Arithmetization of Σ_1^1 relations in Halo 2

Morgan Thomas

Orbis Labs
team@orbislabs.com

July 29, 2022

Abstract

Orbis Labs presents a method for compiling ("arithmetizing") relations, expressed as Σ_1^1 formulas in the language of rings, into Halo 2 arithmetic circuits. This method offers the possibility of creating arithmetic circuits without laborious and error-prone manual circuit design and implementation, by instead expressing the relation to be arithmetized in a concise mathematical notation and generating the circuit based on that expression.

Contents

1 Introduction **2**

2 Halo 2 circuits **3**

3 Σ_1^1 formulas over the language of rings **6**

4 Arithmetizing Σ_1^1 formulas over the language of rings **12**
4.1 Circuit columns 18
4.2 Circuit constraints 25
4.3 The circuit itself 32
4.4 Variations on this construction 32

5 Proof sketch of correctness of the arithmetization **32**
5.1 Categories of circuits 33
5.1.1 Initial objects ($\perp$) 35
5.1.2 Products ($\times$) 36
5.1.3 Coproducts ($\oplus$) 37
5.1.4 Negations ($\perp^{\mathcal{C}}$) 38
5.2 Functorial arithmetization 39
5.3 Byte decompositions 48
5.3.1 Coproducts ($\oplus$) 49
5.3.2 Functorial arithmetization 50
5.4 Quantified formulas 53

6 The other direction **54**

1 Introduction

Halo 2 [1, 2, 3] is a mathematical theory and Rust library for creating and verifying zk-SNARKs. A zk-SNARK is a succint message which probabilistically proves that some statement is true, meaning that it shows that there is at most a negligible probability that the statement is not true. A zk-SNARK is zero-knowledge, meaning it does not reveal any additional information which does not follow logically from the statement being proven.

The process of creating and verifying zk-SNARKs using Halo 2 starts with defining some class of statements to generate zk-SNARKs for. The definition takes the form of a "circuit." The name "circuit" comes from an analogy with hardware circuits, as in the integrated circuits (ICs) used to build digital computers. A Halo 2 circuit is really an abstract object which bears only a loose resemblance to a hardware circuit layout.

Hardware circuits in modern computers typically use boolean or binary circuits, where each gate has two possible states: on and off, also known as true and false, or 1 and 0. Halo 2 circuits use instead the elements of some finite field $\mathbb{F}$ as the possible gate states. The gate states are represented as variables ranging over $\mathbb{F}$, and the connections between the gates are represented as a system of polynomial constraints with a particular structure, consisting of custom gate constraints, lookup constraints, and equality constraints. Section 2 provides more detail on the structure of Halo 2 circuits.

A Halo 2 circuit takes as input a matrix of values, elements of the field $\mathbb{F}$. This matrix is known as an "instance." For a given instance, the circuit either is or is not satisfiable. The circuit is satisfiable if and only if there is a matrix of values, known as a "witness," which together with the instance satisfies all of the system's constraints.

A zk-SNARK proves that a circuit is satisfiable on a given instance. Producing such a zk-SNARK requires a witness satisfying the circuit together with the instance. Having such a zk-SNARK does not help to deduce a witness which shows that the instance is satisfiable; that is a consequence of the zero knowledge property.

Creating and verifying zk-SNARKs using Halo 2 involves creating Rust code which constructs a circuit. It also involves creating Rust code which, given a satisfiable instance and possibly some additional private information, produces a witness satisfying the circuit together with the instance. Also, it is necessary to design some encoding as instances of the objects represented by the instances, and to create some Rust code which performs the encoding.

You can think of the process of creating Rust code to construct a circuit and instances and witnesses as the process of "arithmetizing" a relation. The result is a circuit which is satisfiable on some instances and not others. This circuit expresses a relation, the relation which is true of all and only the objects encoded as instances where the circuit is satisfiable. If the circuit is correct, then it expresses the relation which was to be arithmetized.

The process of arithmetizing a relation is rather laborious and error prone when the relation to be arithmetized is complex. That is when the process is done by hand. It would be helpful to be able to automate the process, so that from a high level description of a relation in a math-like notation, it would be possible to generate Rust code which creates a circuit. Such a process also would need to create code which encodes objects in the domain of the relation as instances of that circuit, and code which generates witnesses satisfying the circuit together with any given instance encoding an object for which the relation is true.

The goal of this research is to describe (and prove the correctness of) a process for compiling a relation defined in a high-level math-like notation into an arithmetization of that relation. It is hoped that this research will lead to a compiler which generates practical, efficient Rust code for creating and verifying zk-SNARKs using Halo 2. This paper does not contain a full correctness proof for the provided constructions. Rather, it sketches a proof approach. There is work underway to formally prove the same results claimed in the proof sketch.

The high-level math-like notation for defining relations is Σ_1^1 formulas over the language of rings. This notation is defined in Section 3. The arithmetization process is described in Section 4, and proven correct in Section 5.

2 Halo 2 circuits

The following definition of Halo 2 circuits does not correspond exactly with the current implementation of Halo 2. As of this writing, Halo 2 does not support lookup tables with advice or instance columns, whereas this definition assumes that it does. Work on support for lookup tables with advice columns is underway, and there are no plans to support lookup tables with instance columns, but this is without loss of generality, because instance columns in lookup tables can be emulated by adding an advice column to use in the lookup table which is constrained to be equal to the corresponding instance column at all rows.

Another notable difference between Halo 2 in reality and Halo 2 as it is idealized in this paper is that in reality, Halo 2 requires some extra rows to be present and filled with random data. This paper ignores this detail, but accounting for this detail would not add much extra complexity.

For a positive integer x, let $[x]$ denote the set $\{1, ..., x\}$.

By definition, a "circuit" is a tuple

$$\mathcal{C} = (\mathbb{F}, \vec{c}, d, \vec{p}, \vec{\ell}, r, \vec{e}, X), \tag{1}$$

where:

1. $\mathbb{F}$ is a finite field. $\mathbb{F}$ is called the underlying field.
2. $\vec{c} = c_1, ..., c_n$, where for all $i \in [n]$,

$$c_i \in \{F, A, I\} \times \{E, N\}. \tag{2}$$

   Here F, A, I, E, N are distinct constants with the following interpretations:

F	Fixed column
A	Advice column
I	Instance column
E	Equality constrainable column
N	Non equality constrainable column

   $\vec{c}$ is called the column type vector.

3. d is a positive integer. d is called the polynomial degree bound.
4. $\vec{p} = p_1, ..., p_m$, where for all $i \in [m]$, p_i is a polynomial of degree $\leq d$ with coefficients in $\mathbb{F}$, where variable names are pairs $(j, k) \in [n] \times \mathbb{Z}$ where j is a column index and k is a relative row reference. $k = 0$ refers to the current row, and in general, k refers to the row whose index is the index of the current row plus k (with wraparound, modulo r).

   $\vec{p}$ is called the sequence of polynomial constraints.

5. $\vec{\ell} = \ell_1, ..., \ell_k$, where for all $i \in [k]$, ℓ_i is a tuple of the form $(\vec{x}, \vec{a})$ where for some positive integer q:

(a) $\vec{x} = x_1, ..., x_q$, where for all $j \in [q]$, x_j is a polynomial of degree $\leq d$ with coefficients in $\mathbb{F}$, where variables are pairs $(k, l) \in [n] \times \mathbb{Z}$ where k is a column index and l is a relative row reference. $\vec{x}$ is called the input expression.

(b) $\vec{a} = a_1, ..., a_q$, where for all $j \in [q]$, a_j is a column index ($a_j \in [n]$). $\vec{a}$ is called the vector of table columns.

$\vec{l}$ is called the sequence of lookup arguments.

6. r is a positive integer. r is called the number of rows.

7. $\vec{e} = e_1, ..., e_t$, where for all $i \in [t]$, $e_i \in ([n] \times [r])^2$ is a pair of pairs of indices, representing an equality constraint in the form of a pair of absolute cell references.

8. $X \in \mathbb{F}^{[u] \times [r]}$, where u is the number of fixed columns, contains the values of each fixed column in each row. [1]

A circuit $\mathcal{C} = (\mathbb{F}, \vec{c}, d, \vec{p}, \vec{\ell}, r, \vec{e}, X)$ arithmetizes a relation over matrices in $\mathbb{F}^{v \times r}$, where v is the number of instance columns in $\vec{c}$:

$$v = |\{i \in [n] \mid \pi_1(c_i) = I\}|. \tag{3}$$

Similarly, u is the number of fixed columns and w is the number of advice columns:

$$u = |\{i \in [n] \mid \pi_1(c_i) = F\}|. \tag{4}$$

$$w = |\{i \in [n] \mid \pi_1(c_i) = A\}|. \tag{5}$$

The next goal is to define what it means for a circuit to arithmetize a relation, or in other words, to define what relation a given circuit arithmetizes.

Without loss of generality, consider only circuits whose columns are ordered such that all fixed columns come before all non-fixed columns and all instance columns come before all advice columns. This is without loss of generality because the columns in a circuit can be permuted without changing the meaning. This restriction offers some notational convenience in what follows.

The relation which $\mathcal{C}$ arithmetizes can be defined as the set of instances $Y \in \mathbb{F}^{[v] \times [r]}$ such that there exists a witness $Z \in \mathbb{F}^{[w] \times [r]}$ such that the matrix

$$M = [XYZ] \in \mathbb{F}^{[n] \times [r]} \tag{6}$$

satisfies the constraints of $\mathcal{C}$. By definition, the constraints of $\mathcal{C}$ are the following conditions, which are defined for any matrix $M \in \mathbb{F}^{[n] \times [r]}$:

1. For all $i \in [u], j \in [r]$, $M_{i,j} = X_{i,j}$. That is to say, the first u columns of M are the fixed columns enumerated in X.

2. For all $i \in [m], j \in [r]$,

$$0 = \sum_{k \in [s]} \left(a_k \cdot \prod_{l \in [d_k]} M_{b_{k,l} + (0,j)} \right), \tag{7}$$

where

$$p_i = \sum_{k \in [s]} \left(a_k \cdot \prod_{l \in [d_k]} b_{k,l} \right), \tag{8}$$

[1]Matrices are written in column major notation in this paper, meaning the column index comes first and the row index second.

where for all $i \in [s]$, $d_i \leq d$ and $a_i \in \mathbb{F}$ and for all $j \in [d_i]$, $b_{i,j} \in [n] \times \mathbb{Z}$.

That is to say, the polynomial constraints are satisfied at each row.

3. For all $i \in [k]$, $\rho \in [r]$,
$$\vec{y} \in \{(M_{a_1,\lambda}, ..., M_{a_q,\lambda}) \mid \lambda \in [r]\}, \tag{9}$$
where
$$(\vec{x}, \vec{a}) = \ell_i, \tag{10}$$
$$\vec{x} = x_1, ..., x_q, \tag{11}$$
$$\vec{a} = a_1, ..., a_q, \tag{12}$$
$$\vec{y} = y_1, ..., y_q, \tag{13}$$
and for each $j \in [q]$,
$$y_j = \sum_{k \in [s]} \left(a_k \cdot \prod_{l \in [d_k]} M_{b_{k,l}+(0,\rho)} \right), \tag{14}$$
where
$$x_j = \sum_{k \in [s]} \left(a_k \cdot \prod_{l \in [d_k]} b_{k,l} \right), \tag{15}$$
where $d_k \leq d$ for all $k \in [s]$.

That is to say, the lookup argument constraints are satisfied.

4. For all $i \in [t]$, $M_a = M_b$, where $(a, b) = e_i$. That is to say, the equality constraints are satisfied.

5. For all $i \in [t]$, where $(a, b) = e_i$, $a, b \in$ EQCs, where
$$\text{EQCs} = \{i \in [n] \mid \pi_2(c_i) = E\}. \tag{16}$$
That is to say, the equality constraints refer only to cells in the equality constrainable columns.

In practice, the domain of the relation to be arithmetized is not always isomorphic to a matrix over a finite field. It is helpful to define also what it means to arithmetize a relation in these cases.

Let D be a set, the domain of the relation to be arithmetized. Let $R \subseteq D$ be the relation to be arithmetized. Assume that R is the projection of another relation $Q \subseteq D \times W$, where W is another set, the witness set. To say that R is the projection of Q means that $R = \pi_1(Q)$, or in other words
$$R = \{x \in D \mid (x, w) \in Q \text{ for some } w \in W\}. \tag{17}$$

In this general setting, a relation is a set $R \subseteq D$, where D is another set, called the domain. By definition, an "arithmetization" of R is a pair $(\mathcal{C}, f)$, where $\mathcal{C}$ is a circuit
$$\mathcal{C} = (\mathbb{F}, \vec{c}, d, \vec{p}, \vec{l}, r, \vec{e}, X) \tag{18}$$
which arithmetizes a relation $R' \subseteq \mathbb{F}^{v \times r}$, and f is a computable function $f : D \to \mathbb{F}^{v \times r}$, such that for all $x \in D$, $x \in R$ iff $f(x) \in R'$.

