CONSTRAINT SYSTEM DESIGN

WORKING NOTES

Tristan Freiberg, Inference Labs Inc.
tristan@inferencelabs.com[1]

PREFACE

These notes define and study *arithmetic constraint systems*, which occupy a liminal space between arithmetic circuits and the computations they represent. We do *not* mean Rank-1 Constraint Systems or Quadratic Arithmetic Programs. Our focus lies at the frontend abstraction layer, exemplified by the *Expander Compiler Collection* [3], where computations are expressed through a programming interface exposing methods like `api.add` and `api.mul` that denote finite-field operations. These instructions are compiled into arithmetic circuits and then passed to a backend proof system such as *Expander* [2].

The goal is to formalize the mathematics that implicitly guides frontend development in zero-knowledge systems, presented in a didactic, self-contained way. While we occasionally reference backend concepts such as proofs, verification, and commitments, this serves only to motivate definitions that may at times depart from standard formulations. No prior knowledge of cryptography is assumed.

The computations we ultimately wish to verify are typically integer-based, yet arithmetic circuits describe relations over finite fields. Interpreting what a field-based circuit says about integer computations is subtle and central to constraint-system design. Our definitions therefore address not only the *syntax* of constraints (the symbolic expressions that specify them) but also their *semantics* (the domains and models in which they are satisfied). This motivates notions such as ambient domain and admissibility.

These notes are part of an ongoing effort to articulate the mathematics of constraint-system design. The author hopes they will be useful to practitioners and researchers in verifiable computation, helping to bring greater rigor to constraint-system design and to facilitate clearer communication of ideas, constructions, and code. Corrections or feedback of any kind are warmly welcome; please contact the author, who is solely responsible for all remaining errors or omissions.

ACKNOWLEDGMENTS

The author thanks the entire team at Inference Labs for their encouragement and support, and especially Jonathan Gold for many insightful discussions and clarifying feedback that greatly improved the presentation.

[1] Corrections and feedback welcome.

1. BASIC CONCEPTS

1.1. Context and motivating examples. Arithmetic circuits provide a way to describe and verify computations over finite fields. They consist of a fixed set of gates, each performing a basic arithmetic operation (addition or multiplication) over a finite field, usually of prime order. Given a set of inputs and a corresponding circuit, the output is computed deterministically by evaluating the gates in topological order. In zero-knowledge proof systems, such circuits are used to represent computations in a form that can be efficiently verified cryptographically.

However, in practice, especially when working with high-level circuit construction tools, engineers do not write circuits directly. Instead, they specify constraints that the inputs and intermediate values must satisfy, and the underlying system compiles these constraints into an arithmetic circuit (or some other low-level representation suitable for proof generation).

In this setting, constraints take the form

$$P(x_1,\dots,x_n) \equiv 0 \bmod p, \tag{1.1}$$

where p is a large prime and P is a polynomial in n indeterminates with integer coefficients. Given a subset D of $\mathbb{Z}^n$, the naive goal of constraint design is to describe a set C of integer polynomials whose mod p vanishing set is precisely D, i.e. $(a_1,\dots,a_n) \in D$ if and only if $P(a_1,\dots,a_n) \equiv 0 \bmod p$ for all P in C.

For example, suppose x is an integer and we wish to verify that $x = 0$ within an arithmetic circuit. This is impossible. An arithmetic circuit may give rise to the constraint $x \equiv 0 \bmod p$, and while $x = 0$ certainly implies this congruence, the converse does not hold: the condition $x \equiv 0 \bmod p$ implies only that $x \in p\mathbb{Z}$. More generally, if a system of congruences of the form (1.1) is satisfied by an integer n-tuple $(a_1,\dots,a_n)$, then it is also satisfied by every element of the infinite set

$$\{(a_1 + k_1 p,\dots,a_n + k_n p) : (k_1,\dots,k_n) \in \mathbb{Z}^n\}.$$

Thus, constraint systems arising from arithmetic circuits over finite fields cannot directly encode even the simplest of statements about integers. They *can* capture statements about integers *under certain assumptions* on the possible values those integers may take. For example, *assuming* $-(p-1) \leqslant x \leqslant p-1$, the congruence $x \equiv 0 \bmod p$ *does* imply $x = 0$. Similarly, if an integer n-tuple $(a_1,\dots,a_n)$ satisfies a system of polynomial congruences mod p, *and* we assume each a_i lies in a specified complete set of representatives modulo p, such as the least nonnegative residues $\{0,\dots,p-1\}$ or the balanced residues $\{-(p-1)/2,\dots,(p-1)/2\}$, *then* we can uniquely determine each a_i from its class modulo p.

Example 1.1. (a) Consider the statement $0 \leqslant x \leqslant 15$. An integer x in this range satisfies the congruence

$$x(x-1)\cdots(x-15) \equiv 0 \bmod 101.$$

Conversely, since 101 is prime (and 0 is the only zero divisor in a field), any integer x satisfying this congruence belongs to the set $\{0,\dots,15\} + 101\mathbb{Z}$. Assuming that $-50 \leqslant x \leqslant 50$, we may conclude from this that $x \in \{0,\dots,15\}$. In fact, under the weaker assumption that $-85 \leqslant x \leqslant 100$, we may still conclude that $x \in \{0,\dots,15\}$.

(b) Alternatively, we may proceed as follows. Noting that $15 = 2^4 - 1$, we express x in binary truncated at the fourth least significant bit. That is, for any integer x, we may write

$$x = b_0 + 2b_1 + 4b_2 + 8b_3 + 16q,$$

where each $b_i \in \{0,1\}$ and $q \in \mathbb{Z}$. If $0 \leqslant x \leqslant 15$, then necessarily $q = 0$, and so

$$x \equiv b_0 + 2b_1 + 4b_2 + 8b_3 \bmod 101 \quad \text{and} \quad b_i(b_i - 1) \equiv 0 \bmod 101, \quad i \in \{0,1,2,3\}. \tag{1.2}$$

Conversely, if these five congruences hold, we may deduce that each b_i lies in $\{0,1\} + 101\mathbb{Z}$, and more importantly, that $x \in \{0,\dots,15\} + 101\mathbb{Z}$. Thus, under the assumption that $x \in \{-50,\dots,50\}$, it follows that $x \in \{0,\dots,15\}$. While we are not directly interested in the values of the individual b_i, if we additionally assume $-50 \leqslant b_i \leqslant 50$, we may conclude $b_i \in \{0,1\}$. Indeed, the same conclusions hold under the weaker assumptions $x \in \{-85,\dots,100\}$ and $b_i \in \{-99,\dots,100\}$. ■

Remark 1.2. The five congruences (1.2) may be written as

$$x - (b_0 + 2b_1 + 4b_2 + 8b_3) \equiv 0 \bmod p \quad \text{and} \quad 0x + \prod_{j=0}^{3} \delta_{ij} b_j (b_j - 1) \equiv 0 \bmod p, \quad i \in \{0,1,2,3\},$$

where δ_{ij} equals 1 if $i = j$ and 0 otherwise. Thus, each congruence is indeed of the form $P(x, b_0, b_1, b_2, b_3) \equiv 0 \bmod p$, as in (1.1). The complete solution set is

$$(\{0,\dots,15\} + 101\mathbb{Z}) \times (\{0,1\} + 101\mathbb{Z})^4. \tag{1.3}$$

■

In Example 1.1, the symbols x, b_0, b_1, b_2, b_3 are indeterminates, not integers: the congruences in (1.2) define a relation on the integers, explicitly given in (1.3). Any particular tuple in this set, such as $(7,1,1,1,0)$, corresponding to the binary expansion of $x = 7$, may be called a *witness* for the constraint system. In proof systems, a prover generates and supplies one such witness corresponding to the specific claim being proved, while the verifier checks that the submitted witness satisfies all the constraints.

Suppose we wish only to encode the upper bound $x \leqslant 15$ using polynomial congruences and assumptions about the possible range of values x may take. Either approach outlined in Example 1.1 can be used to encode the stronger condition $0 \leqslant x \leqslant 15$ (under the assumption that $x \in \{-85, \ldots, 100\}$ in the first place). This is acceptable *provided* that no admissible value of x is negative, for example if we know *a priori* that x represents a square.

In the language of interactive proofs, we distinguish between *provers* and *verifiers*, where provers may be *honest* or not. For the present discussion, let us say that provers make a claim about an integer x, and submit it as input to a constraint system designed to test whether the claim is valid. We typically assume that all provers, regardless of intent, can only submit inputs x lying in some finite range such as $\{-(p-1)/2, \ldots, (p-1)/2\}$. However, we may further assume that an *honest* prover submits x lying in a narrower range, such as $\{0, \ldots, (p-1)/2\}$.

