czumaj, Anuj Dawar, and Emanuela Merelli (Eds.), Vol. 168. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 129:1–129:17. https://doi.org/10.4230/LIPIcs.ICALP.2020.129

[20] Oded Goldreich. 2005. *Foundations of cryptography: a primer*. Vol. 1. Now Publishers Inc.

[21] William Andrew Johnson. 2016. *Fun with fields*. Ph.D. Dissertation. UC Berkeley.

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

[23] Catarina Kiefe. 1976. Sets definable over finite fields: their zeta-functions. *Trans. Amer. Math. Soc.* 223 (1976), 45–59.

[24] Alan GB Lauder and Daqing Wan. 2008. Counting points on varieties over finite fields of small characteristic. In *Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography*. Cambridge University Press, 579–612.

[25] Axel Legay, Andrzej S Murawski, Joël Ouaknine, and James Worrell. 2008. On automated verification of probabilistic programs. In *14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*

(TACAS'08) (Lecture Notes in Computer Science), Vol. 4963. Springer, 173–187.
[26] Rudolf Lidl and Harald Niederreiter. 1983. *Finite fields.* Addison-Wesley.
[27] Yehuda Lindell. 2017. How to Simulate It - A Tutorial on the Simulation Proof Technique. In *Tutorials on the Foundations of Cryptography*, Yehuda Lindell (Ed.). Springer International Publishing, 277–346. https://doi.org/10.1007/978-3-319-57048-8_6
[28] Michael L Littman, Judy Goldsmith, and Martin Mundhenk. 1998. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research* 9 (1998), 1–36.
[29] Stefan Maubach. 2001. Polynomial automorphisms over finite fields. *Serdica Math. J* 27 (2001), 343–350.
[30] Scott McCallum and Volker Weispfenning. 2012. Deciding polynomial-transcendental problems. *Journal of Symbolic Computation* 47, 1 (2012), 16–31.
[31] Maurice Mignotte, Tarlok Nath Shorey, and Robert Tijdeman. 1984. The distance between terms of an algebraic recurrence sequence. *Journal für die reine und angewandte Mathematik* 349 (1984), 63–76.
[32] Gary L Mullen and Daniel Panario. 2013. *Handbook of finite fields.* Chapman and Hall/CRC.
[33] Andrzej S. Murawski and Joël Ouaknine. 2005. On Probabilistic Program Equivalence and Refinement. In *CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings (Lecture Notes in Computer Science)*, Martín Abadi and Luca de Alfaro (Eds.), Vol. 3653. Springer, 156–170. https://doi.org/10.1007/11539452_15
[34] Tobias Nipkow. 1990. Unification in Primal Algebras, Their Powers and Their Varieties. *J. ACM* 37, 4 (Oct. 1990), 742–776.
[35] Joël Ouaknine and James Worrell. 2012. Decision problems for linear recurrence sequences. In *6th International Workshop on Reachability Problems (RP'12) (Lecture Notes in Computer Science)*, Vol. 7550. Springer, 21–28.
[36] Joël Ouaknine and James Worrell. 2014. Positivity problems for low-order linear recurrence sequences. In *25th ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*. Society for Industrial and Applied Mathematics, 366–379.
[37] Joël Ouaknine and James Worrell. 2014. Ultimate positivity is decidable for simple linear recurrence sequences. In *International Colloquium on Automata, Languages, and Programming*. Springer, 330–341.
[38] Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22, 11 (Nov. 1979), 612–613. https://doi.org/10.1145/359168.359176
[39] Jacobo Toràn. 1988. An oracle characterization of the counting hierarchy. In *3rd Annual Structure in Complexity Theory Conference*. 213–223.
[40] Jacobo Torán. 1991. Complexity classes defined by counting quantifiers. *J. ACM* 38 (1991), 753–774.
[41] NK Vereshchagin. 1985. The problem of appearance of a zero in a linear recurrence sequence. *Mat. Zametki* 38, 2 (1985), 609–615.
[42] André Weil. 1949. 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 [5].