3 Σ^1_1 formulas over the language of rings

The goal of this research is to be able to arithmetize relations into Halo 2 by describing them in a high-level math-like notation and running a compiler which translates the notation into circuits. What is a good notation for describing relations which can be arithmetized into Halo 2 circuits? Perhaps unsurprisingly, Halo 2 circuits turn out to have a close relationship with a certain class of logic formulas, namely Σ^1_1 formulas over the language of rings (not in the usual sense but in a specialized sense defined in this section).

Section 4 shows how to arithmetize an arbitrary Σ^1_1 formula over the language of rings, as a Halo 2 circuit. Section 5 proves that the arithmetization is correct, in the sense that the resulting circuit is satisfiable on only the instances which satisfy the relation, and all instances which satisfy the relation and the numeric size limits of the circuit. Each circuit can accept only finitely many instances. An infinite relation (i.e., a relation true of infinitely many different things) can be arithmetized as the limit of an infinite sequence of circuits, each recognizing progressively larger subsets of the relation.

What are Σ^1_1 formulas over the language of rings? In general, they are formulas of second-order logic, where the only second-order quantifiers are zero or more existential second-order quantifiers on the outside of the statement, before any other quantifiers. The language of rings is $\{\cdot, +, 0, 1, -1\}$, where $\cdot$ and $+$ are binary operations and $0, 1$, and -1 are constant symbols. Σ^1_1 is one of the first levels of the analytical hierarchy, which is the second-order extension of the arithmetical hierarchy of first-order formulas.

In the following presentation of Σ^1_1 formulas over the language of rings, quantifiers are always bounded by constants. This is different from other presentations. This is more than a cosmetic difference, meaning that the definition of Σ^1_1 formulas found here is not a substitute for definitions found elsewhere.

Think of a formula as defining a relation over integers, that is, a subset of $\mathbb{Z}^i$ for some positive integer i. The effect of the modification of the definition of Σ^1_1 formulas, to require constant bounds on all quantifiers, is that the truth or falsehood of a formula can be evaluated in a finite amount of time. This picture is spelled out in more detail in Section 4. In short, bounding quantifiers by constants allows for all (thusly restricted) Σ^1_1 formulas over the language of rings to be arithmetized into circuits.

The following presentation will use de Bruijn indices for variable names. A de Bruijn index is a positive integer representing a reference to an enclosing scope, with 1 referring to the variable declared in the innermost scope. Consider a formula with three existential quantifiers:

$$\exists x < a.\ \exists y < b.\ \exists z < c.\ \phi(x, y, z). \tag{19}$$

Here $\phi(x, y, z)$ denotes some formula with the free variables x, y, z. Replacing the variable names with de Bruijn indices, this formula becomes:

$$\exists < a.\ \exists < b.\ \exists < c.\ \phi(3, 2, 1). \tag{20}$$

To avoid confusion in formulas featuring both numeric constants and de Bruijn indices, let x_i denote the de Bruijn index i, so the example formula gets written:

$$\exists < a.\ \exists < b.\ \exists < c.\ \phi(x_3, x_2, x_1). \tag{21}$$

Variables in Σ^1_1 formulas are of two kinds: first-order variables which denote numbers, and second-order variables which denote functions of one or more arguments. An n-ary function variable denotes a function which takes n numbers as input and outputs a number. Second-order variables may denote partial functions, which are not defined on all inputs.

First-order and second-order variables are in two different namespaces. The de Bruijn index i may, depending on the context in which it appears, denote either a first-order variable (in which case the index is written as x_i) or a second-order variable (in which case the index is written as f_i).

The first step in defining Σ^1_1 formulas (over the language of rings) is to define a language of terms. A term is a syntactic object which denotes a number, given a context which bestows values to the variables it contains. Terms are defined by the following recursive definition.

1. For each positive integer i, x_i is a term. x_i is called a first-order variable.
2. For each positive integer i and all sequences of terms $\tau_1, ..., \tau_n$, $f_i(\tau_1, ..., \tau_n)$ is a term, if f_i has arity n. $f_i(\tau_1, ..., \tau_n)$ is called a function application.
3. For all terms τ, μ, $(\tau + \mu)$ is a term and $(\tau \cdot \mu)$ is a term.
4. 0 is a term. 1 is a term. -1 is a term. These are called constant symbols.

The arity of a second order variable is implicitly defined by its occurrences in a term or formula. All of the occurrences of a second order variable must have the same arity in a syntactically valid term or formula.

In addition, it is useful to define positive constant terms, which can be done with the following recursive definition.

1. 1 is a positive constant term.
2. For all positive constant terms τ, μ, $(\tau + \mu)$ is a positive constant term.

This recursive definition of positive constant terms includes closure under addition. It could also include closure under multiplication, but it does not for the reason that multiplication is not needed; without loss of generality, multiplication can be omitted.

First-order formulas (over the language of rings) are defined by the following recursive definition.

1. For all terms τ, μ,
$$(\tau = \mu) \tag{22}$$
is a first-order formula. $\tau = \mu$ is called an atomic formula or an equation.
2. For all first-order formulas ϕ,
$$\neg\phi \tag{23}$$
is a first-order formula. $\neg\phi$ is called a negation.
3. For all first-order formulas ϕ, ψ,
$$(\phi \wedge \psi) \tag{24}$$
is a first-order formula. $(\phi \wedge \psi)$ is called a conjunction.
4. For all first-order formulas ϕ, ψ,
$$(\phi \vee \psi) \tag{25}$$
is a first-order formula. $(\phi \vee \psi)$ is called a disjunction.
5. For all first-order formulas ϕ, ψ,
$$(\phi \rightarrow \psi) \tag{26}$$
is a first-order formula. $(\phi \rightarrow \psi)$ is called an implication.

6. For all first-order formulas ϕ and positive constant terms β,
$$\forall < \beta.\ \phi \tag{27}$$
is a first-order formula. $\forall$ is called the universal quantifier. β is called the upper bound.

7. For all first-order formulas ϕ and positive constant terms β,
$$\exists < \beta.\ \phi \tag{28}$$
is a first-order formula. $\exists$ is called the first-order existential quantifier. β is called the upper bound.

Σ_1^1 formulas (over the language of rings) are defined by the following recursive definition.

1. Every first-order formula is a Σ_1^1 formula.

2. For all Σ_1^1 formulas ϕ and positive constant terms γ and non-empty sequences of positive constant terms $\beta_1, ..., \beta_n$,
$$\exists_f < \gamma(< \beta_1, ..., < \beta_n).\ \phi \tag{29}$$
is a Σ_1^1 formula. $\exists_f$ is called the second-order existential quantifier. n is called the arity of the function. $\beta_1, ..., \beta_n$ are called the upper bounds of the dimensions of the domain of the function. γ is called the upper bound of the codomain of the function.

It is often useful to know the set of variables which are free in a given formula. How is this set defined when variables are denoted by de Bruijn indices, which can denote different variables in different contexts? Even two free occurrences of the same de Bruijn index in the same formula can denote different variables if they are in different contexts with different levels of quantifier nesting. The trick here is to define the set of free variables as the de Bruijn indices, relative to the outermost scope of the formula (outside of all the quantifiers), of all the free variables referenced in the program, regardless of which de Bruijn indices denote those variables in the context(s) in which those variables are referenced. The following recursive equations define the set of free variables in a Σ_1^1 formula or a term.

$$\text{free} : \{\phi \mid \phi \text{ is a } \Sigma_1^1 \text{ formula}\} \oplus \{\tau \mid \tau \text{ is a term}\} \to \{x_i\}_{i\in\mathbb{Z}^+} \oplus \{f_i\}_{i\in\mathbb{Z}^+} \tag{30}$$

$$\text{free}(x_i) = \{x_i\} \tag{31}$$

$$\text{free}(f_i(\tau_1, ..., \tau_n)) = \{f_i\} \cup \bigcup_{i\in[n]} \text{free}(\tau_i) \tag{32}$$

$$\text{free}(\tau + \mu) = \text{free}(\tau) \cup \text{free}(\mu) \tag{33}$$

$$\text{free}(\tau \cdot \mu) = \text{free}(\tau) \cup \text{free}(\mu) \tag{34}$$

$$\text{free}(0) = \text{free}(1) = \text{free}(-1) = \emptyset \tag{35}$$

$$\text{free}(\tau = \mu) = \text{free}(\tau) \cup \text{free}(\mu) \tag{36}$$

$$\text{free}(\neg\phi) = \text{free}(\phi) \tag{37}$$

$$\text{free}(\phi \wedge \psi) = \text{free}(\phi) \cup \text{free}(\psi) \tag{38}$$

$$\text{free}(\phi \vee \psi) = \text{free}(\phi) \cup \text{free}(\psi) \tag{39}$$

$$\text{free}(\phi \to \psi) = \text{free}(\phi) \cup \text{free}(\psi) \tag{40}$$

$$\text{free}(\forall < \beta.\ \phi) = \{x_i \mid x_{i+1} \in \text{free}(\phi)\} \cup \{f_i \mid f_i \in \text{free}(\phi)\} \tag{41}$$

$$\text{free}(\exists < \beta.\ \phi) = \{x_i \mid x_{i+1} \in \text{free}(\phi)\} \cup \{f_i \mid f_i \in \text{free}(\phi)\} \tag{42}$$

$$\text{free}(\exists_f < \gamma(< \vec{\beta}).\ \phi) = \{x_i \mid x_i \in \text{free}(\phi)\} \cup \{f_i \mid f_{i+1} \in \text{free}(\phi)\} \tag{43}$$

Relative to a suitable model, it is possible to define whether any given Σ^1_1 formula is true or false. For this context, a model, by definition, is a tuple

$$M = (R, \cdot, +, 0, 1, -1, <, F, S), \tag{44}$$

where:

1. $(R, \cdot, +, 0, 1, -1)$ is a ring:
   (a) R is a set.
   (b) $\cdot : R \times R \to R$ is a binary operation.
   (c) $+ : R \times R \to R$ is a binary operation.
   (d) $0, 1, -1 \in R$.
   (e) $+$ is associative and commutative, meaning, for all $x, y, z \in R$,

   $$(x + y) + z = x + (y + z), \tag{45}$$

   $$x + y = y + x. \tag{46}$$

   (f) Each element of R has an additive inverse, meaning, for each $x \in R$ there is $y \in R$ such that $x + y = 0$.
   (g) $\cdot$ is associative, meaning, for all $x, y, z \in R$,

   $$(x \cdot y) \cdot z = x \cdot (y \cdot z) \tag{47}$$

   (h) 0 is the additive identity, meaning, for each $x \in R$,

   $$x + 0 = x = 0 + x. \tag{48}$$

   (i) 1 is the multiplicative identity, meaning, for each $x \in R$,

   $$x \cdot 1 = x = 1 \cdot x. \tag{49}$$

   (j) -1 is the additive inverse of 1, meaning $1 + (-1) = 0$.

(k) The distributive law holds, meaning, for all $x, y, z \in R$,

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z), \tag{50}$$

$$(y + z) \cdot x = (y \cdot x) + (z \cdot x). \tag{51}$$

2. $< \subseteq R \times R$ is a strict total ordering relation on R with a least element:
   (a) $<$ is antisymmetric, meaning there is no $x \in R$ such that $x < x$.
   (b) $<$ is transitive, meaning for all $x, y, z \in R$, if $x < y$ and $y < z$ then $x < z$.
   (c) $<$ is connected, meaning for all $x, y \in R$, if $x \neq y$ then either $x < y$ or $y < x$.
   (d) $<$ has a least element, meaning there exists a $z \in R$ such that for all $x \in R$, if $x \neq z$ then $z < x$. (Such a z is necessarily unique.)
3. $F : \mathbb{Z}^+ \to R$ is a partial function mapping de Bruijn indices naming first-order variables to their corresponding values (if any).
4. $S : \mathbb{Z}^+ \to \sum_{i \in \mathbb{Z}^+}(R^i \to R)$ is a partial function mapping de Bruijn indices naming second-order variables to their corresponding values (if any). The values are partial functions of variable arity from R to R. Here $\sum_{i \in \mathbb{Z}^+}$ denotes the indexed coproduct or disjoint union operation with i ranging over the positive integers.

By definition, M is an "integral model" when

$$(R, +, 0, 1, -1) = (\mathbb{Z}, +, 0, 1, -1) \tag{52}$$

is the standard ring of integers and $<$ puts the integers into the following order:

$$1, 2, ..., 0, -1, -2, ... \tag{53}$$

Let $M = (R, \cdot, +, 0, 1, -1, <, F, S)$ be a model. Let $y \in R$. Define the model

$$M[x_1 \mapsto y] = (R, \cdot, +, 0, 1, <, F', S) \tag{54}$$

by letting

$$\begin{array}{rcl} F'(1) & = & y, \\ F'(n+1) & = & F(n). \end{array} \tag{55}$$

Let n be a positive integer. Let $g : R^n \to R$ be a partial function. Define the model

$$M[f_1 \mapsto g] = (R, \cdot, +, 0, 1, F, S') \tag{56}$$

by letting

$$\begin{array}{rcl} S'(1) & = & g, \\ S'(n+1) & = & S(n). \end{array} \tag{57}$$

The definitions just given of $M[x_1 \mapsto y]$ and $M[f_1 \mapsto g]$ explain how to update a model with a new variable mapping at the least de Bruijn index, pushing up all the existing de Bruijn index mappings. These operations are useful for dealing with quantification in the denotational semantics which follows below.