In this setting, imposing constraints that accept (i.e. are satisfied by) only $x \in \{0, \ldots, 15\} + p\mathbb{Z}$ ensures that all honest provers pass verification, even though the verifier is logically interested only in whether $x \leqslant 15$. That a dishonest prover submitting a negative x might fail verification is merely a bonus—after all, the verifier's primary goal is to reject any $x \geqslant 16$.

To borrow from the nomenclature of interactive proofs, we would say that a constraint system is *complete* if it is satisfied by any honest input, and *sound* if it is not satisfied by any dishonest input.

1.2. Formal definitions. The preceding examples highlight a recurring theme: constraint systems defined by polynomial congruences can only express unambiguous statements about integers under certain assumptions on the values those integers may take. To make these assumptions precise, we introduce notation that distinguishes between three key sets: the *ambient domain* A, representing all integer n-tuples that can be submitted as inputs to a constraint system, whether by honest or dishonest provers; the *desired set* D, representing the tuples that are intended to satisfy a given claim or property; and the *admissible set* $H \subseteq A$, representing the tuples that an honest computation could in principle produce.

This triad allows us to define what it means for a constraint system to be *complete* or *sound* with respect to the intended property, and to characterize when it is over- or underconstrained. The following definitions formalize these ideas.

Definition 1.3. A *constraint system* in n indeterminates $x_1, \ldots, x_n$ is a finite set $C \subseteq \mathbb{Z}[x_1, \ldots, x_n]$ of polynomials with integer coefficients.

Given a positive integer p, the *mod p vanishing set* of C is

$$V_p(C) := \{(a_1, \ldots, a_n) \in \mathbb{Z}^n : P(a_1, \ldots, a_n) \equiv 0 \bmod p \text{ for all } P \in C\}. \tag{1.4}$$

When C or p is clear from context we may abbreviate the notation to V_p, $V(C)$, or simply V. We refer to congruences of the form $P(x_1, \ldots, x_n) \equiv 0 \bmod p$, where $P \in C$, or equivalent ones, as *constraints*. An integer n-tuple $a \in \mathbb{Z}^n$ is a *witness* for C modulo p if and only if $a \in V_p(C)$.

Let $A, D \subseteq \mathbb{Z}^n$, and $H \subseteq A$. We refer to A as the *ambient domain*, and we call an integer n-tuple $a \in \mathbb{Z}^n$ *admissible* if and only if $a \in H$.

For $a \in \mathbb{Z}^n$, we say that C *accepts a (over A modulo p)* if $a \in A \cap V_p(C)$, and that it *rejects a (over A modulo p)* otherwise.

The system C is said to be *complete* (with respect to A, D, H, and p) if it accepts (over A modulo p) every admissible integer n-tuple a lying in D; equivalently,

$$A \cap V_p(C) \supseteq D \cap H.$$

Thus C is not complete if it rejects some admissible $a \in D$, that is, if there exists $a \in D \cap H$ with $a \notin A \cap V_p(C)$.

The system C is said to be *sound* (with respect to A, D, H, and p) if it accepts (over A modulo p) only admissible a lying in D; equivalently,

$$A \cap V_p(C) \subseteq D \cap H.$$

Hence C is not sound if it accepts any integer n-tuple a that is either not in D or not admissible, that is, if there exists $a \in A \cap V_p(C)$ with $a \notin D \cap H$.

We say that C is *underconstrained* (with respect to A, D, H, and p) if it is complete but not sound: the system accepts all admissible $a \in D$, but also at least one a that is either not admissible or not in D. Conversely, C is *overconstrained* (with respect to A, D, H, and p) if it is sound but not complete: the system accepts only admissible $a \in D$, but fails to accept every admissible $a \in D$.

When the parameters A, D, H, and p are clear from context, we may omit them in phrases such as "with respect to A, D, H, and p" or "over A modulo p". ■

1.3. Remarks on the definition. The following remarks elaborate on the meaning and practical implications of Definition 1.3. They clarify how the formal objects A, C, D, H, and p relate to arithmetic circuits, witnesses, and integer semantics, and why certain conventions are adopted throughout.

The objects introduced in Definition 1.3 serve as the bridge between integer computations and their finite-field encodings in arithmetic circuits. Before proceeding to concrete examples, we clarify how these components interact in practice: what role the ambient domain and admissible set play, how completeness and soundness depend on chosen parameters, and how the integer and field viewpoints coexist in a unified framework.

1.3.1. Ambient domain and admissible inputs. Mirroring the language of arithmetic circuits and interactive proofs, it is convenient to think of a as being *submitted* or *supplied* to C as *input*. Because our definition quantifies over all $a \in \mathbb{Z}^n$, we must specify which portion of this space is relevant in practice. We think of the ambient domain A as the set of all a that can be supplied to C as input, irrespective of whether the corresponding computation is honest. This distinguishes A from the admissible set $H \subseteq A$, which may (but need not) be a proper subset of A. Restricting H to a proper subset of A amounts to external assumptions about what "honest" computations can produce, and should be done only when necessary to establish completeness. Of course, if $H = A$, then

$$A \cap V_p(C) = D \quad \Rightarrow \quad A \cap V_p(C) = D \cap A = D \cap H,$$

so if we can show that $A \cap V_p(C) = D$, we may leave admissibility out of the discussion of completeness and soundness.

Typically, each $A = I_1 \times \cdots \times I_n$, where each I_i is an integer interval of length at most p, such as the set of balanced residues modulo p, i.e. $I_i = \{-(p-1)/2, \ldots, (p-1)/2\}$. Occasionally, we assume instead the least nonnegative residues, $I_i = \{0, \ldots, p-1\}$. To cover both possibilities (and infinitely many others) at once, we often assume $I_i = \{h, h+1, \ldots, h+p-1\}$ for some integer h (for odd p, take $h = -(p-1)/2$ for balanced residues). We will see that there are situations—specifically, in the context of ancillary parameters—in which it is reasonable to allow $I_i = \mathbb{Z}$ for some i.

In practice, most systems implicitly interpret any integer outside these ranges by reducing it modulo p. From our perspective, this means that any input, whether honest or not, lying outside A is treated as its mod p representative within A. In effect, the computation is silently coerced into the valid range. A purportedly invalid input such as $a = -p$ is therefore interpreted as $a = 0$ (if we use balanced or least nonnegative residue representatives): the system here does not detect dishonesty but normalizes it, yielding a well-formed input within the ambient domain. Nevertheless, in our definition, we do not assume that A contains a unique representative of each residue class. When required, we interpret any input outside A as mapped to some chosen mod p representative within A.

If we really did need to encode an integer relationship like $99 + 2 = 101$ over $\mathbb{Z}/101\mathbb{Z}$, we would effectively need to change the computation itself. For example, we might represent the operands as digit vectors and introducing carry constraints, so that the modular wraparound $99 + 2 \equiv 0 \bmod 101$ becomes the integer relation $(0,9,9) + (0,0,2) = (1,0,1)$. In other words, when arithmetic is performed modulo p, the semantics of addition or multiplication depend on the representation of values, and reproducing true integer behavior may require an explicit modeling of carries or overflows within the constraint system.

1.3.2. Dependence on parameters, and terminology. The completeness or soundness of a constraint system is always relative to the parameters (A, D, H, p). A system C that is complete for one choice of ambient domain, admissible set, or modulus may fail to be complete for another. Designing constraint systems therefore involves not only specifying the polynomial relations in C, but also identifying the contextual assumptions under which those relations are meant to hold.

The terminology of completeness and soundness is borrowed from interactive proofs, though its meaning here is absolute rather than probabilistic. Our definitions describe exact set-theoretic inclusions among integer tuples, not acceptance probabilities over random transcripts. Nevertheless, when a constraint system is later compiled into an arithmetic circuit and embedded in an interactive proof, these notions correspond directly to the usual completeness and soundness guarantees of the resulting proof system.

Looking ahead, Definition 4.1 generalizes these definitions to incorporate *probabilistic completeness and soundness*, allowing us to reason about verification methods that involve deterministic randomness, such as LogUp and Freivalds' technique, directly at the level of constraint systems.

1.3.3. Integers, fields, and the lifting perspective. For a given prime p, we may assume without loss of generality that all polynomials P in a constraint system C have coefficients in $\{0, \ldots, p-1\}$. Indeed, once p is fixed, one could regard C as a subset of $(\mathbb{Z}/p\mathbb{Z})[x_1, \ldots, x_n]$. However, to study constraint systems abstractly, or to compare them across different moduli, we adopt a definition of C that is independent of p and therefore rooted in the integer ring $\mathbb{Z}$.