Lemma A.1. *Let $P_1, Q_1 \in \mathcal{P}_{\mathbb{F}_q}(I,R)$, $P_2, Q_2 \in \overline{\mathcal{P}}_{\mathbb{F}_q}(I,R)$. 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:*

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

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

$$\begin{aligned}
& P_1 \mid P_2 \approx_{\mathbb{F}_{q^k}} Q_1 \mid Q_2 \\
\overset{(1)}{\iff} \quad & \forall \vec{i} \in \mathbb{F}_{q^k}^{|I|}. \forall c \in \mathbb{F}_{q^k}^{n} [\![(P_1, P_2)]\!]_{\vec{i}}^{\mathbb{F}_{q^k}}(c, \vec{0}) = [\![(Q_1, Q_2)]\!]_{\vec{i}}^{\mathbb{F}_{q^k}}(c, \vec{0}) \\
\overset{(2)}{\iff} \quad & \forall \vec{t} \in \mathbb{F}_{q^k}^{|I|}. \forall c \in \mathbb{F}_{q^k}^{n} [\![(P_1\sigma, P_2\sigma, R_I)]\!]^{\mathbb{F}_{q^k}}(c, \vec{0}, \vec{t}) = [\![(Q_1\sigma, Q_2\sigma, R_I)]\!]^{\mathbb{F}_{q^k}}(c, \vec{0}, \vec{t}) \\
\overset{(3)}{\iff} \quad & \forall c' \in \mathbb{F}_{q^k}^{n+|I|} [\![((P_1\sigma, R_I), P_2\sigma)]\!]^{\mathbb{F}_{q^k}}(c', \vec{0}) = [\![((Q_1\sigma, R_I), Q_2\sigma)]\!]^{\mathbb{F}_{q^k}}(c', \vec{0}) \\
\overset{(4)}{\iff} \quad & (P_1\sigma, R_I) \mid P_2\sigma \approx_{\mathbb{F}_{q^k}} (Q_1\sigma, R_I) \mid Q_2\sigma
\end{aligned}$$

Each equivalence is justified as follows:

(1) This is the expansion of the definition of conditional equivalence.
(2) For any $\vec{i} \in \mathbb{F}_{q^k}^{|I|}$, $[[I]]_{\vec{i}}^{\mathbb{F}_{q^k}}(\vec{i}) = 1$, and we have that $[[(P_1, P_2)]]_{\vec{i}}^{\mathbb{F}_{q^k}}(c, \vec{0}) = [[(P_1, P_2, I)]]_{\vec{i}}^{\mathbb{F}_{q^k}}(c, \vec{0}, \vec{i})$. Furthermore, in the R_I is equal to $\vec{i}$, $P_1\sigma, P_2\sigma$ of course returns the same value as P_1, P_2 that are given $\vec{i}$ as input. We also have that $[[R_I]]^{\mathbb{F}_{q^k}}(\vec{i}) = \frac{1}{q^{k\times|I|}}$, and we get

$$[[(P_1\sigma, P_2\sigma, R_I)]]^{\mathbb{F}_{q^k}}(c, \vec{0}, \vec{i}) = \frac{1}{q^{k\times|I|}} \times [[(P_1, P_2, I)]]_{\vec{i}}^{\mathbb{F}_{q^k}}(c, \vec{0}, \vec{i})$$

Intuitively, we are replacing input variables that have a fix value by a uniform distribution, and thus we reduce by the size of the space the possibility that we obtain the value given by $\vec{i}$. Performing the same operation on the other side and simplifying by $\frac{1}{q^{k\times|I|}}$ yields the expected result.
(3) We are setting $c' = (c, \vec{t})$.
(4) By definition.

□

Corollary A.2. *For any $k \in \mathbb{N}$, $\mathbb{F}_{q^k}$-*equivalence *and $\mathbb{F}_{q^k}$-*conditional equivalence *restricted to programs of fixed arity and without inputs are in* $\mathsf{C_{=}P}$.