Also helpful for defining the denotational semantics will be an extension of the $[i]$ notation previously defined. Previously, $[i]$ was defined as $\{1, ..., i\}$ for a positive integer i. Generalizing this to a ring R for an arbitrary model M, let $[i]$ be defined as $\{x \in R \mid z \leq x \leq i\}$, where z is the least element of R under $<$ and $\leq$ is the non-strict version of $<$.

Given a model,

$$M = (R, \cdot, +, 0, 1, -1, <, F, S), \tag{58}$$

it is possible to define the denotations of terms and formulas, such as by the following recursive definition. The denotation of a term is an element of R, whereas the denotation of a formula is a truth value. The set of truth values is the set $\{0, 1\}$, where 0 represents false and 1 represents true. Due to the partiality of F, S, and the functions in the codomain of S, not every term or formula has a denotation in every model. The following recursive clauses define the denotation $\delta_M(\tau)$ for each term τ which has a denotation in M and the denotation $\delta_M(\phi)$ for each term ϕ which has a denotation in M.

1. For all positive integers i,
$$\delta_M(x_i) = F(i), \tag{59}$$
if $F(i)$ is defined.

2. For all positive integers i and non-empty sequences of terms $\tau_1, ..., \tau_n$,
$$\delta_M(f_i(\tau_1, ..., \tau_n)) = S(i)(\delta_M(\tau_1), ..., \delta_M(\tau_n)), \tag{60}$$
if $S(i)$ is defined, and $\delta_M(\tau_i)$ is defined for each i, and
$$S(i)(\delta_M(\tau_1), ..., \delta_M(\tau_n)) \tag{61}$$
is defined.

3. For all terms τ, μ,
$$\delta_M(\tau + \mu) = \delta_M(\tau) + \delta_M(\mu), \tag{62}$$
if $\delta_M(\tau)$ and $\delta_M(\mu)$ are defined.

4. For all terms τ, μ,
$$\delta_M(\tau \cdot \mu) = \delta_M(\tau) \cdot \delta_M(\mu), \tag{63}$$
if $\delta_M(\tau)$ and $\delta_M(\mu)$ are defined.

5. $\delta_M(0) = 0$.

6. $\delta_M(1) = 1$.

7. $\delta_M(-1) = -1$.

8. For all terms τ, μ,
$$\delta_M(\tau = \mu) = 1 \tag{64}$$
if $\delta_M(\tau)$ and $\delta_M(\mu)$ are defined and
$$\delta_M(\tau) = \delta_M(\mu). \tag{65}$$

9. For all terms τ, μ,
$$\delta_M(\tau = \mu) = 0 \tag{66}$$
if $\delta_M(\tau)$ and $\delta_M(\mu)$ are defined and
$$\delta_M(\tau) \neq \delta_M(\mu). \tag{67}$$

10. For all first-order formulas ϕ,
$$\delta_M(\neg\phi) = 1 - \delta_M(\phi), \tag{68}$$
if $\delta_M(\phi)$ is defined.

11. For all first-order formulas ϕ, ψ,
$$\delta_M(\phi \wedge \psi) = \min\{\delta_M(\phi), \delta_M(\psi)\}, \tag{69}$$
if $\delta_M(\phi)$ and $\delta_M(\psi)$ are defined.

12. For all first-order formulas ϕ, ψ,
$$\delta_M(\phi \vee \psi) = \max\{\delta_M(\phi), \delta_M(\psi)\}. \tag{70}$$

13. For all positive constant terms β and first-order formulas ϕ,
$$\delta_M(\forall < \beta.\ \phi) = \min_{i \in [\delta_M(\beta)]} \delta_{M[x_1 \mapsto i]}(\phi), \tag{71}$$
if $\delta_{M[x_1 \mapsto i]}(\phi)$ is defined for each i.

14. For all positive constant terms β and first-order formulas ϕ,
$$\delta_M(\exists < \beta.\ \phi) = \max_{i \in [\delta_M(\beta)]} \delta_{M[x_1 \mapsto i]}(\phi), \tag{72}$$
if $\delta_{M[x_1 \mapsto i]}(\phi)$ is defined for each i.

15. For all positive constant terms γ and non-empty sequences of positive constant terms $\beta_1, ..., \beta_n$ and Σ^1_1 formulas ϕ,
$$\delta_M(\exists_f < \gamma(< \beta_1, ..., < \beta_n).\ \phi) = \max_{g \in [\delta_M(\gamma)]^{\prod_{j \in [n]} [\delta_M(\beta_j)]}} \delta_{M[f_1 \mapsto g]}(\phi), \tag{73}$$
if $\delta_{M[f_1 \mapsto g]}(\phi)$ is defined for each g.

4 Arithmetizing Σ^1_1 formulas over the language of rings

The primary goal of this section is: given a relation presented as a Σ^1_1 formula, find an arithmetization of the relation. This section presents a scheme for doing so, proven correct in Section 5.

Let Φ be a Σ^1_1 formula to be arithmetized. Without loss of generality, assume that Φ is of the form
$$\Phi = \Sigma\Delta\Psi, \tag{74}$$
where Σ is a string of $n \geq 0$ second-order quantifiers,[2]
$$\Sigma = \exists_f < \gamma_1(< \vec{\beta}_1).\ \cdots\ \exists_f < \gamma_n(< \vec{\beta}_n)., \tag{75}$$
Δ is a string of $m \geq 0$ first-order quantifiers,
$$\Delta = \Delta_1 < \epsilon_1.\ \cdots\ \Delta_m < \epsilon_m., \tag{76}$$

[2]In (75), $(< \vec{\beta}_i)$ is a shorthand for $(< \beta_{i,1}, ..., < \beta_{i,k_i})$.

where $\Delta_i \in \{\exists, \forall\}$ for all $i \in [m]$, and Ψ is a first-order formula without quantifiers. This assumption is without loss of generality because for any first-order formula, there is an equivalent formula in prenex normal form, meaning, with all quantifiers brought to the outside of the formula.

The arithmetization of Φ proceeds as follows.

Choose a power of two, 2^W. W is called the word size. Let $W = BN$. B is called the number of bits in a byte. N is called the number of bytes in a word. Choose these numbers such that the following inequalities hold.

$$\gamma_i \le 2^W, \tag{77}$$

for all $i \in [n]$.

$$\beta_{i,j} \le 2^W, \tag{78}$$

for all $i \in [n], j \in [k_i]$.

$$\epsilon_i \le 2^W, \tag{79}$$

for all $i \in [m]$.

Choose a finite field $\mathbb{F}$ such that $|\mathbb{F}|$, the cardinality or number of elements of $\mathbb{F}$, is greater than $2M_\delta$, where M_δ is the largest integer which is the denotation of any subterm of any term occurring in Φ in any integral model which satisfies Φ, wherein the free first-order variables assume integer values in the open interval $(-2^W, 2^W)$ and each n-ary free second-order variable assumes a value in the function space

$$(-2^W, 2^W)^{(-2^W, 2^W)^n} \tag{80}$$

with cardinality no greater than some chosen maximum M_F. This means that the number of possible values for a free n-ary function variable, in this arithmetization, is less than

$$2^{(W+1)\cdot M_F \cdot n}, \tag{81}$$

which represents the number of ways of making $M_F \cdot n$ choices (with repetition allowed) out of 2^{W+1} options.

If this scheme results in too many possible instance values, then each of the free first-order variables and each argument place and each of the outputs of each free second-order function variable can have their own individual bounds for the purposes of the arithmetization, with relatively straightforward changes to what follows. The solution presented here economizes on the number of columns in the circuit with the tradeoff of allowing for more possible instance values than might be needed.

What follows is, first, a high-level description of the arithmetization, and then a detailed, rigorous description.

Each row of the circuit's input matrix checks one possible combination of values of the universally quantified variables. It is not always necessary for the matrix to include rows for all possible combinations of values of the universally quantified variables. It is sufficient to include a *falsifying set* of combinations of values. A falsifying set is one such that for each unsatisfiable combination of instance values, and each possible combination of advice values, there is a combination of universally quantified variable values in the falsifying set which witnesses the falsehood of the statement.

The combinations of values of universally quantified variables are expressed in fixed columns. There are advice columns containing, in each row, the values of the first-order existentially quantified variables corresponding to the values of the universally quantified variables in the row. There are instance columns containing the values of the free variables. There are advice columns containing the values of the second-order existentially quantified variables. There are advice columns containing the values of all function application subterms. There are advice columns containing the truth values of all equality subformulas. Bytewise decompositions help to implement bounds checks, and there are fixed columns and advice columns which help to implement bytewise decompositions and (in)equality tests.

Bytewise decomposition of a non-negative (unsigned) integer uses N bytes, where a byte is an integer in the closed interval $[0, 2^B - 1]$. The integer x to be decomposed must be in the closed interval $[0, 2^W - 1]$. The bytewise decomposition is a sequence of bytes $x_1, ..., x_N$ such that

$$x = \sum_{i=1}^{N} 2^{B \cdot (i-1)} \cdot x_i. \tag{82}$$

Bytewise decomposition of a signed integer uses N bytes and a sign bit. The integer x to be decomposed must be in the open interval $(-2^W, 2^W)$. The bytewise decomposition consists of a sign bit $s \in \{1, -1\}$ and a sequence of bytes $x_1, ..., x_N$ such that

$$x = s \cdot \sum_{i=1}^{N} 2^{B \cdot (i-1)} \cdot x_i. \tag{83}$$

What follows is a description of each column in the input matrix, including a name for the column vector.

The instance columns are as follows.

1. For each first-order variable $x_i \in \text{free}(\Phi)$, an instance column $\vec{x}_i$. This column holds, in each row, the (same) value assigned to x_i.

2. For each second-order variable $f_i \in \text{free}(\Phi)$ with arity n,[3] instance columns

$$\vec{f}_i, \vec{w}_{i,1}, ..., \vec{w}_{i,n}. \tag{84}$$

   $\vec{f}_i$ holds the value of $f_i(\vec{\vec{w}}_i)$ in each row. $\vec{w}_{i,1}, ..., \vec{w}_{i,n}$ hold the values of $\vec{\vec{w}}_i$ in each row.

There are five fixed columns which are present in all cases. These five "fixed fixed" columns are depicted in Figure 1. These five fixed columns help with checking bytewise decompositions and creating truth tables for equality and inequality statements.

There are additional fixed columns which are some subset of zero or more of the columns depicted in Figure 2.

There are additional fixed columns, one for each universally quantified first-order variable, whose rows contain a falsifying set of combinations of values of the universally quantified first-order variables.

Setting aside bytewise decompositions and advice columns based on the bytewise decompositions, the advice columns are as follows:

1. $2n$ advice columns per second-order n-ary function variable free in Φ, used to track the deltas between inputs of adjacent table rows and the truth values of inequality statements about inputs of adjacent rows.

2. $3n+1$ advice columns per existentially quantified (second-order) n-ary function variable: $n+1$ columns are used to define the witness function as a lookup table, n columns track the deltas between inputs of adjacent rows, and n columns track the truth values of inequality statements about inputs of adjacent rows.

[3]The arity of a free second-order variable is determined by the arity of an occurrence of the variable at the head of a function application. The arity of the variable must be the same in all occurrences for the formula to be interpretable.

$$\begin{array}{ccccc}
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 \\
0 & 2 & 1 & 1 & 2 \\
0 & 3 & 1 & 1 & 3 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
0 & N & 1 & 1 & N \\
0 & N+1 & 0 & 0 & 1 \\
0 & N+2 & 0 & 0 & 2 \\
0 & N+3 & 0 & 0 & 3 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 2N & 0 & 0 & N \\
0 & 2N+1 & 0 & 0 & N \\
0 & 2N+2 & 0 & 0 & N \\
0 & 2N+3 & 0 & 0 & N \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 2^B-1 & 0 & 0 & N \\
0 & 2^B-1 & 0 & 0 & N \\
0 & 2^B-1 & 0 & 0 & N \\
\vdots & \vdots & \vdots & \vdots & \vdots
\end{array} \tag{85}$$

Figure 1: The five fixed columns featuring in all arithmetizations. The first column has value 1 in row 1 and 0 in all other rows. The second column has value 0 in row 1, and montonically increasing values up to the maximum value $2^B - 1$ which is repeated until the final row. The third column consists of $N + 1$ ones followed by zeroes. The fourth column consists of a zero followed by N ones followed by zeroes. The fifth column consists of a zero followed by two copies of the sequence $1, ..., N$, followed by copies of N up to the number of rows.

$$\begin{array}{ccccccccc}
0 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & \cdots & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & \cdots & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & \cdots & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & \cdots & 1 & 1 & 1 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \cdots & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & \cdots & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array} \tag{86}$$

Figure 2: The additional fixed columns, any of which may be present to help with bounds checking. These columns respectively contain the output columns of the indicator functions of each non-empty downward closed subset of the open interval $(0, 2^B)$ which is not the whole set. The related input column is the fixed column $\vec{b}$ in each case.