In practice, arithmetic circuits operate over a finite field such as $\mathbb{Z}/p\mathbb{Z}$, where elements are equivalence classes of integers modulo p. Although these classes are typically represented by their least nonnegative residues $\{0, \ldots, p-1\}$, the representation obscures the fact that infinitely many distinct integers correspond to the same field element. For example, in *Expander Compiler Collection* [3], the instruction `api.add(-1,1)` will raise an error: the value `-1` is not interpreted as $p-1$, nor as the residue

class $-1+p\mathbb{Z}$. Yet, the logical intent of an arithmetic circuit is to enforce relations among integers, possibly negative, rather than purely among symbolic field elements.

Conflating integers with field elements, even within the range $\{0,\dots,p-1\}$, can lead to semantic inconsistencies. For instance, if $p \geqslant 2$, the inequality $0 < p-1$ is true for integers but meaningless in $\mathbb{Z}/p\mathbb{Z}$. If we instead interpret $p-1$ as representing the integer -1, the same expression encodes $0 < -1$, which is false. Thus, reasoning directly in the field erases order relations and other integer properties that many constraint systems implicitly rely upon.

To avoid this, we deliberately interpret modular equations as statements *about integers modulo p*, rather than as equations within the quotient field itself. Each tuple $a = (a_1,\dots,a_n)$ satisfying the constraints modulo p represents an entire congruence class

$$a + p\mathbb{Z}^n = \{(a_1 + k_1 p,\dots,a_n + k_n p) : (k_1,\dots,k_n) \in \mathbb{Z}^n\},$$

all of whose members satisfy the same congruences. Arithmetic circuits, by contrast, operate only on the residue class $(a_1 + p\mathbb{Z},\dots,a_n + p\mathbb{Z}) \in (\mathbb{Z}/p\mathbb{Z})^n$, even though this class is typically represented concretely by the tuple of least nonnegative residues.

Thus, the circuit-level witness is the residue-class representative, while our definition regards any integer lift of that class as a legitimate witness. Adopting this *lifting perspective* allows us to analyze modular constraint satisfaction entirely within $\mathbb{Z}$, so that both satisfying and non-satisfying tuples can be examined using integer notions such as ordering, boundedness, and range, that have no counterpart in the field setting. This choice is crucial for developing a general theory of constraint systems that remains compatible with practical circuit semantics.

1.4. Examples following the definition. To illustrate Definition 1.3, we now revisit the range-check examples introduced earlier, reformulating them in terms of the sets A, D, H, the constraint system C, and the modulus p. We conclude with a new worked example (Example 1.5) for the maximum function.

Example 1.4. (a) As in Example 1.1, assume that $x \in \{-50,\dots,50\}$ and consider the claim $0 \leqslant x \leqslant 15$, which we wish to encode via a constraint system modulo $p = 101$. Here,

$$A = \{-50,\dots,50\} \quad \text{and} \quad D = \{0,\dots,15\}.$$

We need no special assumptions about so-called admissible x, so we let $H = A$:

$$D \cap H = D \cap A = D = \{0,\dots,15\}.$$

The constraint system $C = \{P\}$ consists of the single univariate polynomial

$$P(x) = x(x-1)\cdots(x-15) \in \mathbb{Z}[x].$$

Thus, the witnesses (modulo 101) are precisely the integers in

$$V(C) = \{x \in \mathbb{Z} : x(x-1)\cdots(x-15) \equiv 0 \bmod 101\} = \{0,\dots,15\} + 101\mathbb{Z},$$

and

$$A \cap V(C) = \{0,\dots,15\}.$$

That is, C is both complete and sound (with respect to A, D, H, and p):

$$A \cap V(C) = D \cap H.$$

(b) Alternatively, let $A = I^5$, where $I = \{-50,\dots,50\}$,

$$D = \{0,\dots,15\} \times \{0,1\}^4, \quad \text{and} \quad H = A,$$

so that, again, $D \cap H = D \cap A = D$. Let $C = \{P, Q_0,\dots,Q_3\} \subseteq \mathbb{Z}[x, b_0, b_1, b_2, b_3]$, with

$$P(x, b_0, b_1, b_2, b_3) = x - (b_0 + 2b_1 + 4b_2 + 8b_3) \quad \text{and} \quad Q_i(x, b_0, b_1, b_2, b_3) = b_i(b_i - 1), \quad i \in \{0,1,2,3\}.$$

Then

$$V(C) = (\{0,\dots,15\} + 101\mathbb{Z}) \times (\{0,1\} + 101\mathbb{Z})^4,$$

and

$$A \cap V(C) = \{0,\dots,15\} \times \{0,1\}^4.$$

That is, C is both complete and sound (with respect to A, D, H, and p):

$$A \cap V(C) = D \cap H.$$

In this case, we are really only interested in the projection of $A \cap V(C)$ (equivalently, $D \cap H$) onto its first coordinate: the indeterminates b_0, b_1, b_2, b_3 are ancillary. This means we could also let

$$A = \{-50, \dots, 50\} \times \mathbb{Z}^4 \quad \text{and} \quad D = \{0, \dots, 15\} \times (\{0,1\} + 101\mathbb{Z})^4,$$

while maintaining $H = A$. Indeed, this follows best practice of using fewer and weaker assumptions wherever possible.

(c) Assume that $x \in \{-50, \dots, 50\}$ and consider the claim $x \leqslant 15$, which we wish to encode via a constraint system modulo $p = 101$. Here,

$$A = \{-50, \dots, 50\} \quad \text{and} \quad D = \{x \in \mathbb{Z} : x \leqslant 15\}.$$

We assume that only nonnegative x are admissible (as would be the case, for instance, if x were the result of squaring an integer):

$$H = A \cap \mathbb{N}_0 = \{0, \dots, 50\} \quad \text{and} \quad D \cap H = \{0, \dots, 15\}.$$

If C is exactly as in (a), then $A \cap V(C) = D \cap H$ and C is complete and sound (with respect to A, D, H, and p). We can also proceed as in (b), but with $H = (A \cap \mathbb{N}_0) \times \mathbb{Z}^4$.

(d) Assume that $x \in \{-50, \dots, 50\}$ and consider the claim $x \leqslant 14$, which we wish to encode via a constraint system modulo $p = 101$. Here, $A = \{-50, \dots, 50\}$ as in (c), but

$$D = \{x \in \mathbb{Z} : x \leqslant 14\}.$$

We again assume that only nonnegative x are admissible:

$$H = A \cap \mathbb{N}_0 = \{0, \dots, 50\} \quad \text{and} \quad D \cap H = \{0, \dots, 14\}.$$

If we let C be exactly as in (a) again, then

$$A \cap V(C) = \{0, \dots, 15\} \supsetneq D \cap H.$$

That is, all admissible $x \leqslant 14$ satisfy the constraint system and lie in the ambient interval A, but so does $x = 15$. Thus, C is complete, but not sound (with respect to A, D, H, and p): it is underconstrained.

(e) If we replace the claim $x \leqslant 14$ by $x \leqslant 16$ in (d), the given C will be overconstrained:

$$A \cap V(C) = \{0, \dots, 15\} \subsetneq \{0, \dots, 16\} = D \cap H.$$

That is, no admissible $x > 16$ in the ambient interval is accepted by the constraint system (it is sound), but there are admissible $x \leqslant 16$ (namely, $x = 16$) in the ambient interval that are rejected by the constraint system (it is not complete). ■

Example 1.5. Consider a claim of the form $M = \max\{a, b\}$, where $a, b, M \in \mathbb{Z}$. This is equivalent to the three statements

$$M \in \{a, b\}, \quad M - a \geqslant 0, \quad \text{and} \quad M - b \geqslant 0,$$

which are in turn equivalent (since $\mathbb{Z}$ is an integral domain) to

$$(M - a)(M - b) = 0, \quad M - a \geqslant 0, \quad \text{and} \quad M - b \geqslant 0.$$

To encode this claim, or set of claims, within a constraint system over $\mathbb{Z}/p\mathbb{Z}$, where p is a large prime, we first note that if $(M - a)(M - b) = 0$, then

$$(M - a)(M - b) \equiv 0 \bmod p. \tag{1.5}$$

Conversely, if this congruence holds, then either $M - a \equiv 0 \bmod p$ or $M - b \equiv 0 \bmod p$; that is, either $M - a \in p\mathbb{Z}$ or $M - b \in p\mathbb{Z}$.

To rule out spurious multiples of p, we assume that the differences $M - a$ and $M - b$ lie in a known range. For instance, if

$$-(p - 1) \leqslant M - a, M - b \leqslant p - 1,$$

then (1.5) implies that either $M - a = 0$ or $M - b = 0$. A slightly stronger but more canonical assumption is

$$-\frac{p-1}{2} \leqslant M, a, b \leqslant \frac{p-1}{2}, \tag{1.6}$$

which guarantees that modular equality implies integer equality.