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

For some c, we have that verifying if

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

is in $\mathsf{C_{=}P}$. As $\mathsf{C_{=}P}$ is closed under finite intersection [39], we can decide in $\mathsf{C_{=}P}$ if:

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

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

Corollary 3.4. *$\mathbb{F}_{q^k}$-*equivalence *and $\mathbb{F}_{q^k}$-*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 3.2. Let $P_1, Q_1 \in \mathcal{P}_{\mathbb{F}_q}(\emptyset, R)$,$P_2, Q_2 \in \overline{\mathcal{P}}_{\mathbb{F}_q}(\emptyset, R)$ with $|P_1| = |Q_1| = n$.

Now, by unfolding the definition,

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

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

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

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

The decision problem is then directly inside $\mathsf{coNP}^{\mathsf{C_{=}P}}$, where coNP is required for the universal quantification over $c \in \mathbb{F}_{q^k}^n$. □

Lemma A.3. *Given a CNF formula $\phi(I, R)$ over two sets of variables and $(\vee, \wedge)$, we can produce in polynomial time a program $P \in \mathcal{P}_{\mathbb{F}_2}(I, R)$ 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, $P = \phi' \in \mathcal{P}_{\mathbb{F}_2}(I,R)$ is a program equivalent to ϕ. □

Lemma 3.5. $\mathbb{F}_2$-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 $P = \phi' \in \mathcal{P}_{\mathbb{F}_2}(I,R)$ obtained according to Lemma A.3. Given a fresh random variable r:

$$\begin{aligned} P(I,R) \approx_{\mathbb{F}_2} r &\Leftrightarrow \text{for all valuations of } I\text{, P returns true} \\ &\quad\ \text{for half of the valuations of } R \\ &\Leftrightarrow \text{for all valuations of } I\text{, } \phi \text{ is true} \\ &\quad\ \text{for half of the valuations of } R \\ &\Leftrightarrow \phi(I,R) \in \mathsf{A{-}halfSAT} \end{aligned}$$

□

Lemma 3.8. *Let $P_1,\dots,P_n$ be programs over $\mathcal{P}_{\mathbb{F}_{q^k}}(I,R)$, and $Y \subset R$.*

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

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

Proof.

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

This proof only relies on the observation that fixing the value of the variables in Y and asking for the equality of distributions for all inputs, is the same as asking that for all values of Y and all inputs, we have the equality of distributions. In the middle equivalence, we are essentially setting $\vec{j} = (\vec{i},\vec{i'})$. □

Lemma 3.9. *Let $P_1,\dots,P_n$ be programs over $\mathcal{P}_{\mathbb{F}_{q^k}}(I,\{r_1,\dots,r_m\})$*

$$\perp_{\mathbb{F}_{q^k}}(P_1,\dots,P_n) \Leftrightarrow (P_1,\dots,P_n) \approx_{\mathbb{F}_{q^k}} (P_1\sigma_1,\dots,P_n\sigma_n)$$

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

Proof.

$$\begin{aligned} \perp_{\mathbb{F}_{q^k}}(P_1,\dots,P_n) &\overset{(1)}{\Longleftrightarrow} \forall \vec{i} \in \mathbb{F}_{q^k}^{|X|}.\ [\![P_1,\dots,P_n]\!]^{\mathbb{F}_{q^k}}_{\vec{i}} = ([\![P_1]\!]^{\mathbb{F}_{q^k}}_{\vec{i}},\dots,[\![P_n]\!]^{\mathbb{F}_{q^k}}_{\vec{i}}) \\ &\overset{(2)}{\Longleftrightarrow} \forall \vec{i} \in \mathbb{F}_{q^k}^{|X|}.\ [\![P_1,\dots,P_n]\!]^{\mathbb{F}_{q^k}}_{\vec{i}} = ([\![P_1\sigma_1]\!]^{\mathbb{F}_{q^k}}_{\vec{i}},\dots,[\![P_n\sigma_n]\!]^{\mathbb{F}_{q^k}}_{\vec{i}}) \\ &\overset{(3)}{\Longleftrightarrow} \forall \vec{i} \in \mathbb{F}_{q^k}^{|X|}.\ [\![P_1,\dots,P_n]\!]^{\mathbb{F}_{q^k}}_{\vec{i}} = [\![(P_1\sigma_1,\dots,P_n\sigma_n)]\!]^{\mathbb{F}_{q^k}}_{\vec{i}} \\ &\overset{(4)}{\Longleftrightarrow} (P_1,\dots,P) \approx_{\mathbb{F}_{q^k}} (P_1\sigma_1,\dots,P_n\sigma_n) \end{aligned}$$

Each equivalence is justified as follows.