3. One advice column per existentially quantified first-order variable, used to define the value witnessing the truth of the statement Ψ in each row.

4. Two advice columns per equality subformula $\tau_i = \mu_i$ in Φ, one column $\vec{\delta_i}$ containing the difference $\tau_i - \mu_i$ in each row, and one column $\vec{\epsilon_i}$ containing the truth value of the equality claim;

$$\epsilon_{i,j} = \begin{cases} 0 & \text{if } \delta_{i,j} \neq 0, \\ 1 & \text{if } \delta_{i,j} = 0. \end{cases} \tag{87}$$

5. $n+1$ advice columns per n-ary function application subterm in Φ, containing the inputs and output of the subterm at each row.

Each bytewise decomposition of an unsigned number takes up N advice columns. Each bytewise decomposition of a signed number takes up $N+1$ advice columns. The following bytewise decompositions are included in the circuit:

1. $n+1$ unsigned and n signed bytewise decompositions per (second-order) n-ary function variable free in Φ, used to decompose the values in the function lookup table provided in the instance and the deltas between inputs in adjacent rows, respectively.

2. One unsigned bytewise decomposition per first-order variable free in Φ, used to decompose the value provided in the instance.

3. $n+1$ unsigned and n signed bytewise decompositions per existentially quantified (second-order) n-ary function variable, used to decompose the values in the witness function lookup table and the deltas between the input values in adjacent rows, respectively.

4. One unsigned bytewise decomposition per existentially quantified first-order variable, used to decompose the witness values in each row.

5. One signed bytewise decomposition per equality subformula $\tau_i = \mu_i$ in Φ, used to decompose the difference $\delta_i = \tau_i - \mu_i$.

In addition, N advice columns per equality subformula are needed to check the truth value of the equality subformula based on the bytewise decomposition. Also, $N \cdot n$ advice columns per second-order n-ary function variable are needed to check the truth values of the "delta is positive" statements used in checking the lexicographical ordering of function tables (discussed below). Up to $2N \cdot (n+1)$ additional advice columns per second-order n-ary function variable are needed to check the quantifier bounds. Finally, (up to) N advice columns per first-order existential quantifier are needed to check the quantifier bounds.

Here is an informal summary of the circuit constraints and how they are checked or expressed.

1. Each element of a bytewise decomposition is a byte, or in the case of a sign bit, a bit. This is checked using a lookup argument.

2. Each bytewise decomposition adds up to the number it is supposed to be a decomposition of. This is checked using a polynomial constraint.

3. Each truth value $\vec{\epsilon_i}$ of an equality subformula is computed correctly from $\vec{\delta_i}$. This is checked using lookup arguments to verify the truth values of whether or not each element of the bytewise decomposition is zero, and a polynomial constraint to verify the truth value of $\delta_i = 0$ for each row based on the truth values for the bytewise decompositions.

4. Each delta column $\vec{\delta}_i$ is computed correctly using addition and multiplication from the values of free and quantified variables and function application terms in each row. This is checked using polynomial constraints.

5. Each function application term input advice column is computed correctly using addition and multiplication from the values of free and quantified variables and function application terms in each row. This is checked using polynomial constraints.

6. Each function application term output advice column is computed correctly using the function application input advice columns and a lookup into a function table either in advice columns (if the function is existentially quantified) or in instance columns (if the function is a free variable). This is checked using lookup arguments.[4]

7. All quantifier bounds are satisfied. This is checked using bytewise decompositions and lookup arguments. The number of lookup arguments required depends on the divisibility of the quantifier bounds.

8. The truth value of the statement Ψ is 1 (true) in each row. This is checked using a polynomial constraint based on the truth values of the equality subformulas.

9. The values in first-order existential quantifier advice columns are equal across different rows when required for consistency of the description of the witness to Φ according to the quantifier ordering. This is checked using equality constraints.

10. The values in any one first-order free variable instance column are all equal to each other. This is checked using equality constraints.

11. The lookup tables for free and existentially quantified second-order variables each define a function, meaning that if f_i is an n-ary function variable and $L \subseteq \mathbb{F}^{n+1}$ is its lookup table, then for all $x \in \mathbb{F}^n$ and $y, y' \in \mathbb{F}$, if $(x, y) \in L$ and $(x, y') \in L$ then $y = y'$. This is checked using the following steps.

    (a) The rows of each lookup table are lexicographically ordered. This is checked using polynomial constraints and the truth values of inequality statements about inputs of adjacent rows. To say that $\vec{x} < \vec{y}$ in lexicographical ordering, where $\vec{x}$ and $\vec{y}$ are vectors of non-negative integers of length n, is to say that for some $1 \leq i \leq n$, $x_i < y_i$, and $x_j = y_j$ for all $0 \leq j < i$.

    (b) In adjacent rows with equal input column values, the output column values are the same. This is checked using a polynomial constraint.

4.1 Circuit columns

Next up, a more detailed description of the circuit just outlined, beginning with how to compute the number of columns of each of the three column types (fixed, advice, and instance), and how to refer to each column.

The number of fixed columns is no less than $u + 2$, where u is the number of universal quantifiers in Φ. The number of fixed columns is no more than $u + 1 + 2^B$. The number of fixed columns depends on how many lookup tables are required in order to implement the quantifier bounds checks, as follows.

Let $\mathbf{B} \subset \mathbb{Z}^+$ be the set of quantifier bounds plus the set of numbers $i + 1$ where i is the bound of a first-order existential quantifier or the bound on an input or the output of a second-order existential

[4]Being able to have free function variables in Φ requires Halo 2 to allow for instance columns to be used in the lookup table of a lookup argument.

u	is the number of universal quantifiers in Φ.
χ	is the number of byte lookup tables required to check the quantifier bounds in Φ.
k	is the number of first-order variables appearing free in Φ.
n	is the number of second-order variables appearing free in Φ.
a_i	is the arity of the ith second-order variable appearing free in Φ.
N	is the number of bytes per word.
m	is the number of second-order existential quantifiers in Φ.
b_i	is the arity of the ith second-order existential quantifier in Φ.
e	is the number of first-order existential quantifiers in Φ.
u	is the number of first-order universal quantifiers in Φ.
s	is the number of equality subformulas in Φ.
t	is the number of function application subterms in Φ.
c_i	is the arity of the ith function application subterm in Φ.
r	is the number of rows.
θ_i	is the upper bound on the ith first-order existential quantifier.
$\lambda_{i,j}$	is the upper bound on the jth input of the ith second-order existential quantifier.
$\eta_{i,\ell}$	is the ℓth byte of the byte decomposition of the upper bound on the output of the ith second-order existential quantifier.

Figure 3: Names of numbers used in the column naming scheme.

quantifier. Let $b \in \mathbf{B}$. b is a word; that is, $0 \leq b < 2^W$. Let $b_1, ..., b_N$ be a binary decomposition of b; that is, let

$$b = \sum_{i=1}^{N} 2^{B\cdot(i-1)} \cdot b_i \tag{88}$$

while $b_i \in [0, 2^B - 1]$ for all $i \in [N]$. Each nonzero b_i gives the upper bound of one of the sets to be captured by a fixed column. The number of these nonzero b_i over all $b \in \mathbf{B}$ (without double counting) is the number χ in the formula

$$\chi + u + 5 \tag{89}$$

which gives the number of fixed columns.

Figure 3 gives the names of various numbers used in the column naming scheme, defining the variables in the formula below.

The number of instance columns is

$$k + \sum_{i=1}^{n} (a_i + 1), \tag{90}$$

where k is the number of first-order variables appearing free in Φ, n is the number of second-order variables appearing free in Φ, and a_i is the arity of the ith free second-order variable.

The number of advice columns is

$$\begin{array}{rcl} kN & + & \left(\sum_{i=1}^{n} 3a_i(2+N) + N\right) \\ & + & \left(\sum_{i=1}^{m} 1 + 4b_i + N(4b_i + 2)\right) \\ & + & N(2n+1) + (3N+1)e + (3+2N)s \\ & + & \left(\sum_{i=1}^{t} c_i + 1\right). \end{array} \tag{91}$$

In order to concisely describe the circuit constraints, it is helpful to have a naming scheme for the cells in the circuit. The following naming scheme assigns names to each column vector, whereby each cell in a

Fixed

$\vec{z}$	$\{0\}$ indicator function range.
$\vec{b}$	Byte domain.
$\vec{b}_i$	Subset of bytes closed downward with max i.
$\vec{\psi}_i$	ith column of $x > 0$ truth table.
$\vec{v}_i$	Value of ith universally quantified variable.

Figure 4: Summary of fixed column vector naming scheme.

Instance

$\vec{x}_i$	Value of ith free first-order variable.
$\vec{f}_i$	Output column of lookup table for ith free second-order variable.
$\vec{a}_{i,j}$	Input column j of lookup table for ith free second-order variable.

Figure 5: Summary of instance column vector naming scheme.

column vector $\vec{v}$ in row i is implicitly assigned the name v_i. See also Figures 4, 5, 6 for a summary of the naming scheme.

The following names name fixed column vectors:

$$\vec{z} = \langle 1\ 0\ 0\ 0\ \cdots \rangle \tag{92}$$

$$\vec{b} = \langle 0\ 1\ 2\ \cdots\ (2^B - 1)\ (2^B - 1)\ (2^B - 1)\ \cdots \rangle \tag{93}$$

The following names:

$$\vec{\psi}_1, \vec{\psi}_2, \vec{\psi}_3 \tag{94}$$

name fixed column vectors which are respectively equal to the columns of the matrix depicted in Figure 7.

For any $i \in [2^B]$,

$$\vec{b}_i \tag{96}$$

is a fixed column vector containing i ones followed by $2^B - i$ zeroes. (Only the subset of these column vectors which are actually used are included in the circuit.)

For any $i \in [u]$, u the number of universal quantifiers in Φ,

$$\vec{v}_i \tag{97}$$

names a fixed column which holds the value of the ith universally quantified (first-order) variable.

For any $i \in [k]$, k the number of first-order variables appearing free in Φ,

$$\vec{x}_i \tag{98}$$

names an instance column which holds the value of the ith free first-order variable.

For any $i \in [n]$, n the number of second-order variables appearing free in Φ,

$$\vec{f}_i \tag{99}$$

names an instance column which is the output column of the function lookup table representing the value of the ith second-order variable appearing free in Φ.

Advice

$\vec{\sigma}_{i,j}$	$= \vec{a}_{i,j} - \langle 0 \ a_{i,j,1} \ \cdots \ a_{i,j,a_i-1} \rangle$.
$\vec{\tau}_{i,j,\pi}$	Truth value of $\vec{\sigma}_{i,j} \geq \pi$.
$\vec{g}_i$	Output column of lookup table for ith second-order existentially quantified variable.
$\vec{c}_{i,j}$	Input column j of lookup table for ith second-order existentially quantified variable.
$\vec{\rho}_{i,j}$	$= \vec{c}_{i,j} - \langle 0 \ c_{i,j,1} \ \cdots \ c_{i,j,r-1} \rangle$.
$\vec{\upsilon}_{i,j,\pi}$	Truth value of $\vec{\rho}_{i,j} \geq \pi$.
$\vec{w}_i$	Witness for ith first-order existentially quantified variable.
$\vec{\delta}_i$	Difference between term values for ith equality subformula.
$\vec{\epsilon}_i$	Truth value of ith equality subformula.
$\vec{h}_i$	Output of ith application subterm.
$\vec{y}_{i,j}$	jth input value of ith application subterm.
$\vec{x}_{i,\ell}$	ℓth byte of $\vec{x}_i$.
$\vec{f}_{i,\ell}$	ℓth byte of $\vec{f}_i$.
$\vec{a}_{i,j,\ell}$	ℓth byte of $\vec{a}_{i,j}$.
$\vec{\sigma}^{+/-}_{i,j}$	Sign of $\vec{\sigma}_{i,j}$.
$\vec{\sigma}_{i,j,\ell}$	ℓth byte of unsigned part of $\vec{\sigma}_{i,j}$.
$\vec{g}_{i,\ell}$	ℓth byte of $\vec{g}_i$.
$\vec{c}_{i,j,\ell}$	ℓth byte of $\vec{c}_{i,j}$.
$\vec{\rho}^{+/-}_{i,j}$	Sign of $\vec{\rho}_{i,j}$.
$\vec{\rho}_{i,j,\ell}$	ℓth byte of $\vec{\rho}_{i,j}$.
$\vec{w}_{i,\ell}$	ℓth byte of $\vec{w}_i$.
$\vec{\delta}^{+/-}_i$	Sign of $\vec{\delta}_i$.
$\vec{\delta}_{i,\ell}$	ℓth byte of $\vec{\delta}_i$.
$\vec{\upsilon}'_{i,j,\ell}$	Truth value of $\vec{\rho}_{i,j,\ell} > 0$.
$\vec{\tau}'_{i,j,\ell}$	Truth value of $\vec{\sigma}_{i,j,\ell} > 0$.
$\vec{\epsilon}_{i,\ell}$	Truth value of $\vec{\delta}_{i,\ell} = 0$.
$\vec{\omega}_{i,\ell,\pi}$	Truth value of $\vec{w}_{i,\ell} \in \vec{b}_{\theta_i+\pi}$.
$\vec{\gamma}_{i,j,\ell,\pi}$	Truth value of $\vec{c}_{i,j,\ell} \in \vec{b}_{\lambda_{i,j,\ell}+\pi}$.
$\vec{\kappa}_{i,\ell,\pi}$	Truth value of $\vec{g}_{i,j,\ell} \in \vec{b}_{\eta_{i,\ell}+\pi}$.