To enforce the inequalities $M - a \geqslant 0$ and $M - b \geqslant 0$, we may choose an integer $\kappa \geqslant 0$ such that $2^\kappa \leqslant p - 1$ and use the binary-decomposition method of Example 1.1 to constrain

$$0 \leqslant M - a \leqslant 2^\kappa - 1 \quad \text{and} \quad 0 \leqslant M - b \leqslant 2^\kappa - 1. \tag{1.7}$$

Suppose

$$-2^{\kappa-1} \leqslant a, b \leqslant 2^{\kappa-1} - 1. \tag{1.8}$$

Then $-2^{\kappa} \leqslant a - b \leqslant 2^{\kappa} - 1$, and so whenever $M \in \{a, b\}$ and $M - a, M - b \geqslant 0$, we automatically have $M - a, M - b \leqslant 2^{\kappa} - 1$.

In summary: the desired relation to be encoded is

$$D = \{(x, y, z) \in \mathbb{Z}^3 : x = \max\{y, z\}\} \times (\{0, 1\} + p\mathbb{Z})^{2\kappa};$$

the ambient domain is

$$A = I^3 \times \mathbb{Z}^{2\kappa}, \quad \text{where} \quad I = \{-(p-1)/2, \ldots, (p-1)/2\};$$

the admissible triples are given by

$$H = \{(x, y, z) \in I^3 : -2^{\kappa-1} \leqslant y, z \leqslant 2^{\kappa-1} - 1\} \times \mathbb{Z}^{2\kappa};$$

and

$$C = \{P, Q, R, S_0, \ldots, S_{\kappa-1}, T_0, \ldots, T_{\kappa-1}\} \subseteq \mathbb{Z}[x, y, z, \alpha_0, \ldots, \alpha_{\kappa-1}, \beta_0, \ldots, \beta_{\kappa-1}],$$

where

$$\begin{aligned}
P(x, y, z, \alpha_0, \ldots, \alpha_{\kappa-1}, \beta_0, \ldots, \beta_{\kappa-1}) &= (x - y)(x - z) \\
Q(x, y, z, \alpha_0, \ldots, \alpha_{\kappa-1}, \beta_0, \ldots, \beta_{\kappa-1}) &= x - y - \left(\alpha_0 + 2\alpha_1 + \cdots + 2^{\kappa-1}\alpha_{\kappa-1}\right) \\
R(x, y, z, \alpha_0, \ldots, \alpha_{\kappa-1}, \beta_0, \ldots, \beta_{\kappa-1}) &= x - z - \left(\beta_0 + 2\beta_1 + \cdots + 2^{\kappa-1}\beta_{\kappa-1}\right) \\
S_i(x, y, z, \alpha_0, \ldots, \alpha_{\kappa-1}, \beta_0, \ldots, \beta_{\kappa-1}) &= \alpha_i(\alpha_i - 1), \quad i \in \{0, \ldots, \kappa - 1\} \\
T_i(x, y, z, \alpha_0, \ldots, \alpha_{\kappa-1}, \beta_0, \ldots, \beta_{\kappa-1}) &= \beta_i(\beta_i - 1), \quad i \in \{0, \ldots, \kappa - 1\}.
\end{aligned}$$

To be clear: H imposes no restriction on x (other than that $x \in I$); κ is fixed but can be any nonnegative integer for which $2^{\kappa} \leqslant p - 1$, which guarantees that $\{-2^{\kappa-1}, \ldots, 2^{\kappa-1} - 1\} \subseteq \{-(p-1)/2, \ldots, (p-1)/2\}$.

The constraint system C is both complete and sound with respect to A, D, H, and p:

$$A \cap V(C) = D \cap H = \{(x, y, z) \in \mathbb{Z}^3 : x = \max\{y, z\} \text{ and } -2^{\kappa-1} \leqslant y, z \leqslant 2^{\kappa-1} - 1\} \times (\{0, 1\} + p\mathbb{Z})^{2\kappa}.$$

The α_i and β_i are ancillary indeterminates: we're only interested in the projection of D onto its first three coordinates (x, y, z). ■

1.5. In-field evaluation and witness preparation. Returning to Example 1.5, we ask how one should compute

$$M = \max\{a, b\}$$

so that the resulting value can be supplied to the constraint system. The same issue arises for the binary decompositions of $M - a$, $M - b$, or of the variable x in Example 1.1(b): all such quantities ultimately enter the circuit through their least nonnegative residues modulo p.

In general, a computation can be represented as a directed acyclic graph (DAG) of algebraic operations whose nodes are intermediate quantities. This abstract graph serves as an intermediate representation (IR) of the computation: each node depends on its predecessors through field operations such as addition, multiplication, or modular reduction. A witness is then an assignment of concrete field values to the nodes of this graph that satisfies all algebraic relations. When the circuit is later verified, the constraints check consistency of these relationships.

A naive but opaque strategy is to evaluate quantities such as M or its decompositions entirely outside this framework, reduce them modulo p, and inject their residues directly. In practice, this process is facilitated by so-called *hints*. Although the use of hints is often unavoidable and can yield numerically correct inputs, it severs the procedural link between those values and the algebraic structure intended to verify them. The resulting witness reflects the computation less transparently, yielding a thinner and less intelligible dependency graph that is less conducive to testing, debugging, and reproducibility.

The guiding principle is therefore to remain, wherever feasible, within the arithmetic of the base field, resorting to external evaluation (hints) only when necessary or prohibitively inconvenient.

Example 1.6 below illustrates this approach for $M = \max\{a, b\}$. In this construction we shift a and b by S, performing the additions in-field, then apply a (non-algebraic) max to the resulting least nonnegative residues, and finally subtract S in-field. This shift is not arbitrary: it encodes our assumption that the admissible inputs lie in a range of the form $[-S, T - S]$. By recentering them in the nonnegative range $[0, T]$ before comparison, the IR can correctly interpret the result within $\mathbb{Z}/p\mathbb{Z}$ using only field operations, without losing information about the intended integer semantics.

In this way, the surrounding arithmetic remains entirely in-field, and the only non-native step is the comparison itself. This contrasts with computing $\max\{a, b\}$ in the integers first and only then reducing modulo p, which would make the entire computation, and the assumptions underlying it, opaque to the IR.

Example 1.6. Let $p \geqslant 2$ be a modulus and κ be a nonnegative integer such that $2^\kappa \leqslant p$. Call integers a and b *admissible* if

$$-2^{\kappa-1} \leqslant a, b \leqslant 2^{\kappa-1} - 1.$$

Equivalently,

$$0 \leqslant a + 2^{\kappa-1}, b + 2^{\kappa-1} \leqslant 2^\kappa - 1.$$

In that case, if $\bar{x}$ denotes the least nonnegative residue of x mod p, then for admissible a and b, we have

$$\overline{a + 2^{\kappa-1}} = a + 2^{\kappa-1} \quad \text{and} \quad \overline{b + 2^{\kappa-1}} = b + 2^{\kappa-1}.$$

Now observe that

$$\max\{a,b\} = \max\{a + 2^{\kappa-1}, b + 2^{\kappa-1}\} - 2^{\kappa-1},$$

and so, if $M = \max\{a,b\}$ for admissible a, b, then

$$\overline{M} = \overline{\max\{a + 2^{\kappa-1}, b + 2^{\kappa-1}\} - 2^{\kappa-1}} = \overline{\max\{\overline{a + 2^{\kappa-1}}, \overline{b + 2^{\kappa-1}}\} - 2^{\kappa-1}}.$$

In this way, we may generate a witness $\overline{M}$ to $M = \max\{a,b\}$, directly from $\bar{a}$ and $\bar{b}$, using only addition and subtraction in $\mathbb{Z}/p\mathbb{Z}$, provided a method is available to compute the maximum of two least nonnegative residues. ■

The translation trick in Example 1.6 effectively "recenters" admissible integers in the nonnegative range, allowing their maximum to be computed correctly in the field setting. A similar idea can be used to compute least nonnegative residue representations of the quotient and remainder in Euclidean division.