(1) By definition of independence of distributions.
(2) For any $\vec{i} \in \mathbb{F}_{q^k}^{|X|}$ we have that $[[P_i]]_{\vec{i}}^{\mathbb{F}_{q^k}} = [[P_i\sigma_i]]_{\vec{i}}^{\mathbb{F}_{q^k}}$ as we are only renaming random variables.
(3) The programs $P_1\sigma_1, \ldots, P_n\sigma_n$ do not share any random variable, and thus trivially verify:

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

(4) By definition of independence.

□

Theorem 3.12. *$\mathbb{F}_{q^k}$-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 $P = \phi' \in \mathcal{P}_{\mathbb{F}_2}(I, R)$ obtained according to Lemma A.3. With r a fresh random variable, recall that:

$$\begin{aligned} P \approx_{\mathbb{F}_2} r &\Leftrightarrow \text{for all valuation of } I,\ \phi \text{ is true for half of the valuation of } R \\ &\Leftrightarrow \phi(I, R) \in \mathsf{A{-}halfSAT} \end{aligned}$$

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

$$\begin{aligned} P \approx_{\mathbb{F}_2} r &\overset{(1)}{\Longleftrightarrow} P + x \approx_{\mathbb{F}_2} r + x \\ &\overset{(2)}{\Longleftrightarrow} P + x \approx_{\mathbb{F}_2} r \\ &\overset{(3)}{\Longleftrightarrow} (P + r', r') \approx_{\mathbb{F}_2} (r, r') \\ &\overset{(4)}{\Longleftrightarrow} (P + r', r') \approx_{\mathbb{F}_2} (P + r, r') \\ &\overset{(5)}{\Longleftrightarrow} \perp_{\mathbb{F}_2}^{\emptyset} (P + r', r') \end{aligned}$$

Where each equivalence is justified as follows.

(1) We apply the bijection $u \mapsto u + x$ on both sides, thus preserving the equality of distributions.
(2) The distribution $r + x$ is equal to the distribution r, as shifting the uniform distribution yields the uniform distribution.
(3) By application of Lemma A.1, we replace the input x by a random variable r' synchronized on both sides.
(4) We once again have that r and $P + r$ have the same distribution, as r does not occur in P, and thus shifting the uniform distribution yields the uniform distribution.
(5) By application of Lemma 3.9, where the substitution σ_1 maps r' to r.

And thus, we conclude with:

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

□

Lemma 3.13. *For any $k \in \mathbb{N}$, $\mathbb{F}_{q^k}$-0-majority reduces in polynomial time to $\mathbb{F}_{q^k}$-0-majority with $r = 1$.*

Proof. For any n, we first show how to construct a program T_n such that $[[T_n]]^{\mathbb{F}_{q^k}}(\vec{0}) = \frac{1}{n}$. There exists m such that $q^m > n$. For such m, let us denote by D_n any subset of $\mathbb{F}_q^m$ such that $|D_n| = n$. If

we denote by d_n a fixed element of D_n, let T_n be the program:

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

We of course have by construction $[\![T_n]\!]^{\mathbb{F}_{q^k}}(\vec{0}) = \frac{1}{n}$, as we sample inside a set of size n, and test equality with a given element of this set. 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:

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

□

Lemma 3.14. *For any $k \in \mathbb{N}$, $\mathbb{F}_{q^k}$-0-majority restricted to inputless programs is* PP*-complete.*

Proof. Membership

Let $P, Q \in \mathcal{P}_{\mathbb{F}_q}(\emptyset, R)$. Let us reuse the polynomial time Turing Machine M defined in Lemma 3.3. Given P_1, P_2, Q_1, Q_2 and $\vec{c}$, it was such that:

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

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

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

Thus, we do have:

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

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 $P = \phi' \in \mathcal{P}_{\mathbb{F}_2}(R)$ obtained according to Lemma A.3. We then have:

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

□

Lemma 3.15. *$\mathbb{F}_{q^k}$-majority is* coNP^{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 3.14 to obtain a polynomial $P \in \mathcal{P}_{\mathbb{F}_2}(I, R)$ whose truth value is equivalent of ϕ.