Figure 6: Summary of advice column vector naming scheme.

$$\begin{bmatrix}\vec{\psi}_1 & \vec{\psi}_2 & \vec{\psi}_3\end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 3 \\ \vdots & \vdots & \vdots \\ 1 & 1 & N \\ 0 & 0 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 3 \\ \vdots & \vdots & \vdots \\ 0 & 0 & N \end{bmatrix} \tag{95}$$

Figure 7: The fixed columns $\vec{\psi}_1$, $\vec{\psi}_2$, and $\vec{\psi}_3$ constitute the truth table for checking the truth values of statements of the form $x > 0$ based on a byte decomposition of x.

For any $i \in [n]$ and $j \in [a_i]$, a_i the arity of the ith second-order variable appearing free in Φ,

$$\vec{a}_{i,j} \tag{100}$$

names an instance column which is the jth input column of the function lookup table representing the value of the ith second-order variable appearing free in Φ.

For any $i \in [n]$ and $j \in [a_i]$,

$$\vec{\sigma}_{i,j} = \vec{a}_{i,j} - \langle 0 \; a_{i,j,1} \; \cdots \; a_{i,j,a_i-1} \rangle \tag{101}$$

names an advice column holding the differences between successive values of $\vec{a}_{i,j}$, and for any $\pi \in \{0, 1\}$,

$$\vec{\tau}_{i,j,\pi} \tag{102}$$

names an advice column such that for all $k \in [r]$,

$$\tau_{i,j,\pi,k} = \begin{cases} 1 & \text{if } \vec{\sigma}_{i,j,k} \geq \pi, \\ 0 & \text{otherwise.} \end{cases} \tag{103}$$

For any $i \in [m]$,

$$\vec{g}_i \tag{104}$$

names an advice column which is the output column of the lookup table definition of the witness value for the ith existentially quantified second-order variable.

For any $i \in [m], j \in b_i$,

$$\vec{c}_{i,j} \tag{105}$$

names an advice column which is the input column of the lookup table definition of the witness value for the ith existentially quantified second-order variable.

For any $i \in [m], j \in b_i$,

$$\vec{\rho}_{i,j} = \vec{c}_{i,j} - \langle 0 \; \vec{c}_{i,j,1} \; \cdots \; c_{i,j,r-1} \rangle \tag{106}$$

names an advice column which holds the differences between successive rows of $\vec{c}_{i,j}$, and $\pi \in \{0, 1\}$,

$$\vec{\upsilon}_{i,j,\pi} \tag{107}$$

names an advice column such that for all $\pi \in \{0,1\}, k \in [r]$,

$$\vec{v}_{i,j,\pi,k} = \begin{cases} 1 & \text{if } \rho_{i,j,k} \geq \pi, \\ 0 & \text{otherwise.} \end{cases} \tag{108}$$

For all $i \in [e]$,

$$\vec{w}_i \tag{109}$$

names an advice column containing the witness for the ith first-order existentially quantified variable. The witness consists of one value for each checked value of the first-order universally quantified variables. For any universal quantifier which appears after the ith existential quantifier, the values of $\vec{w}_i$ must be independent of the value of the universally quantified variable. So for example, if the ith first-order existential quantifier does not appear inside a universal quantifier, then the values of $\vec{w}_i$ must all be equal.

For all $i \in [s]$,

$$\vec{\delta}_i \tag{110}$$

names an advice column containing the values of the difference $\tau_i - \mu_i$, where $\tau_i = \mu_i$ is the ith equality subformula in Φ, and

$$\vec{\epsilon}_i \tag{111}$$

names an advice column such that for all $j \in [r]$,

$$\epsilon_{i,j} = \begin{cases} 1 & \text{if } \delta_{i,j} = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{112}$$

For all $i \in [t]$,

$$\vec{h}_i \tag{113}$$

names an advice column which contains the output values at each row of the ith function application term in Φ.

For all $i \in [t], j \in c_i$,

$$\vec{c}_{i,j} \tag{114}$$

names an advice column which contains the input values at each row of the jth argument of the ith function application subterm in Φ.

The following names name advice columns which are parts of byte decompositions of the values in the columns with coordinated names:

1. $\vec{x}_{i,\ell}$, for all $i \in [k], \ell \in [N]$. (They decompose $\vec{x}_i$.)
2. $\vec{f}_{i,\ell}$, for all $i \in [n], \ell \in [N]$. (They decompose $\vec{f}_i$.)
3. $\vec{a}_{i,j,\ell}$, for all $i \in [n], j \in [a_i], \ell \in [N]$. (They decompose $\vec{a}_{i,j}$.)
4. $\vec{\sigma}^{+/-}_{i,j}, \vec{\sigma}_{i,j,\ell}$, for all $i \in [n], j \in [a_i], \ell \in [N]$. (They decompose $\vec{\sigma}_{i,j}$.)
5. $\vec{g}_{i,\ell}$, for all $i \in [m], \ell \in [N]$. (They decompose $\vec{g}_{i,\ell}$.)
6. $\vec{c}_{i,j,\ell}$, for all $i \in [m], j \in b_i, \ell \in [N]$. (They decompose $\vec{c}_{i,j}$.)
7. $\vec{\rho}^{+/-}_{i,j}, \vec{\rho}_{i,j,\ell}$, for all $i \in [m], j \in b_i, \ell \in [N]$. (They decompose $\vec{\rho}_{i,j}$.)
8. $\vec{w}_{i,\ell}$, for all $i \in [e]$. (They decompose $\vec{w}_i$.)

9. $\vec{\delta}_i^{+/-}, \delta_{i,\ell}$, for all $i \in [s]$. (They decompose $\vec{\delta}_i$.)

For all $i \in [m], j \in [b_i], \ell \in [N]$,

$$\vec{\upsilon}_{i,j,\ell} \tag{115}$$

names an advice column such that for each $k \in [r]$, $\upsilon_{i,j,\ell,k}$ is the truth value of the formula

$$\rho_{i,j,\ell,k} > 0. \tag{116}$$

For all $i \in [s], \ell \in [N]$,

$$\epsilon_{i,\ell} \tag{117}$$

names an advice column such that for each $k \in [r]$, $\epsilon_{i,\ell,k}$ is the truth value of the formula

$$\delta_{i,\ell,k} = 0. \tag{118}$$

For all $i \in [n], j \in [a_i]$, and $\ell \in [N]$,

$$\vec{\tau}'_{i,j,\ell} \tag{119}$$

names an advice column such that for each $k \in [r]$, $\tau'_{i,j,\ell,k}$ is the truth value of the formula

$$\sigma_{i,j,\ell,k} > 0. \tag{120}$$

For all $i \in [m], j \in [b_i]$, and $\ell \in [N]$,

$$\vec{\upsilon}'_{i,j,\ell} \tag{121}$$

names an advice column such that for all $k \in [r]$, $\upsilon'_{i,j,\ell,k}$ is the truth value of the formula

$$\rho_{i,j,\ell,k} > 0. \tag{122}$$

For all $i \in [e], \ell \in [N], \pi \in \{0, 1\}$,

$$\vec{\omega}_{i,\ell,\pi} \tag{123}$$

names an advice column such that for each $k \in [r]$, $\omega_{i,\ell,k}$ is the truth value of the formula

$$w_{i,\ell,k} \in \vec{b}_{\theta_{i,\ell}+\pi}. \tag{124}$$

For all $i \in [m], j \in [b_i], \ell \in [N], \pi \in \{0, 1\}$,

$$\vec{\gamma}_{i,j,\ell,\pi} \tag{125}$$

names an advice column such that for each $k \in [r]$, $\gamma_{i,j,\ell,k}$ is the truth value of the formula

$$c_{i,j,\ell,k} \in \vec{b}_{\lambda_{i,j}+\pi}. \tag{126}$$

For all $i \in [m], \ell \in [N], \pi \in \{0, 1\}$,

$$\vec{\kappa}_{i,\ell,\pi} \tag{127}$$

names an advice column such that for each $k \in [r]$, $\kappa_{i,\ell,k}$ is the truth value of the formula

$$g_{i,\ell,k} \in \vec{b}_{\eta_i+\pi,\ell} \tag{128}$$

That completes the description of the columns which the circuit of the arithmetization of Φ has. The next step in defining the circuit is to state the constraints of the circuit, which ensure that the values in the circuit matrix have their intended interpretations and the circuit arithmetizes the intended relation.

4.2 Circuit constraints

The circuit arithmetizing Φ expresses the following constraints on the input matrix. For definitions of the names referenced in the constraint statements, see Subsection 4.1 and in particular Figures 3, 4, 5, and 6.

Constraints are of three types: lookup constraints, equality constraints, and polynomial constraints. Polynomial constraints apply at each row and use cell references with relative row indices. Equality constraints apply between individual cells and use absolute cell references. Lookup constraints apply between subsets of the columns, asserting that every subrow in a subset of the columns is a subrow in another subset of the columns. Lookup constraints will be expressed in the form $(x_1, ..., x_n) \in (\vec{v}_1, \cdots, \vec{v}_n)$, where $(x_1, ..., x_n)$ is a tuple of elements of $\mathbb{F}$ and $\vec{v}_1, ..., \vec{v}_n$ is a same-length tuple of column vectors. This notation means that there is some row in the table with columns $\vec{v}_1, ..., \vec{v}_n$ which is equal to $(x_1, ..., x_n)$.

Each byte element of a bytewise decomposition is a byte and each sign element is 0 or 1. This constraint is expressed by the following statements.

1. For all $i \in [k], \ell \in [N], j \in [r]$,
$$x_{i,\ell,j} \in \vec{b}. \tag{129}$$
2. For all $i \in [n], j \in [a_i], \ell \in [N], k \in [r]$,
$$f_{i,\ell,j} \in \vec{b}, \tag{130}$$
$$a_{i,j,\ell,k} \in \vec{b}, \tag{131}$$
$$\sigma^{+/-}_{i,j,k} \in \vec{z}, \tag{132}$$
$$\sigma_{i,j,\ell,k} \in \vec{b}. \tag{133}$$
3. For all $i \in [m], j \in [b_i], \ell \in [N], k \in [r]$,
$$g_{i,\ell,k} \in \vec{b}, \tag{134}$$
$$c_{i,j,\ell,k} \in \vec{b}, \tag{135}$$
$$\rho^{+/-}_{i,j,k} \in \vec{z}, \tag{136}$$
$$\rho_{i,j,\ell,k} \in \vec{b}. \tag{137}$$
4. For all $i \in [e], \ell \in [N], k \in [r]$,
$$\vec{w}_{i,\ell,k} \in \vec{b}. \tag{138}$$
5. For all $i \in [s], \ell \in [N], k \in [r]$,
$$\delta^{+/-}_{i,k} \in \vec{z}, \tag{139}$$
$$\delta_{i,\ell,k} \in \vec{b}. \tag{140}$$

Each bytewise decomposition adds up to the value it decomposes. This constraint is expressed by the following statements.

1. For all $i \in [k], j \in [r]$,

$$x_{i,j} = \sum_{\ell=1}^{N} 2^{B\cdot(\ell-1)} \cdot x_{i,\ell,j}. \tag{141}$$

2. For all $i \in [n], j \in [a_i], k \in [r]$,

$$f_{i,k} = \sum_{\ell=1}^{N} 2^{B\cdot(\ell-1)} \cdot f_{i,\ell,k}, \tag{142}$$

$$a_{i,j,k} = \sum_{\ell=1}^{N} 2^{B\cdot(\ell-1)} \cdot a_{i,j,\ell,k}, \tag{143}$$

$$\sigma_{i,j,k} = (2 \cdot \sigma^{+/-}_{i,j,k} - 1) \cdot \sum_{\ell=1}^{N} 2^{B\cdot(\ell-1)} \cdot \sigma_{i,j,\ell,k}. \tag{144}$$

3. For all $i \in [m], j \in [b_i], k \in [r]$,

$$g_{i,k} = \sum_{\ell=1}^{N} 2^{B\cdot(\ell-1)} \cdot g_{i,\ell,k}, \tag{145}$$

$$c_{i,j,k} = \sum_{\ell=1}^{N} 2^{B\cdot(\ell-1)} \cdot c_{i,j,\ell,k}, \tag{146}$$

$$\rho_{i,j,k} = (2 \cdot \rho^{+/-}_{i,j,k} - 1) \cdot \sum_{\ell=1}^{N} 2^{B\cdot(\ell-1)} \cdot \rho_{i,j,\ell,k}. \tag{147}$$

4. For all $i \in [e], k \in [r]$,

$$w_{i,k} = \sum_{\ell=1}^{N} 2^{B\cdot(\ell-1)} \cdot w_{i,\ell,k}. \tag{148}$$

5. For all $i \in [s], k \in [r]$,

$$\delta_{i,k} = (2 \cdot \delta^{+/-}_{i,k} - 1) \cdot \sum_{\ell=1}^{N} 2^{B\cdot(\ell-1)} \cdot \delta_{i,\ell,k}. \tag{149}$$

Each equality statement truth value is evaluated properly with respect to the correlated $\vec{\delta}_i$ advice column. This constraint is expressed by checking that each $\vec{\delta}_i$ advice column decomposes into all zeroes at all and only the rows where $\vec{\epsilon}_i$ is zero, as follows.