Example 1.7. Assume p is a large prime, and fix a positive integer α. Suppose there exists a nonnegative integer T such that $\alpha T \leqslant p - 1$. Define an integer c to be *admissible* if $c + \alpha S \in [0, \alpha T]$ for some integer S. In particular, if c is admissible, then

$$\overline{c + \alpha S} = c + \alpha S.$$

Let $q^\sharp$ and r be the unique integers such that

$$\overline{c + \alpha S} = \alpha q^\sharp + r \quad \text{with} \quad 0 \leqslant r \leqslant \alpha - 1.$$

(Since α and $\overline{c + \alpha S}$ lie in $\{0, \ldots, p-1\}$, so do $q^\sharp$ and r.) If c is admissible, then the integers $q = q^\sharp - S$ and r satisfy

$$c = \alpha q + r \quad \text{with} \quad 0 \leqslant r \leqslant \alpha - 1.$$

Hence we may generate a witness pair

$$\bar{q} = \overline{q^\sharp - S} \quad \text{and} \quad \bar{r} = r,$$

representing the least nonnegative residues of the quotient and remainder in the division of c by α, using only arithmetic in $\mathbb{Z}/p\mathbb{Z}$. This requires a method for computing the quotient and remainder from a least nonnegative residue $\overline{c + \alpha S}$ and the divisor α, such as Expander Compiler Collection's `api.unconstrained_int_div` and `api.unconstrained_mod`. ■

1.6. Toward primitive constraint systems. At this stage, we have only discussed how to express seemingly simple integer relations, such as $\max\{a,b\}$, Euclidean division, or a small range check, within the formal framework of modular arithmetic. Even these basic examples reveal how subtle the transition from *integer* reasoning to *field-level* reasoning can be, and how easily completeness or soundness can be lost if admissibility conditions are not made explicit.

This is precisely why real-world constraint-system codebases can appear bewildering at first glance: what looks like a forest of low-level polynomials, auxiliary variables, and shifting constants is in fact a carefully constructed bridge between discrete arithmetic semantics and algebraic syntax over $\mathbb{Z}/p\mathbb{Z}$. When viewed through this lens, every seemingly *ad hoc* constraint is part of a larger pattern ensuring that the intended integer properties survive the modular translation.

We are now ready to build a library of *primitive constraint systems* (range checks, equality and inequality tests, etc.), which will serve as the building blocks for higher-level circuit design.

Later, when we extend this framework to randomized verification (e.g. Freivalds' algorithm), we will generalize the notions of completeness and soundness to their probabilistic counterparts (see Section 4), reflecting that real proof systems often trade exactness for efficiency.

2. NOTATION

Throughout, unless stated explicitly to the contrary, we employ the following notation and conventions. This list isn't exhaustive: much of our notation is standard, e.g. $\mathbb{Z}$ denotes the set of integers, and other notation may be introduced *in situ*.

- We include 0 among the natural numbers. To avoid ambiguity, we use the notation
$$\mathbb{N}_0 := \{0, 1, \ldots\},$$
and tend to use the term *nonnegative integers*.
- If $m \in \mathbb{Z}$ and $A, B \subseteq \mathbb{Z}$, then $mA := \{ma : a \in A\}$ and $A + B := \{a + b : a \in A,\, b \in B\}$. For instance,
$$\{0, \ldots, 15\} + 101\mathbb{Z} = \{a + 101k : a \in \{0, \ldots, 15\},\, k \in \mathbb{Z}\}.$$
We may also write $a + B$ instead of $\{a\} + B$ in the case where A is a singleton set. The same notation applies if $A, B \subseteq \mathbb{Z}^n$, with scalar multiplication and addition interpreted coordinate-wise as is standard. For instance, if $a = (a_1, \ldots, a_n) \in \mathbb{Z}^n$, then
$$a + 101\mathbb{Z}^n = \{(a_1 + 101k_1, \ldots, a_n + 101k_n) : (k_1, \ldots, k_n) \in \mathbb{Z}^n\}.$$
- p denotes an integer modulus with $p \geqslant 2$. Often, p denotes an odd prime, but there are certain results where primality of p is not required.
- $\mathbb{Z}/p\mathbb{Z}$ denotes the ring of integers modulo p, whose elements are the equivalence (or residue) classes
$$0 + p\mathbb{Z}, \ldots, p - 1 + p\mathbb{Z},$$
also denoted 0 mod $p, \ldots, p - 1$ mod p. We may also use the notation
$$[a]_p := a + p\mathbb{Z},$$
possibly omitting the subscript p if it is clear from context.

  If A is a set, multiset, sequence, or matrix of integers, then $A + p\mathbb{Z}$, A mod p denotes the set, multiset, sequence, or matrix of integers obtained by replacing each integer in A by its least nonnegative residue modulo p. Similarly, if P is a polynomial over $\mathbb{Z}$, P mod p denotes the polynomial over $\mathbb{Z}/p\mathbb{Z}$ defined by replacing each coefficient of P with the residue class it represents.

  Given any integer a in an equivalence class, we say that *a represents* that class, or that the class is represented by a. To be clear: elements of the ring $\mathbb{Z}/p\mathbb{Z}$ are equivalence classes, not integers; integers represent elements of $\mathbb{Z}/p\mathbb{Z}$, but are not elements of this quotient ring. We strive not to identify integers with the residue classes they represent.
- Given an integer $p \geqslant 2$, the *least nonnegative residues* modulo p are the integers $0, 1, \ldots, p - 1$. If p is odd, the *balanced residues* modulo p are the integers $-(p-1)/2, \ldots, (p-1)/2$. Both are examples of a *complete set of residues* modulo p, i.e. they represent distinct
- $\bar{z}$ denotes the least nonnegative residue of an integer z modulo p, when the modulus is clear from context. That is, $\bar{z}$ is the unique element in $\{0, \ldots, p - 1\} \cap (z + p\mathbb{Z})$. If A is a set, multiset, sequence, or matrix of integers, then $\bar{A}$ denotes the set, sequence, or matrix obtained by replacing each integer in A by its least nonnegative residue modulo p. Similarly, if P is a polynomial over $\mathbb{Z}$, $\bar{P}$ denotes the polynomial over $\mathbb{Z}$ obtained by replacing each coefficient of p by the residue class modulo P that it represents.
- Probability functions usually denoted by Pr. Thus, if $(\Omega, \mathscr{F}, \Pr)$ is a probability space with sample space Ω, event space $\mathscr{F}$, and probability function Pr, the probability of the event $E \in \mathscr{F}$ is denoted $\Pr[E]$.
- Given a random variable $X : \Omega \to S$ and $A \subseteq S$, $\Pr[\{\omega \in \Omega : X(\omega) \in A\}]$ is usually abbreviated to $\Pr[X \in A]$.
- We will only encounter discrete probability spaces, i.e. ones for which Ω is finite or countably infinite, $\mathscr{F} = 2^\Omega$, and Pr is determined by $\Pr[\omega] = \Pr[\{\omega\}]$ for each $\omega \in \Omega$.
- We avoid the double-brace notation for multisets, as it can be ambiguous (is $\{\{1, 1\}\}$ a multiset of the singleton set whose element is the singleton set $\{1\}$?). We prefer to define a multiset m over a set A as a function
$$m : A \to \mathbb{N}_0,$$
where $m(a)$ denotes the *multiplicity* of element a. The underlying set is the support of m:
$$\operatorname{supp}(m) := \{a \in A : m(A) > 0\}.$$
The *cardinality* or *total size* of the multiset is
$$\#m := \sum_{a \in A} m(a).$$
For our purposes, the set A will always be finite or countably infinite: if $\operatorname{supp}(m)$ is infinite, then so is $\#m$.

3. Elementary constraint systems

We begin with the simplest cases, where the intended relation expresses an exact equality. These examples illustrate how integer equalities can be encoded by polynomial congruences modulo p, provided we lift solutions to an interval of length p.

Proposition 3.1 (Constant equality)**.** *Let $p \geqslant 2$ be an integer modulus, $I = \{h, h+1, \dots, h+p-1\}$ for some integer h, and $A = H = I^n$ ($n \geqslant 1$). Fix $c = (c_1, \dots, c_n) \in A$, and set $D = \{c\}$. Let*

$$C = \{P_1, \dots, P_n\} \subseteq \mathbb{Z}[x_1, \dots, x_n], \quad P_i(x_1, \dots, x_n) := x_i - c_i \quad (i = 1, \dots, n).$$

Then

$$A \cap V_p(C) = D \cap H.$$

Proof. Since $A = H$, the set equality $A \cap V_p(C) = D \cap H$ is implied by the set equality $A \cap V_p(C) = D$, which we now establish. By definition,

$$V_p(C) = \{(a_1, \dots, a_n) \in \mathbb{Z}^n : a_i - c_i \equiv 0 \bmod p \text{ for all } i\} = \{(c_1 + k_1 p, \dots, c_n + k_n p) : (k_1, \dots, k_n) \in \mathbb{Z}^n\}.$$

If $a \in D$, then $a = c$, which clearly lies in both A (by hypothesis) and $V_p(C)$ (by taking $k_i = 0$). Hence $D \subseteq A \cap V_p(C)$.