We have, for some variable r inside R:

$$\phi \in \text{A–MINSAT} \Leftrightarrow r \prec_{\mathbb{F}_2} P$$

Membership

Let $P, Q \in \mathcal{P}_{\mathbb{F}_{q^k}}(I, R)$. We slightly modify M from Lemma 3.14, 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:

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

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

Lemma 4.3. $\mathbb{F}_{q^\infty}$-equivalence *restricted to linear programs is in* PTIME.

Proof. Without loss of generality, we only consider programs without input variables (Lemma 3.2).

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

- if $\text{vars}(P_i) \cap S \neq \emptyset$, let $r = \min(\text{vars}(P_i) \cap S)$ 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 $O(n \times |R|)$ 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 4.10. $\mathbb{F}_{q^\infty}$-conditional equivalence *and* $\mathbb{F}_{q^\infty}$-equivalence *restricted to arithmetic programs are in* EXP.

Proof. [24, 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 A.4. *Let $P, Q \in \overline{\mathcal{P}}_{\mathbb{F}_2}(\emptyset, R)$ without any multiplication.*

$$P \approx_{\mathbb{F}_2} Q \Leftrightarrow P \approx_{\mathbb{F}_{2^\infty}} Q$$

Proof.
$\Leftarrow$ Trivial direction.
$\Rightarrow$ As outlined in [4], one can decide if $P \approx_{\mathbb{F}_2} Q$ 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 $P \approx_{\mathbb{F}_{2^\infty}} Q$.

□

Lemma A.5. *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.
$\underline{b := B = 0}$ Direct, with $X' = X$ and $B_1 = B$.
$\underline{b := B' \neq 0}$ 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 $tB = 1$.
$\underline{b := \bigvee_{1 \leq i \leq l} B_i = 0}$

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

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

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

$\underline{b := b_1 \vee b_2}$ 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 4.11. *For any $k \in \mathbb{N} \cup \{\infty\}$, $\mathbb{F}_{q^k}$-*conditional equivalence *restricted to programs without failures reduces in exponential time to $\mathbb{F}_{q^k}$-*conditional equivalence *restricted to arithmetic programs.*

Proof. Let $P_1, Q_1 \in \mathcal{P}_{\mathbb{F}_q}(\emptyset, R), P_2, Q_2 \in \overline{\mathcal{P}}_{\mathbb{F}_q}(\emptyset, R)$, 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 $P_1 := C[\mathsf{if}\ B \neq 0\ \mathsf{then}\ P_1^t\ \mathsf{else}\ P_1^f]$ 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):

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

We then have that:

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

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

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

Now, for any k and c we have that:

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

Indeed, for any variable t and polynomial B:

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

Finally:

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

Putting everything together, we get that:

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

And we finally have:

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

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

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

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 `fail` 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 $P_1 := \text{if } b \text{ then } P_1^t \text{ else } \texttt{fail}$ for some program P_1^t which cannot fail.

Now, with Lemma A.5, 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:

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

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

$$\text{if } b \text{ then } P_1^t \text{ else } \texttt{fail} \mid P_2 \approx_{\mathbb{F}_{q^k}} Q_1 \mid Q_2 \Leftrightarrow P_1^t \mid P_2, B_1, \dots, B_n \approx_{\mathbb{F}_{q^k}} Q_1 \mid Q_2, B_1, \dots, B_n$$

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

Lemma 4.14. $\mathbb{F}_q$-equivalence *reduces in polynomial time to* $\mathbb{F}_{q^\infty}$-equivalence. $\mathbb{F}_{q^\infty}$-equivalence *is thus* $\mathsf{coNP}^{\mathsf{C}=\mathsf{P}}$*-hard.*

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

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

□

Lemma A.6. *Given $l \in \mathbb{N}$, with $n = \sharp I + \sharp R$,*

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

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 $\bot$. We finally obtain the expected result.