For all $i \in [s]$, $\ell \in [N]$, $k \in [r]$,

$$(\delta_{i,\ell,k}, \epsilon_{i,\ell,k}) \in (\vec{b}, \vec{z}), \tag{150}$$

$$\epsilon_{i,k} = \prod_{\ell=1}^{N} \epsilon_{i,\ell,k}. \tag{151}$$

Each $\vec{\delta}_i$ advice column is equal to the differences between the values of the terms in the correlated equality subformula. This constraint is expressed by the following s polynomial constraints.

For all $i \in [s]$,

$$\vec{\delta}_i = \text{eval}(\tau_i) - \text{eval}(\mu_i), \tag{152}$$

where $\tau_i = \mu_i$ is the ith equality subformula in Φ, and eval is a partial function from terms to column vectors, defined as follows:

1. $\text{eval}(0) = 0^{[r]}$, where $0^{[r]}$ denotes the column vector consisting of r zeroes.

2. $\text{eval}(1) = 1^{[r]}$, similarly.

3. If x_j is the ith first-order variable occurring free in Φ, then

$$\text{eval}(x_j) = \vec{x}_i. \tag{153}$$

4. If τ_i is the ith function application subterm occurring in Φ, then

$$\text{eval}(\tau_i) = \vec{h}_i. \tag{154}$$

5. For all terms τ, μ and column vectors $\vec{\alpha}, \vec{\beta}$, if

$$\text{eval}(\tau) = \vec{\alpha} \text{ and} \tag{155}$$

$$\text{eval}(\mu) = \vec{\beta}, \tag{156}$$

then

$$\text{eval}(\tau \cdot \mu) = \vec{\alpha} \cdot \vec{\beta}, \tag{157}$$

where $\cdot$ is the Hadamard (elementwise) product of column vectors, and

$$\text{eval}(\tau + \mu) = \vec{\alpha} + \vec{\beta}. \tag{158}$$

For these constraints to be well defined, it must be the case that $\text{eval}(\tau_i)$ and $\text{eval}(\mu_i)$ are defined for each $i \in [s]$. This can easily be seen to be the case by the construction of eval.

Each function application term input column is equal to the evaluation of the input term. This constraint is expressed by the following $\sum_{i=1}^{t} c_i$ polynomial constraints.

For all $i \in [t]$, let $\tau_1, ..., \tau_{c_i}$ be the input terms of the ith function application subterm appearing in Φ. For each $j \in [c_i]$,

$$\vec{y}_{i,j} = \text{eval}(\tau_j). \tag{159}$$

Each function application term output column is equal to the function applied to the inputs. This constraint is expressed by the following lookup constraints.

Let $i \in [t]$. Suppose that the ith function application subterm head is the jth second-order variable occurring free in Φ. That variable has arity c_i. For all $k \in [r]$,

$$(h_{i,k}, y_{i,1,k}, ..., y_{i,a_i,k}) \in (\vec{f}_j, \vec{a}_{j,1}, ..., \vec{a}_{j,a_i}). \tag{160}$$

Let $i \in [t]$. Suppose that the ith function application subterm head is the variable quantified over by the jth second-order existential quantifier occurring in Φ. That variable has arity c_i. For all $k \in [r]$,

$$(h_{i,k}, y_{i,1,k}, ..., y_{i,a_i,k}) \in (\vec{g}_i, \vec{c}_{i,1}, ..., \vec{c}_{i,a_i}). \tag{161}$$

Each quantifier bound is satisfied. For universal quantifiers, this constraint is satisfied by virtue of the fact that the universally quantified variable values are fixed. For first-order and second-order existential quantifiers, this constraint is expressed by performing the comparisons between the witness values and the bounds using the lexicographical ordering of their respective byte decompositions. This is expressed precisely as follows.

1. **Each truth value in $\vec{\omega}_{i,\ell}$, $\vec{\gamma}_{i,j,\ell}$, and $\vec{\kappa}_{i,\ell}$ is accurate.** That is,

   (a) For all $i \in [e], \ell \in [N], k \in [r]$,
$$(w_{i,\ell,k}, \omega_{i,\ell,k}) \in (\vec{b}, \vec{b}_{\eta_i,\ell}). \tag{162}$$

   (b) For all $i \in [m], j \in [b_i], \ell \in [N], k \in [r]$,
$$(c_{i,j,\ell,k}, \gamma_{i,j,\ell,k}) \in (\vec{b}, \vec{b}_{\lambda_{i,j}}). \tag{163}$$

   (c) For all $i \in [m], \ell \in [N], k \in [r]$,
$$(g_{i,j,\ell,k}, \kappa_{i,\ell,k}) \in (\vec{b}, \vec{b}_{\eta_{i,\ell}}). \tag{164}$$

2. **The byte decompositions of $\vec{w}_i, \vec{c}_{i,j}$ and $\vec{g}_i$ are less than the bounds in lexicographical ordering, as indicated by the truth values $\vec{\omega}_{i,\ell}$, $\vec{\gamma}_{i,j,\ell}$, and $\vec{\kappa}_{i,\ell}$.** That is,

   (a) For all $i \in [e], k \in [r]$,
$$\text{Lex}_{<}(\vec{w}_{i,-,k}, \vec{b}_{\eta_i,-}) = 1. \tag{165}$$

   The function
$$\text{Lex}_{<} : [2]^N \times [2]^N \to [2] \tag{166}$$
   is defined by the following recursive clauses.
$$\text{Lex}_{<,1}(\vec{p}, \vec{q}) = p_N \tag{167}$$
$$\text{Lex}_{<,i+1}(\vec{p}, \vec{q}) = q_{N-i} \wedge (p_{N-i} \vee \text{Lex}_i(\vec{p}, \vec{q})) \tag{168}$$
$$\text{Lex}_{<} = \text{Lex}_{<,N} \tag{169}$$

   In 168, the logical connectives $\wedge$ and $\vee$ are interpreted as the arithmetized boolean operations:
$$x \wedge y = x \cdot y \tag{170}$$
$$x \vee y = x + y - x \cdot y \tag{171}$$

   The vectors
$$\vec{w}_{i,-,k}, \vec{b}_{\eta_i,-} \in [2]^N \tag{172}$$
   are defined as:
$$\vec{w}_{i,-,k} = \langle w_{i,1,k}, ..., w_{i,N,k} \rangle \tag{173}$$
$$\vec{b}_{\eta_i,-} = \langle b_{\eta_i,1}, ..., b_{\eta_i,N} \rangle \tag{174}$$

   This expresses a set of polynomial constraints.

   The remaining constraints are expressed using these same notations.

   (b) For all $i \in [m], j \in [b_i], k \in [r]$,
$$\text{Lex}(\vec{c}_{i,j,-,k}, \vec{b}_{\lambda_i,j}) = 1. \tag{175}$$

   (c) For all $i \in [m], k \in [r]$,
$$\text{Lex}(\vec{g}_{i,j,-,k}, \vec{\kappa}_{i,-,k}) = 1. \tag{176}$$

The statement Ψ is true at each row. This constraint is expressed as follows. Extend the eval partial function from terms to column vectors to a partial function from formulas to column vectors, using the following recursive clauses.

1. For all terms τ, μ and column vectors $\vec{\alpha}, \vec{\beta}$, if

$$\text{eval}(\tau) = \vec{\alpha} \text{ and} \tag{177}$$

$$\text{eval}(\mu) = \vec{\beta} \tag{178}$$

then

$$\text{eval}(\tau = \mu) = \iota_0(\vec{\alpha} - \vec{\beta}), \tag{179}$$

where ι_0 is the zero indicator function defined on scalars by

$$\iota_0(x) = \begin{cases} 1 & \text{if } x = 0, \\ 0 & \text{otherwise,} \end{cases} \tag{180}$$

and extended to vectors by applying it pointwise.

2. For all quantifier-free formulas ϕ, if eval(ϕ) is defined then

$$\text{eval}(\neg\phi) = 1 - \text{eval}(\phi). \tag{181}$$

3. For all quantifier-free formulas ϕ, ψ, if eval(ϕ) is defined and eval(ψ) is defined then:

$$\text{eval}(\phi \wedge \psi) = \text{eval}(\phi) \cdot \text{eval}(\psi) \tag{182}$$

$$\text{eval}(\phi \vee \psi) = \text{eval}(\phi) + \text{eval}(\psi) - \text{eval}(\phi) \cdot \text{eval}(\psi) \tag{183}$$

The values in first-order existential quantifier advice columns are equal across different rows when required for consistency of the description of the witness to Φ according to the quantifier ordering. In other words, for any two rows, if all universally quantified variable values in the row are equal for all universal quantifiers preceding the ith first-order existential quantifier, then the values for the ith first-order existentially quantifier variable must be equal in those two rows. This is expressed precisely as follows.

Let

$$\Delta = \Delta_1 < \beta_1. \ \cdots \ \Delta_n < \beta_\Pi. \tag{184}$$

be the first-order quantifier string of Φ (which is assumed to be in prenex normal form). $\Delta_i \in \{\forall, \exists\}$ for each $i \in [\Pi]$. Also, each β_i (for $i \in [\Pi]$) is assumed to be a positive constant term denoting a value less than 2^W. (For any given Φ, there is some choice of W which makes these assumptions true.)

Let

$$\iota : [e] \to [\Pi] \tag{185}$$

be the function defined by the following recursive clauses:

$$\begin{array}{rcl} \iota(1) & = & \text{the least } i \text{ such that } \Delta_i = \exists \\ \iota(j+1) & = & \text{the least } i \text{ such that } \Delta_i = \exists \text{ and } i > \iota(j) \end{array} \tag{186}$$

The fact that this function is total follows from the definition of e as the number of first-order existential quantifiers in Φ.

Similarly, let

$$\zeta : [u] \to [\Pi] \tag{187}$$

be the function defined by the following recursive clauses:

$$\begin{array}{rcl} \zeta(1) & = & \text{the least } i \text{ such that } \Delta_i = \forall \\ \zeta(j+1) & = & \text{the least } i \text{ such that } \Delta_i = \forall \text{ and } i > \zeta(j) \end{array} \tag{188}$$

The fact that this function is total follows from the definition of u as the number of first-order universal quantifiers in Φ.

For all $i \in [e]$ and $k, k' \in [r]$, if k and k' are i-compatible then

$$w_{i,k} = w_{i,k'}, \tag{189}$$

where k and k' are said to be i-compatible iff for all $j \in [u]$, if $\zeta(j) < \iota(i)$ then $v_{j,k} = v_{j,k'}$.

The values in any first-order variable value instance column are all equal to each other. In other words, for all $i \in [k]$ and $j, j' \in [r]$,

$$x_{i,j} = x_{i,j'}. \tag{190}$$

The lookup tables for free and existentially quantified second-order variables each define a function. This is checked using the following constraints.

1. **The truth values of inequality statements about bytes of deltas of elements of adjacent input row vectors are correct.** The truth values checked by these constraints are $\vec{\tau}'_{i,j,\ell}$ for all $i \in [n], j \in [a_i]$, and $\ell \in [N]$, and $\vec{\upsilon}'_{i,j,\ell}$ for all $i \in [m], j \in [b_i]$, and $\ell \in [N]$. The constraints are as follows.

   (a) For all $i \in [n]$, $j \in [a_i]$, $\ell \in [N]$, and $k \in [r]$,

   $$(\sigma_{i,j,\ell,k}, 1 - \tau'_{i,j,\ell,k}) \in (\vec{b}, \vec{z}). \tag{191}$$

   (b) For all $i \in [m], j \in [b_i]$, and $\ell \in [N]$,

   $$(\rho_{i,j,\ell,k}, 1 - \upsilon'_{i,j,\ell,k}) \in (\vec{b}, \vec{z}). \tag{192}$$

2. **The truth values of inequality statements about deltas of elements of adjacent input row vectors are correct.** The truth values checked by these constraints are $\vec{\tau}_{i,j,\pi}$ for all $i \in [n], j \in [a_i]$, and $\pi \in \{0, 1\}$, and $\vec{\upsilon}_{i,j,\pi}$ for all $i \in [m], j \in [b_i]$, and $\pi \in \{0, 1\}$. The constraints are as follows.