Conversely, suppose $a \in A \cap V_p(C)$. Then $a_i = c_i + k_i p$ for some $k_i \in \mathbb{Z}$, and $a_i, c_i \in I = \{h, \dots, h+p-1\}$. Thus

$$h \leqslant c_i + k_i p \leqslant h + p - 1 \quad \text{and} \quad h \leqslant c_i \leqslant h + p - 1,$$

implying $-(p-1) \leqslant k_i p \leqslant p - 1$. Hence $k_i = 0$, and so $a = c$. Therefore, $A \cap V_p(C) \subseteq D$. □

Proposition 3.2 (Variable equality)**.** *Let $p \geqslant 2$ be an integer modulus, $I = \{h, h+1, \dots, h+p-1\}$ for some integer h, and $A = H = I^{2n}$ ($n \geqslant 1$). Let*

$$D = \{(x, y) \in \mathbb{Z}^n \times \mathbb{Z}^n : x = y\},$$

and

$$C = \{P_1, \dots, P_n\} \subseteq \mathbb{Z}[x_1, y_1, \dots, x_n, y_n], \quad P_i(x_1, \dots, x_n, y_1, \dots, y_n) := x_i - y_i.$$

Then

$$A \cap V_p(C) = D \cap H.$$

Proof. Since $A = H$, the set equality $A \cap V_p(C) = D \cap H$ is implied by the set equality $A \cap V_p(C) = D$, which we now establish. We have

$$V_p(C) = \{(a_1, \dots, a_n, b_1, \dots, b_n) \in \mathbb{Z}^{2n} : a_i - b_i \in p\mathbb{Z} \text{ for all } i\}.$$

Write a and b for $(a_1, \dots, a_n)$ and $(b_1, \dots, b_n)$, respectively. If $(a, b) \in D \cap H$, then $a = b$, and thus $a_i - b_i = 0 \in p\mathbb{Z}$ for each i, so $(a, b) \in V_p(C)$. Hence $D \cap H \subseteq A \cap V_p(C)$.

Conversely, if $(a, b) \in A \cap V_p(C)$, then $a_i, b_i \in I = \{h, \dots, h+p-1\}$, so

$$-(p-1) \leqslant a_i - b_i \leqslant p - 1.$$

Since $a_i - b_i \in p\mathbb{Z}$, we must have $a_i - b_i = 0$ for each i, and thus $a = b$. Therefore, $A \cap V_p(C) \subseteq D \cap H$. □

The following setup generalizes constant equality (in the one-dimensional case).

Proposition 3.3 (Set membership)**.** *Let p be a prime modulus, and let $A = H = \{h, h+1, \dots, h+p-1\}$ for some integer h. Let $D \subseteq A$, and define*

$$C = \{P\} \subseteq \mathbb{Z}[x], \quad P(x) = \prod_{d \in D} (x - d).$$

Then

$$A \cap V_p(C) = D \cap H.$$

Proof. Since $A = H$, the set equality $A \cap V_p(C) = D \cap H$ is implied by the set equality $A \cap V_p(C) = D$, which we now establish. Since $\mathbb{Z}/p\mathbb{Z}$ is an integral domain,

$$V_p(C) = \{a \in \mathbb{Z} : \textstyle\prod_{d \in D}(a - d) \equiv 0 \bmod p\} = D + p\mathbb{Z}.$$

($D = \varnothing$ gives $P \equiv 1$ and $V_p(C) = \varnothing$, so the claim is trivial in that case.)

If $a \in D$, then, since $D \subseteq A$, $a \in A$. Also, $P(a) = 0$ and so $a \in V_p(C)$ as well. Hence $D \subseteq A \cap V_p(C)$.

Conversely, let $a \in A \cap V_p(C)$. Then $a \equiv d \bmod p$ for some $d \in D$. Because $a, d \in A = \{h, \dots, h+p-1\}$, we have

$$a - d \in \{-(p-1), \dots, p-1\},$$

and the only multiple of p in this interval is 0. Therefore, $a = d$, and hence $a \in D$. Hence $A \cap V_p(C) \subseteq D$. □

Proposition 3.4 (Addition)**.**

Proof. □

Proposition 3.5 (Multiplication)**.**

Proof. □

Proposition 3.6 (Generalized matrix multiplication)**.**

Proof. □

Proposition 3.7 (Division)**.**

Proof. □

4. Probabilistic completeness and soundness

4.1. Motivation. Consider Proposition 3.3, in which we showed that membership of a subset D, of size $\#D = n$, of an interval

$$A := \{h, h+1, \ldots, h+p-1\}$$

of length p (prime), can be encoded via the degree-n polynomial

$$P(x) = \prod_{d \in D} (x - d).$$

This deterministic encoding is exact but costly: when n is large, it induces an equally large arithmetic circuit and thus an inefficient proof system.

To mitigate this, we can shift to a *probabilistic* form of verification. In informal terms, the verifier no longer fixes a single constraint system in advance, but instead defines a *family* of systems parameterized by a random value. The prover first commits to their purported witness a (and, if required, auxiliary information derived from it), and only then does the verifier sample a random challenge from this family and request consistency with it. (To say that the prover "commits" to a witness means that they supply a in a cryptographically binding way, so that they cannot later modify their submission to adapt to the verifier's random challenge.)

The family is constructed so that any *honest prover* (one providing an $a \in D$), will satisfy every possible member of the family, ensuring correctness (completeness). Conversely, a *dishonest prover* (one submitting $a \notin D$) can succeed only if the verifier's random choice happens to produce a degenerate instance, which occurs with negligible probability.

This idea leads naturally to the notion of *probabilistic soundness*, where the constraint system itself depends on randomness, and soundness is guaranteed not absolutely but with overwhelming probability over the verifier's random choice. In the following example we illustrate this concept concretely by replacing the deterministic constraint $P(a) \equiv 0 \bmod p$ with a randomized variant involving an auxiliary polynomial Q with a random field element u.

Although the resulting construction may at first appear more elaborate, introducing auxiliary polynomials, random parameters, and additional constraints, in fact the verifier's random constraint involves only a *linear* polynomial in the challenge variable.

Choose an integer u uniformly at random from $\{0, \ldots, p-1\}$. Set

$$Q_a(x) := \prod_{\substack{d \in D \\ d \neq a}} (x - d) \quad \text{and} \quad R_{a,u}(x) := P(u) - (u - x)Q_a(u).$$

Then $R_{a,u}$ is a linear polynomial in $\mathbb{Z}[x]$, and if $a \in D$, then

$$R_{a,u}(a) = P(u) - (u - a)Q_a(u) = P(u) - P(u) = 0.$$

On the other hand, if $a \notin D$, then $Q_a(u) = P(u)$ and so

$$R_{a,u}(a) = P(u) - (u - a)Q_a(u) = (1 - u + a)P(u),$$

which vanishes modulo p if and only if u is congruent to an integer in $D \cup \{a+1\}$ modulo p. This happens with probability at most $(n+1)/p$, which is negligible if n is small enough relative to p.

To formulate this in terms of Definition 1.3, we begin by viewing the evaluations univariate polynomials Q_a and $R_{a,u}$ as evaluations of multivariate polynomials, as follows. As $Q_a(x)$ is a product of at most $n-1$ linear factors if $a \in D$, and at most n linear factors if $a \notin D$, it has degree at most n in the indeterminate x, and we may write

$$Q_a(u) = q_{a,n}u^n + q_{a,n-1}u^{n-1} + \cdots + q_{a,0} = Q_u^\flat(q_{a,0}, \ldots, q_{a,n}),$$

say, where

$$Q_u^\flat(y_0, \ldots, y_n) := u^n y_n + u^{n-1} y_{n-1} + \cdots + y_0 \in \mathbb{Z}[y_0, \ldots, y_n].$$

Thus,

$$R_{a,u}(a) = P(u) - (u-a)Q_a(u) = P(u) - (u-a)Q_u^\flat(q_{a,0}, \ldots, q_{a,n}) = R_u^\flat(a, q_{a,0}, \ldots, q_{a,n}),$$

where

$$R_u^\flat(x, y_0, \ldots, y_n) := P(u) - (u-x)Q_u^\flat(y_0, \ldots, y_n) \in \mathbb{Z}[x, y_0, \ldots, y_n].$$

Finally, given $a \in \mathbb{Z}$ and $q = (q_0, \ldots, q_n) \in \mathbb{Z}^{n+1}$, define a univariate polynomial $S_{a,q}$, of degree at most $n+1$, by

$$S_{a,q}(w) := R_w^\flat(a, q_0, \ldots, q_n) \in \mathbb{Z}[w].$$

(We will continue to use q and q_a as shorthand for $(q_0, \ldots, q_n)$ and $(q_{a,0}, \ldots, q_{a,n})$, respectively, in what follows.)