□

Lemma 4.21. *For any $l \in \mathbb{N}$, $k \in \mathbb{N} \cup \{\infty\}$, $\mathbb{F}_{q^k}$-*equivalence *reduces in polynomial time to $l, \mathbb{F}_{q^k}$-*simulatability.

Proof. Let $P, Q \in \mathcal{P}_{\mathbb{F}_q}(I, R)$. 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 $P \approx_{\mathbb{F}_{q^k}} Q$, then we trivially have $a + bP \sqsubseteq^1_{q^k} a + bQ$ with S as the identity. Let us assume that we have $S \in \mathcal{P}_{\mathbb{F}_q}(\{i\}, R')$ such that $\deg(S) \leq l$ and $S[{}^{a+bQ}/_i] \approx_{\mathbb{F}_{q^k}} a + bP$. 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 4.17. *$\mathbb{F}_{q^\infty}$-0-*majority *restricted to linear programs is in* PTIME.

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

Given $P_1, \dots, P_n \in \overline{\mathcal{P}}_{\mathbb{F}_q}(\emptyset, R)$ 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 $[[P]]^{\mathbb{F}_{q^k}}(\vec{c}) = \begin{cases} \frac{1}{q^{k \times |I_P|}} & \text{if the linear constraints are satisfiable} \\ 0 & \text{else} \end{cases}$ 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 $P, Q \in \overline{\mathcal{P}}_{\mathbb{F}_q}(\emptyset, R)$ without multiplications, we know that:

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

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

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

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

Lemma 4.18. $\mathbb{F}_{2^\infty}$-0-majority *is* PP-*hard.*

Proof. We prove that $\mathbb{F}_2$-0-*majority* reduces to $\mathbb{F}_{2^\infty}$-0-*majority* in polynomial time.
Let $P, Q \in \mathcal{P}_{\mathbb{F}_2}(\emptyset, R)$.

$$
\begin{aligned}
P \prec^{1}_{\mathbb{F}_2} Q &\overset{(1)}{\iff} \left|\{X \in \mathbb{F}_2^m \mid P(X) = \vec{0}\}\right| \leq \left|\{X \in \mathbb{F}_2^m \mid q(X) = \vec{0}\}\right| \\
&\overset{(2)}{\iff} \forall k.\ \left|\{X \in \mathbb{F}_{2^k}^m \mid P(X) = \vec{0} \wedge X \in \mathbb{F}_2^m\}\right| \leq \left|\{X \in \mathbb{F}_{2^k}^m \mid Q(X) = \vec{0} \wedge X \in \mathbb{F}_2^m\}\right| \\
&\overset{(3)}{\iff} \forall k.\ \left|\{X \in \mathbb{F}_{2^k}^m \mid P(X) = \vec{0} \wedge x_1(x_1+1) = 0 \wedge \cdots \wedge x_m(x_m+1) = 0\}\right| \\
&\qquad \leq \left|\{X \in \mathbb{F}_{2^k}^m \mid Q(X) = \vec{0} \wedge x_1(x_1+1) = 0 \wedge \cdots \wedge x_m(x_m+1) = 0\}\right| \\
&\overset{(4)}{\iff} \forall k.\ \left|\{X \in \mathbb{F}_{2^k}^m \mid (P(X), x_1(x_1+1), \ldots, x_m(x_m+1)) = \vec{0}\}\right| \\
&\qquad \leq \left|\{X \in \mathbb{F}_{2^k}^m \mid (Q(X), x_1(x_1+1), \ldots, x_m(x_m+1)) = \vec{0}\}\right| \\
&\overset{(5)}{\iff} (P, x_1(x_1+1), \ldots, x_m(x_m+1)) \prec^{1}_{\mathbb{F}_{2^\infty}} (Q, x_1(x_1+1), \ldots, x_m(x_m+1))
\end{aligned}
$$

Each equivalence is justified as follows.

(1) By unfolding the definition.
(2) We quantifiy over all extensions of the field, but condition on X being in the original field.
(3) We express the fact that all components of X are in $\mathbb{F}_2$ by saying that they must either be equal to 0 or 1, and thus be a root of the polynom x(x+1).
(4) We express this condition as being part of the program.
(5) By unfolding the definition.

□