   (a) For all $i \in [n], j \in [a_i], k \in [r]$,

   $$\tau_{i,j,0,k} = \frac{1}{2}\left(\sigma^{+/-}_{i,j,k} + 1\right). \tag{193}$$

   $\frac{1}{2}$ denotes the multiplicative inverse of two in $\mathbb{F}$.

   (b) For all $i \in [m], jin[b_i], k \in [r]$,

   $$\upsilon_{i,j,0,k} = \frac{1}{2}\left(\rho^{+/-}_{i,j,k}\right). \tag{194}$$

   (c) For all $i \in [n], j \in [a_i], k \in [r]$,

   $$\left(\tau_{i,j,0,k}, \tau_{i,j,1,k}, \left(\sum_{\ell=1}^{N} \sigma_{i,j,\ell,k}\right)\right) \in (\vec{\psi}_1, \vec{\psi}_2, \vec{\psi}_3). \tag{195}$$

(d) For all $i \in [m], j \in [b_i], k \in [r]$,

$$\left(\upsilon_{i,j,0,k}, \upsilon_{i,j,1,k}, \left(\sum_{\ell=1}^{N} \upsilon_{i,j,\ell,k}\right)\right) \in (\vec{\psi}_1, \vec{\psi}_2, \vec{\psi}_3 + 1). \tag{196}$$

3. **The rows of each lookup table are lexicographically ordered.** This is expressed precisely as follows.

(a) For all $i \in [n], k \in [r]$,

$$\mathrm{Lex}_{\leq}(\vec{\tau}_{i,-,1,k}, \vec{\tau}_{i,-,0,k}) = 1. \tag{197}$$

The function

$$\mathrm{Lex}_{\leq} : [2]^{a_i} \to [2] \tag{198}$$

is defined by the following recursive clauses:

$$\mathrm{Lex}_{\leq,1}(\vec{p}, \vec{q}) = q_{a_i} \tag{199}$$

$$\mathrm{Lex}_{\leq,j+1}(\vec{p}, \vec{q}) = q_{a_i-j} \wedge (p_{a_i-j} \vee \mathrm{Lex}_i(\vec{p}, \vec{q})) \tag{200}$$

$$\mathrm{Lex}_{\leq} = \mathrm{Lex}_{\leq,a_i} \tag{201}$$

The vectors

$$\vec{\tau}_{i,-,0,k}, \vec{\tau}_{i,-,1,k} \in [2]^N \tag{202}$$

are defined as

$$\vec{\tau}_{i,-,\pi,k} = \langle \tau_{i,1,\pi,k}, ..., \tau_{i,N,\pi,k} \rangle, \tag{203}$$

for all $\pi \in \{0, 1\}$. The next constraint is expressed using these same notations, with appropriate variable substitutions.

(b) For all $i \in [m], k \in [r]$,

$$\mathrm{Lex}_{\leq}(\vec{\upsilon}_{i,-,1,k}, \vec{\upsilon}_{i,-,0,k}) = 1. \tag{204}$$

4. **In adjacent rows with equal input column values, the output column values are the same.** This is expressed precisely as follows.

(a) For all $i \in [n], k \in [r]$,

$$0 = \left(\prod_{j=1}^{a_i} (1 - \tau_{i,j,1,k})\right) \cdot (f_{i,k} - f_{i,k+1}). \tag{205}$$

(b) For all $i \in [m], k \in [r]$,

$$0 = \left(\prod_{j=1}^{b_i} (1 - \upsilon_{i,j,1,k})\right) \cdot (g_{i,k} - g_{i,k+1}). \tag{206}$$

In these formulas, the subscripts $k + 1$ are computed with wraparound modulo r.

4.3 The circuit itself

In order to assemble the foregoing information into a circuit, some additional information needs to be provided. The information provided thus far is the column types, the constraints, and the fixed column values. The unspecified information thus far is the underlying finite field $\mathbb{F}$, the polynomial degree bound d, and the row count r.

Beyond the constraints imposed by the proving system itself, the only constraint on $\mathbb{F}$ is that its order must be greater than $2M$, where M is the maximum absolute value of an intermediate result obtained in an evaluation of the truth value of Φ in some integral model wherein the values interpreting free variables are integers in the range $[0, 2^W)$ (for first-order variables) and functions in $[0, 2^W)^{[0,2^W)^n}$ (for n-ary function variables).

M is efficiently computable from Φ using the quantifier bounds and the terms of Φ. This is because all intermediate results are obtained from calculations using only addition and multiplication of instance values, witness values, and constants, and all instance and witness values are bounded by constants.

The only constraint on d imposed by the construction is that d must be no smaller than each of the degrees of the polynomial constraints. Therefore the minimum value of d is efficiently computable from Φ by generating the polynomial constraints and counting their degrees.

The only constraint on r imposed by the construction is that r must be no less than one and no less than $\prod_{i=1}^{u} \beta_i$, where u is the number of universal quantifiers and β_i is the bound on the ith universal quantifier in Φ.

4.4 Variations on this construction

The polynomial degree bound d can be reduced in some cases by increasing the number of advice columns, by adding advice columns to store intermediate results of calculations of the values of terms and subformulas, resulting in simpler constraints. Increasing the number of columns while decreasing the polynomial degree bound may impact various metrics of efficiency in various ways, and thus some experimentation may be helpful to find a variation on this construction which is more optimized in general or for a particular formula.

5 Proof sketch of correctness of the arithmetization

What it means for this arithmetization to be correct is that the resulting circuit arithmetizes the relation defined by the formula Φ. What it means for a circuit to be an arithmetization of a relation is defined in Section 2. In order to define what it means for the arithmetization to be correct, it suffices to define what is the relation defined by a formula Φ.

The "domain of Φ," dom(Φ), is a set, of which the relation defined by Φ is a subset,

$$\mathrm{dom}(\Phi) = [0, 2^W)^k \times \prod_{i=1}^{n} \{x \in \mathcal{P}([0, 2^W)^{a_i} \times [0, 2^W)) \mid x \text{ is a function and } |x| \leq r\}, \tag{207}$$

where $\mathcal{P}$ is the power set operator, $|x|$ is the cardinality of x, k is the number of first-order variables free in Φ, n is the number of second-order variables free in Φ, and a_i is the arity of the ith second-order variable occurring free in Φ. An element of dom(Φ) contains a value for each free variable in Φ.

Each $x \in \mathrm{dom}(\Phi)$ yields an integral model, M_x, defined as

$$M_x = (\mathbb{Z}, \cdot, +, 1, -1, F, S), \tag{208}$$

where

$$F(i) = \begin{cases} \pi_i(\pi_1(x)) & \text{if } i \in [1,k], \\ 0 & \text{otherwise.} \end{cases} \tag{209}$$

$$S(i) = \begin{cases} \pi_i(\pi_2(x)) & \text{if } i \in [1,n], \\ \text{const}(0) & \text{otherwise.} \end{cases} \tag{210}$$

The relation defined by Φ is the set

$$\{x \in \text{dom}(\Phi) \mid \delta_{M_x}(\Phi) = 1\}. \tag{211}$$

Let Φ be a formula. Let $\mathcal{C}$ be the circuit arithmetizing Φ as defined in Section 4. Let $f : \text{dom}(\Phi) \to \mathbb{F}^{v \times r}$, where v is the number of instance columns of $\mathcal{C}$ and r is the row count, be defined as

$$f(x) = \begin{bmatrix} x_1 & \cdots & x_k & y_{1,1,1} & \cdots & y_{1,a_1,1} & z_{1,1} & \cdots & y_{n,1,1} & \cdots & y_{n,a_n,1} & z_{n,1} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & . & \vdots & \vdots \\ x_1 & \cdots & x_k & y_{1,1,r} & \cdots & y_{1,a_1,r} & z_{1,r} & \cdots & y_{n,1,r} & \cdots & y_{n,a_n,r} & z_{n,r} \end{bmatrix} \tag{212}$$

where

$$x_i = h(\pi_i(\pi_1(x))), \tag{213}$$

$$y_{i,j,\ell} = h(\pi_j(\pi_1(e_\ell(\pi_i(\pi_2(x)))))), \tag{214}$$

$$z_{i,\ell} = h(\pi_2(e_\ell(\pi_i(\pi_2(x))))). \tag{215}$$

The function e_i goes from a well-ordered finite set to the ith element of the set in the implied well-ordering, or if i is greater than the cardinality of the input set, then e_i returns the last element of the set.

The function $h : \mathbb{Z} \to \mathbb{F}$ is the unique ring homomorphism from $\mathbb{Z}$ to $\mathbb{F}$. The unique existence of h follows from the fact that $\mathbb{Z}$ is the zero object in the category of rings.

This is the core correctness claim:

Theorem 1. *$(\mathcal{C}, f)$ is an arithmetization of the relation defined by Φ.*

The strategy for proving the correctness claim for the construction of $\mathcal{C}$ is to first prove the correctness claim for a different construction which is less efficient but simpler to prove the claim for, and then proving that the construction of $\mathcal{C}$ is equivalent to the simpler construction. The simpler construction is described in category theory terms, using the operations of a Cartesian closed category, namely products, coproducts, and exponentials.

5.1 Categories of circuits

In order to define a category of circuits usable for the present purposes, it is necessary to redefine circuits with some additional data and assumptions. These redefined circuits will be called "composable circuits." Not every circuit can be made into a composable circuit, but all of the circuits resulting from the arithmetization construction can be made into composable circuits.

By definition, a "composable circuit" is a tuple

$$\mathcal{C} = (\mathbb{F}, \vec{c}, d, \vec{p}, \vec{\ell}, r, \vec{e}, X, \tau, f), \tag{216}$$

where:

1. $\mathbb{F}$ is a finite field. $\mathbb{F}$ is called the underlying field.

2. $\vec{c} = c_1, ..., c_n$, where for all $i \in [n]$,

$$c_i \in \{F, D, O, I\} \times \{E, N\}. \tag{217}$$

Here F, D, O, I, E, N are distinct constants with the following interpretations:

F	Fixed column
D	Deterministic advice column
O	Nondeterministic advice column
I	Instance column
E	Equality constrainable column
N	Non equality constrainable column

$\vec{c}$ is called the column type vector.

3. d is a positive integer. d is called the polynomial degree bound.

4. $\vec{p} = p_1, ..., p_m$, where for all $i \in [m]$, p_i is a polynomial of degree $\leq\ d$ with coefficients in $\mathbb{F}$, where variable names are pairs $(j, k) \in [n] \times \mathbb{Z}$ where j is a column index and k is a relative row reference. $k = 0$ refers to the current row, and in general, k refers to the row whose index is the index of the current row plus k (with wraparound, modulo r).

   $\vec{p}$ is called the sequence of polynomial constraints.

5. $\vec{\ell} = \ell_1, ..., \ell_k$, where for all $i \in [k]$, ℓ_i is a tuple of the form $(\vec{x}, \vec{a})$ where for some positive integer q:

   (a) $\vec{x} = x_1, ..., x_q$, where for all $j \in [q]$, x_j is a polynomial of degree $\leq\ d$ with coefficients in $\mathbb{F}$, where variables are pairs $(k, l) \in [n] \times \mathbb{Z}$ where k is a column index and l is a relative row reference. $\vec{x}$ is called the input expression.

   (b) $\vec{a} = a_1, ..., a_q$, where for all $j \in [q]$, a_j is a column index ($a_j \in [n]$). $\vec{a}$ is called the vector of table columns.

   $\vec{\ell}$ is called the sequence of lookup arguments.

6. r is a positive integer. r is called the number of rows.

7. $\vec{e} = e_1, ..., e_t$, where for all $i \in [t]$, $e_i \in ([n] \times [r])^2$ is a pair of pairs of indices, representing an equality constraint in the form of a pair of absolute cell references.

8. $X \in \mathbb{F}^{[u] \times [r]}$, where u is the number of fixed columns, contains the values of each fixed column in each row.

9. $\tau \in [n]$, where $\pi_1(c_\tau) = D$. τ is the index of the column containing all ones when the statement expressed by the circuit is true, and otherwise contains some combination of ones and/or zeroes.

10. $f : \mathbb{F}^{[n_I + n_O] \times [r]} \to \mathbb{F}^{[n_D] \times [r]}$ is a function, where for all $X \in \{I, O, D\}$,

$$n_X = |\{i \in [n] \mid c_i = X\}|. \tag{218}$$

   f is called the deterministic advice computer.

11. For all $Y \in \mathbb{F}^{[n_I + n_O] \times [r]}$, if there exists $Z \in \mathbb{F}^{[n_D] \times [r]}$ such that $[XYZ]$ satisfies $\mathcal{C}$, then $[XYf(Y)]$ satisfies $\mathcal{C}$, and $f(Y)$ is the unique such Z. (Satisfaction of a composable circuit is defined the same way as satisfaction of a circuit.)