Consider the mod p vanishing set of the random constraint system $C_u = \{R_u^\flat\}$. By definition,

$$\begin{aligned} V_p(C_u) &= \{(a,q) \in \mathbb{Z}^{n+2} : R_u^\flat(a,q) \equiv 0 \bmod p\} \\ &= \{(a,q) \in \mathbb{Z}^{n+2} : S_{a,q}(u) \equiv 0 \bmod p\}. \end{aligned}$$

To reiterate, (a,q) is *not* random, and $\mathbb{Z}^{n+2}$ is *not* the sample space. The tuple (a,q) is predetermined and committed to by a prover. The random parameter is u, which is selected from the sample space $\{0, \ldots, p-1\}$. That's why we switch from the definition of $V_p(C_u)$ in terms of roots (a,q) of $R_u^\flat$, to the equivalent formulation in terms of roots u of $S_{a,q}$. We're interested in the event that u mod p belongs to the set of roots of $S_{a,q}$ mod p. We *could* indicate the probability of this event by writing $\Pr[(a,q) \in V_p(C_u)]$, but this is an abuse of notation and could mislead. We want

$$\Pr\left[\{u \in \{0, \ldots, p-1\} : S_{a,q}(u) \equiv 0 \bmod p\}\right].$$

If $a \in D$, there exists $q_a \in \mathbb{Z}^{n+1}$ such that

$$S_{a,q_a}(u) = 0$$

for all $u \in \mathbb{Z}$, namely the coefficients q_a of Q_a defined earlier (the quotient in the division of $P(x)$ by $x-a$). That is, if $a \in D$, then

$$\Pr[\{u \in \{0, \ldots, p-1\} : S_{a,q_a}(u) \equiv 0 \bmod p\}] = \Pr[\{0, \ldots, p-1\}] = 1.$$

The same holds for all $a' \in D + p\mathbb{Z}$, for $D \subseteq A = \{h, h+1, \ldots, h+p-1\}$, and we may choose the unique $a \in D$ such that $a \equiv a' \bmod p$, then compute q_a. Thus, $S_{a',q_a}(u) \equiv S_{a,q_a}(u) \equiv 0 \bmod p$.

If there exists a $q' \in \mathbb{Z}^{n+1}$ for which $S_{a',q'}(u) \equiv 0 \bmod p$ for all $u \in \{0, \ldots, p-1\}$, then there are infinitely many such q', and the prover may take their pick. The only requirement for completeness is that, for any $a \in D$, there exists a q for which the desired relation holds. We are not interested in describing all such q, because this is a tuple of ancillary parameters: we're ultimately interested in whether $a \in D$.

Suppose, on the other hand, that $a \notin D + p\mathbb{Z}$, and let $q \in \mathbb{Z}^{n+1}$ be arbitrary. Unless $S_{a,q}$ is identically zero mod p, there are at most $\deg S_{a,q} \leqslant n+1$ distinct congruence classes u mod p for which $S_{a,q} \equiv 0 \bmod p$ (see Proposition 4.2). To see that $S_{a,q}$ is *not* identically zero mod p, note that

$$S_{a,q}(a) = R_a^\flat(a,q) = P(a) - (a-a)Q_a^\flat(q) = P(a),$$

and therefore, since $a \notin D + p\mathbb{Z}$,

$$S_{a,q}(a) \equiv P(a) \not\equiv 0 \bmod p.$$

Hence

$$\Pr[\{u \in \{0, \ldots, p-1\} : S_{a,q}(u) \equiv 0 \bmod p\}] \leqslant \frac{n+1}{p}.$$

In summary, an honest prover is guaranteed to have their input accepted by the randomly selected constraint system C_u, even without knowing u, provided $a \in D + p\mathbb{Z}$ and q_a is given by the quotient polynomial Q_a. A dishonest prover that supplies any (a,q) with $a \notin D + p\mathbb{Z}$ will only have their input accepted by C_u if u mod p happens to be a root of the nonzero polynomial $S_{a,q}$ mod p, which is extremely unlikely if p is extremely large relative to n. The process could be repeated with multiple, independently chosen u, in order to make the probability of acceptance of a false witness as small as desired.

One might well ask whether this probabilistic procedure actually reduces the size of the arithmetic circuit. In the deterministic encoding, verifying membership of $a \in D$ requires computing the degree-n product

$$P(a) = \prod_{d \in D} (a - d)$$

within the circuit, introducing $n-1$ multiplication gates. By contrast, in the probabilistic construction, the circuit never multiplies these factors. After the prover has fixed (a, q), the verifier samples a random field element u and computes

$$S_{a,q}(u) = P(u) - (u - a)Q_u^\flat(q)$$

outside the circuit. This evaluation involves only a few scalar multiplications and additions by the verifier, while $P(u)$, a single field element, can be treated as a known public input. The prover merely provides coefficients q claimed to satisfy the relation, and the circuit enforces a single constraint

$$S_{a,q}(u) \equiv 0 \bmod p.$$

Although $S_{a,q}$ is a polynomial of degree at most $n+1$, the in-circuit computation verifying this equality is constant in size, independent of n. The expensive product defining $P(u)$ is performed off-circuit by the verifier, while soundness relies on the fact that an incorrect q satisfies this linear check with probability at most $(n+1)/p$. This probabilistic relaxation underlies modern lookup arguments such as LogUp, which replace high-degree, prover-side constraints with compact, verifier-evaluated checks while preserving negligible soundness error.

4.2. Formal probabilistic definitions. Let us now express all of this in relation to Definition 1.3. We are only interested in the projection of $V_p(C_u)$ onto its first coordinate a, and whether or not this lies in D. The q_i are therefore ancillary parameters, and we may take our ambient domain to be

$$A^\flat := A \times \mathbb{Z}^{n+1}.$$

We have no special admissibility assumptions here, so $H = A$. As with A, we lift D and H to

$$D^\flat := D \times \mathbb{Z}^{n+1} \quad \text{and} \quad H^\flat := H \times \mathbb{Z}^{n+1}.$$

We have shown that for all $a \in D \cap H$, there exists $q_a \in \mathbb{Z}^{n+1}$ such that

$$\Pr[\{u \in \{0, \dots, p-1\} : S_{a,q_a}(u) \equiv 0 \bmod p\}] = 1.$$

i.e., $(a, q_a) \in A^\flat \cap V_p(C_u)$ with probability 1. (The extra condition that $(a, q_a) \in A^\flat$ follows from our assumption that $D \subseteq A$.) This is completeness. Furthermore, for all $(a, q) \in A^\flat$, if $a \in A \backslash (D \cap H)$, then, since $D \subseteq A$ and A is an interval of length p, we have $a \notin D + p\mathbb{Z}$. Thus,

$$\Pr[\{u \in \{0, \dots, p-1\} : S_{a,q}(u) \equiv 0 \bmod p\}] \leqslant \frac{n+1}{p},$$

i.e. $(a, q) \in A^\flat \cap V_p(C_u)$ only with negligible probability (assuming n is much smaller than p). This is soundness.

The parameter u can be chosen according to any distribution on the sample space $\{0, \dots, p-1\}$, not just the uniform distribution as in the above example. In the degenerate case where the verifier's randomness is fixed, say $u = u_0$ with probability 1, the random family $\{C_u\}$ collapses to a single constraint system C_{u_0}. The probabilistic completeness and soundness conditions then reduce to the deterministic inclusions of Definition 1.3, recovering the classical setting. That is, completeness becomes the statement that, for all $a \in D \cap H$, there exists q_a such that $(a, q_a) \in A^\flat \cap V_p(C_{u_0})$. Soundness becomes the statement that, for all $(a, q) \in A^\flat$ with $a \notin D \cap H$, we have $(a, q) \notin V_p(C_{u_0})$. Thus the probabilistic inclusions reduce to the deterministic inclusions

$$D^\flat \cap H^\flat \subseteq A^\flat \cap V_p(C_{u_0}) \subseteq D^\flat \cap H^\flat.$$

Definition 4.1. Fix $\varepsilon, \delta \in [0, 1]$. Let p be an integer modulus, and let $\{C_0, \dots, C_{p-1}\}$ be a family of constraint systems, each in $m+n$ indeterminates. Let $A, D \subseteq \mathbb{Z}^m$ and $H \subseteq A$. Let ν be a probability distribution on $\{0, \dots, p-1\}$, and write $u \sim \nu$ for sampling a random index according to ν.

The family $(\{C_u\}, \nu)$ is called δ-*complete* with respect to (A, D, H, p, ν) if, for every $a \in D \cap H$, there exists $b \in \mathbb{Z}^n$ such that

$$\Pr_{u \sim \nu}[(a, b) \in (A \times \mathbb{Z}^n) \cap V_p(C_u)] \geqslant 1 - \delta.$$

It is called ε-*sound* with respect to (A, D, H, p, ν) if, for every $a \in A \backslash (D \cap H)$ and every $b \in \mathbb{Z}^n$,

$$\Pr_{u \sim \nu}[(a, b) \in (A \times \mathbb{Z}^n) \cap V_p(C_u)] \leqslant \varepsilon.$$

We may simply say that the family is *complete* or *sound* when δ or ε are clear from context. If $\delta = 0$, we say it is *perfectly complete*; if $\varepsilon = 0$, we say it is *perfectly sound*. If $(\{C_u\}, \nu)$ is δ-complete but not ε-sound, it is said to be *underconstrained*; if it is ε-sound but not δ-complete, it is *overconstrained*. When the parameters A, D, H, p, and ν are clear from context, we may omit them in such phrases. ■

4.3. Useful results. We collect several foundational results that will be used throughout the sequel. Each expresses a common principle: a random evaluation of an algebraic object over a field almost never hides a genuine inconsistency. Equivalently, two distinct algebraic structures (such as polynomials or vectors) are extremely unlikely to coincide at a randomly chosen point.

The first result, the *Schwartz-Zippel lemma*, bounds the probability that a nonzero polynomial vanishes at a randomly selected evaluation point. The second, a linear-algebraic analogue often used to justify Freivalds' algorithm for verifying matrix multiplication, bounds the probability that a random vector lies in a low-dimensional subspace. Together, these theorems formalize the intuition that random sampling provides an efficient and reliable tool for detecting algebraic disagreement, an idea that underlies the soundness of many probabilistic proof systems.

Historically, the polynomial bound was first proved by DeMillo and Lipton [1] and rediscovered independently by Schwartz [5] and Zippel [6], with a finite-field precursor due to Ore in 1922 [4]. We begin with the univariate case, which will also underlie the probabilistic soundness argument in Subsection 4.1.

Proposition 4.2. *Let $\mathbb{F}$ be a field and $g \in \mathbb{F}[X]$ a nonzero univariate polynomial over $\mathbb{F}$. If g has degree d, then g has at most d roots in $\mathbb{F}$.*

Proof. Suppose $r_0 \in \mathbb{F}$ is a root of g, $g(r_0) = 0$. By the division algorithm, there exists a polynomial $q(X) \in \mathbb{F}[X]$, of degree $d-1$, such that $g(X) = (X - r_0)q(X)$, where $q(X) \in \mathbb{F}[X]$. If r_0 is a repeated root, then we can factor g further as $g(X) = (X - r_0)^{m_0} q_0(X)$, where $m_0 \geqslant 1$ is maximal (i.e., $q_0(r_0) \neq 0$). At this point, $q_0(X)$ is a polynomial of degree $d - m_0$. Now, if there is another root $r_1 \neq r_0$, then $g(r_1) = 0$ implies $q_0(r_1) = 0$ because $(r_1 - r_0)^{m_0} \neq 0$ (fields have no zero divisors). Therefore, $q_0(X)$ must also be divisible by $(X - r_1)$, allowing us to write $g(X) = (X - r_0)^{m_0}(X - r_1)q_1(X)$ for some polynomial $q_1(X)$ of degree $d - m_0 - 1$. We can repeat this process, factoring out distinct roots until we have exhausted the degree d. Since each distinct root reduces the degree of the remaining polynomial by at least 1, the number of roots (counted with multiplicity) cannot exceed d. Thus, g can have at most d roots in $\mathbb{F}$. This argument also generalizes to any integral domain R, where the absence of zero divisors ensures that a similar factorization process holds. □

Proposition 4.3 (Schwartz–Zippel lemma)**.** *Let g be a v-variate ($v \geqslant 1$) nonzero polynomial of total degree d over a finite field $\mathbb{F}$. Then*

$$\#\{r \in \mathbb{F}^v : g(r) = 0\} \leqslant (\#\mathbb{F})^{v-1} d.$$

Equivalently, if $r_0, \ldots, r_{v-1}$ are selected independently and uniformly at random from $\mathbb{F}$, then

$$\Pr[g(r_0, \ldots, r_{v-1}) = 0] \leqslant \frac{d}{\#\mathbb{F}}. \tag{4.1}$$

Proof. The proof is by induction on v. The base case $v = 1$ is Proposition 4.2. Let $v \geqslant 2$ and assume the statement holds for all nonzero polynomials in u indeterminates, $1 \leqslant u < v$.

We can write

$$g(X_0, \ldots, x_{v-1}) = \sum_{e \in \mathbb{N}^v} a_e x_0^{e_0} \cdots x_{v-1}^{e_{v-1}} = \sum_{k=0}^{d} x_{v-1}^k \, h_k(x_0, \ldots, x_{v-2}),$$

where each $h_k \in \mathbb{F}[x_0, \ldots, x_{v-2}]$, and at least one h_k is nonzero. Let $j = \max\{k : h_k \not\equiv 0\}$. We partition the set of roots of g according to whether or not $(r_0, \ldots, r_{v-2})$ is a root of h_j:

$$\{r \in \mathbb{F}^v : g(r) = 0\} \subseteq \{r : h_j(r_0, \ldots, r_{v-2}) = 0\} \cup \{r : h_j(r_0, \ldots, r_{v-2}) \neq 0 \text{ and } g(r) = 0\}.$$

Since $\deg(h_j) \leqslant d - j$, the inductive hypothesis gives

$$\#\{(r_0, \ldots, r_{v-2}) : h_j(r_0, \ldots, r_{v-2}) = 0\} \leqslant (\#\mathbb{F})^{v-2}(d - j).$$

Hence

$$\#\{(r_0, \ldots, r_{v-1}) : h_j(r_0, \ldots, r_{v-2}) = 0\} \leqslant (\#\mathbb{F})^{v-1}(d - j).$$

If $h_j(r_0, \ldots, r_{v-2}) \neq 0$, then $g(r_0, \ldots, r_{v-2}, x_{v-1})$ is a nonzero univariate polynomial of degree j. For each of the at most $(\#\mathbb{F})^{v-1}$ such $(r_0, \ldots, r_{v-2})$, there are at most j corresponding r_{v-1} satisfying $g(r) = 0$. Thus

$$\#\{r : h_j(r_0, \ldots, r_{v-2}) \neq 0 \text{ and } g(r) = 0\} \leqslant (\#\mathbb{F})^{v-1} j.$$

Summing both cases gives the desired bound

$$\#\{r : g(r) = 0\} \leqslant (\#\mathbb{F})^{v-1}(d - j + j) = (\#\mathbb{F})^{v-1} d.$$

□

Remark 4.4. (i) The argument extends beyond finite fields: the same bound holds for any integral domain R, provided the evaluation points are drawn from a finite subset $S \subseteq R$, with $\#S$ replacing $\#\mathbb{F}$ in (4.1). This more general statement, proved independently by DeMillo and Lipton, Schwartz, and Zippel, is not required for our purposes here.

(ii) The base case in Proposition 4.2 is sometimes confused with the *fundamental theorem of algebra*, which asserts that a nonzero polynomial of degree d over $\mathbb{C}$ has exactly d complex roots (counted with multiplicity). In contrast, Proposition 4.2 applies to any field and asserts only that a degree-d polynomial can have at most d roots. ■

The next result is a linear-algebraic analogue of the Schwartz-Zippel lemma. It quantifies how rarely a random vector over a finite field falls into a proper subspace, an observation that underpins Freivalds' algorithm for verifying matrix multiplication. Like the polynomial case, it expresses that random sampling detects algebraic dependence with high probability.

Proposition 4.5. *Let V be an n-dimensional vector space over a finite field $\mathbb{F}$ of order q, and let $W \subseteq V$ be a subspace of dimension k. If $x \in V$ is chosen uniformly at random, then*

$$\Pr[x \in W] = \frac{1}{q^{n-k}}.$$

In particular, if W is the null space of a nonzero matrix $D \in \mathbb{F}^{\ell \times n}$, then

$$\Pr[x \in W] \leqslant \frac{1}{q}.$$

Proof. For the first claim, let $\{w_1, \ldots, w_k\}$ be a basis of W. Each element of W can be written uniquely as $a_1 w_1 + \cdots + a_k w_k$ with $a_i \in \mathbb{F}$, so $\#W = q^k$. Since V has dimension n, we have $\#V = q^n$, and therefore

$$\Pr[x \in W] = \frac{\#W}{\#V} = \frac{q^k}{q^n} = \frac{1}{q^{n-k}}.$$

For the second claim, suppose W is the null space of M, for some nonzero $M \in \mathbb{F}^{\ell \times n}$. If $W = V$, then $Me_j = 0$ for each standard basis vector $e_j \in \mathbb{F}^n$, where e_j has a 1 in the j-th coordinate and 0 elsewhere. But Me_j is the j-th column of M, implying that all columns of M are zero and hence $M = 0$, a contradiction. Thus W is a proper subspace of V, so $\dim W = k \leqslant n-1$, and by the first part,

$$\Pr[x \in W] = \frac{1}{q^{n-k}} \leqslant \frac{1}{q}.$$

□