12. $\mathcal{C}$ includes the following constraints. In these constraints, $\vec{t}$ denotes the value of the τth column.
    (a) $\vec{t} = 1^r$.
    (b) For all $i \in [r]$, $t_i \in \{0, 1\}$.
13. If the constraint $\vec{t} = 1^r$ is removed from $\mathcal{C}$, the resulting circuit $\mathcal{D}$ is uniquely satisfiable for all combinations of instance and nondeterministic advice values. In other words, for all $Y \in \mathbb{F}^{[n_I+n_O]\times[r]}$, $[XYf(Y)]$ satisfies $\mathcal{D}$, and $f(Y)$ is the unique Z such that $[XYZ]$ satisfies $\mathcal{D}$.

For any finite field $\mathbb{F}$, positive integer $r \geq |\mathbb{F}|+1$,[5] and non-negative integer n_I, let $\mathbf{Circ}(\mathbb{F}, n_I, r)$ denote a category. The objects are composable circuits with field $\mathbb{F}$, row count r, and n_I instance columns. A morphism $g : \mathcal{C} \to \mathcal{D}$ is a function

$$g : \mathbb{F}^{[n_I+n_{\mathcal{C},O}]\times[r]} \to \mathbb{F}^{[n_{\mathcal{D},O}]\times[r]}, \tag{219}$$

where $n_{\mathcal{X},O}$ denotes the number of nondeterministic advice columns of the circuit $\mathcal{X}$, such that for all $X \in \mathbb{F}^{[n_I]\times[r]}$ and $Y \in \mathbb{F}^{[n_{\mathcal{C},O}]\times[r]}$, if $[XYf_{\mathcal{C}}([XY])]$ satisfies $\mathcal{C}$, then $[Xg([XY])f_{\mathcal{D}}([Xg([XY])])]$ satisfies $\mathcal{D}$. Here $f_{\mathcal{X}}$ denotes the deterministic advice computer of $\mathcal{X}$.

Equality of morphisms is the maximal equality relation, meaning that for any objects $\mathcal{C}, \mathcal{D}$ and morphisms $f, g \in \hom(\mathcal{C}, \mathcal{D})$, $f = g$. This can be contrasted with the extensional equality relation, where things are equal only to themselves. In the categories $\mathbf{Circ}(\mathbb{F}, n_I, r)$, all elements of a hom-set are considered formally equal. This renders all morphism equality assertions in category theoretic properties trivially true. This approach works because for the present purposes, the distinctions between morphisms and the characteristics of hom-sets are not relevant. What is relevant is whether a hom-set is empty or nonempty, because this determines whether or not one circuit implies another when the circuits are considered as propositions.

Given morphisms $f : \mathcal{C} \to \mathcal{D}$ and $g : \mathcal{D} \to \mathcal{E}$, the morphism $g \circ f$ is defined, for all $X \in \mathbb{F}^{[n_I]\times[r]}$ and $Y \in \mathbb{F}^{[n_{\mathcal{C},O}]\times[r]}$, by

$$(g \circ f)([XY]) = g([Xf([XY])]). \tag{220}$$

It is straightforward to check that $g \circ f$ is a morphism. The composition law is made trivial by the maximal equality relation, but also holds true for extensional equality.

Given an object $\mathcal{C}$, the morphism $\mathrm{id}_{\mathcal{C}}$ is defined, for all $X \in \mathbb{F}^{[n_I]\times[r]}$ and $Y \in \mathbb{F}^{[n_{\mathcal{C},O}]\times[r]}$, by

$$\mathrm{id}_{\mathcal{C}}([XY]) = Y. \tag{221}$$

It is straightforward to check that this is a morphism. The composition identity laws are made trivial by the maximal equality relation, but also hold true for extensional equality.

5.1.1 Initial objects ($\bot$)

Let $\mathbf{Circ}(\mathbb{F}, n_I, r)$ be a category of composable circuits. In this category, find a composable circuit $\bot$ with one deterministic advice column (denoted $\vec{t}$), no other advice columns, and the following constraints:

1. $\vec{t} = 1^r$.
2. $\vec{t} = 0^r$.
3. For all $i \in [r]$, $t_i \in \{0, 1\}$.

[5]The condition $r \geq |\mathbb{F} + 1|$ is required for the lookup arguments in the coproduct construction.

The deterministic advice computer is $\text{const}(0^r)$. Since this circuit is never satisfied, condition 11 of the composable circuit definition is satisfied trivially. Conditions 12 and 13 are also satisfied trivially.

Theorem 2. *$\perp$ is an initial object.*

Proof. Let $\mathcal{C}$ be a circuit. Define a morphism $f : \perp \to \mathcal{C}$ by

$$f = \text{const}(0^{[n_{\mathcal{C},O}]\times[r]}). \tag{222}$$

This is trivially a morphism because $\perp$ is never satisfied. It is trivially unique because of the maximal equality relation.

5.1.2 Products ($\times$)

Let $\mathbf{Circ}(\mathbb{F}, n_I, r)$ be a category of composable circuits. Let $\mathcal{C}$ and $\mathcal{D}$ be objects in this category. In this category, find a circuit $\mathcal{C} \times \mathcal{D}$ with the following characteristics:

1. $\mathcal{C} \times \mathcal{D}$ has the following numbers of deterministic and nondeterministic advice columns:

$$n_{\mathcal{C}\times\mathcal{D},O} = n_{\mathcal{C},O} + n_{\mathcal{D},O} \tag{223}$$

$$n_{\mathcal{C}\times\mathcal{D},D} = n_{\mathcal{C},D} + n_{\mathcal{D},D} + 1 \tag{224}$$

   All but one of these advice columns correspond to those in $\mathcal{C}$ and $\mathcal{D}$. The one deterministic advice column beyond those in $\mathcal{C}$ and $\mathcal{D}$ is the truth value column, denoted $\vec{t}$. In addition, the truth value columns of $\mathcal{C}$ and $\mathcal{D}$ are present, and denoted $\vec{u}$ and $\vec{v}$, respectively.

2. All of the constraints of $\mathcal{C}$ and $\mathcal{D}$ are present, except for $\vec{u} = 1^r$ and $\vec{v} = 1^r$.

3. The following constraints are present:

   (a) For all $i \in [r]$, $t_i \in \{0, 1\}$.
   (b) $\vec{t} = 1^r$.
   (c) $\vec{t} = \vec{u} \cdot \vec{r}$, where $\cdot$ is the Hadamard product.

4. The deterministic advice computer, $f_{\mathcal{C}\times\mathcal{D}}$, is defined as follows, in terms of the deterministic advice computers $f_{\mathcal{C}}$ of $\mathcal{C}$ and $f_{\mathcal{D}}$ of $\mathcal{D}$. For all $X \in \mathbb{F}^{[n_I]\times[r]}$, $Y \in \mathbb{F}^{[n_{\mathcal{C},O}]\times[r]}$, $Z \in \mathbb{F}^{[n_{\mathcal{D},O}]\times[r]}$,

$$f_{\mathcal{C}\times\mathcal{D}}([XYZ]) = [f_{\mathcal{C}}([XY]) f_{\mathcal{D}}([XZ])]. \tag{225}$$

Define morphisms

$$\pi_1 : (\mathcal{C} \times \mathcal{D}) \to \mathcal{C}, \tag{226}$$

$$\pi_2 : (\mathcal{C} \times \mathcal{D}) \to \mathcal{D}, \tag{227}$$

as follows. For all $X \in \mathbb{F}^{[n_I]\times[r]}$, $Y \in \mathbb{F}^{[n_{\mathcal{C},O}]\times[r]}$, and $Z \in \mathbb{F}^{[n_{\mathcal{D},O}]\times[r]}$,

$$\pi_1([XYZ]) = Y, \tag{228}$$

$$\pi_2([XYZ]) = Z. \tag{229}$$

It is straightforward to check that these are morphisms.

Theorem 3. *$\mathcal{C} \times \mathcal{D}$, π_1, and π_2 are a product of $\mathcal{C}$ and $\mathcal{D}$.*

Proof. Let $\mathcal{X}$ be an object. Let $f : \mathcal{X} \to \mathcal{C}$ and $g : \mathcal{X} \to \mathcal{D}$ be morphisms. Define the morphism $f \times g : \mathcal{X} \to (\mathcal{C} \times \mathcal{D})$ as follows. For all $X \in \mathbb{F}^{[n_I]\times[r]}$ and $Y \in \mathbb{F}^{[n_{\mathcal{C},X}]\times[r]}$,

$$(f \times g)([XY]) = [f([XY])g([XY])]. \tag{230}$$

It is straightforward to check that this is a morphism. The diagram

$$\begin{tikzcd} & \mathcal{X} \arrow[dl, "f"'] \arrow[d, "f\times g"] \arrow[dr, "g"] & \\ \mathcal{C} & \mathcal{C} \times \mathcal{D} \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & \mathcal{D} \end{tikzcd} \tag{231}$$

commutes trivially in that all diagrams commute in a category with maximal morphism equality. For the same reason, $f \times g$ is the unique morphism (up to the equality relation) making this diagram commute.

5.1.3 Coproducts ($\oplus$)

Let $\mathbf{Circ}(\mathbb{F}, n_I, r)$ be a category of composable circuits. Let $\mathcal{C}$ and $\mathcal{D}$ be objects in this category. Without loss of generality, assume that $\mathcal{C}$ and $\mathcal{D}$ have the same numbers of nondeterministic advice columns, i.e. $n_{\mathcal{C},O} = n_{\mathcal{D},O}$. This is without loss of generality because if $n_{\mathcal{C},O} \neq n_{\mathcal{D},O}$, then we can pad one of the circuits with extra unused nondeterministic advice columns, resulting in an isomorphic circuit. Let $n_O = n_{\mathcal{C},O} = n_{\mathcal{D},O}$.

In this category, find a circuit $\mathcal{C} \oplus \mathcal{D}$ with the following characteristics:

1. $\mathcal{C} \oplus \mathcal{D}$ has the following numbers of advice columns:
$$n_{\mathcal{C}\oplus\mathcal{D},O} = n_O + 1 \tag{232}$$
$$n_{\mathcal{C}\oplus\mathcal{D},D} = n_{\mathcal{C},D} + n_{\mathcal{D},D} + 3 \tag{233}$$
The nondeterministic advice columns of $\mathcal{C}$ and $\mathcal{D}$ are overlapped, or in other words they share nondeterministic advice. There is an additional nondeterministic advice column, denoted $\vec{s}$, which selects which of the two circuits is applied at each row. The deterministic advice columns of $\mathcal{C}$ and $\mathcal{D}$ are present and separate. Their truth value columns are present and denoted $\vec{u}$ and $\vec{v}$, respectively. There are three additional deterministic advice columns. These are the truth value column, denoted $\vec{t}$, the $s_i \in \{0, 1\}$ truth value column, denoted $\vec{w}$, and the $s_i = s_{i-1}$ truth value column, denoted $\vec{x}$.

2. All of the constraints of $\mathcal{C}$ and $\mathcal{D}$ are present, except for $\vec{u} = 1^r$ and $\vec{v} = 1^r$.

3. The following constraints are present:
   (a) For all $i \in [r]$, $t_i \in \{0, 1\}$.
   (b) $\vec{t} = 1^r$.
   (c) $\vec{t} = (\vec{s} \cdot \vec{u} + (1^r - \vec{s}) \cdot \vec{v}) \cdot \vec{w} \cdot \vec{x}$, where $\cdot$ is the Hadamard product.
   (d) For all $i \in [r]$, $(s_i, w_i) \in \{(0, 1), (1, 1)\} \cup \{(x, 0) \mid x \in \mathbb{F} \setminus \{0, 1\}\}$.
   (e) For all $i \in [r]$, $(s_i - s_{i-1}, x_i) \in \{(0, 1)\} \cup \{(x, 0) \mid x \in \mathbb{F} \setminus \{0\}\}$.

4. The deterministic advice computer, $f_{\mathcal{C}\oplus\mathcal{D}}$, is defined as follows, in terms of the deterministic advice computers $f_{\mathcal{C}}$ of $\mathcal{C}$ and $f_{\mathcal{D}}$ of $\mathcal{D}$. For all $X \in \mathbb{F}^{[n_I]\times[r]}$, $Y \in \mathbb{F}^{[n_O]\times[r]}$, $\vec{s} \in \mathbb{F}^r$,
$$f_{\mathcal{C}\oplus\mathcal{D}}([XY\vec{s}]) = \begin{cases} [f_{\mathcal{C}}([XY])\vec{v}\vec{w}\vec{x}] & \text{if } s_1 = 0, \\ [f_{\mathcal{D}}([XY])\vec{u}\vec{w}\vec{x}] & \text{otherwise,} \end{cases} \tag{234}$$

where $\vec{v}$ denotes the value of the truth value column in $f_{\mathcal{C}}([XY])$, and $\vec{u}$ denotes the value of the truth value column in $f_{\mathcal{D}}([XY])$, and $\vec{w}$ and $\vec{x}$ are the dependent advice columns defined, for all $i \in [r]$, by

$$w_i = \begin{cases} 1 & \text{if } s_i \in \{0, 1\}, \\ 0 & \text{otherwise,} \end{cases} \tag{235}$$

$$x_i = \begin{cases} 1 & \text{if } s_i = s_{i-1}, \\ 0 & \text{otherwise.} \end{cases} \tag{236}$$

It is straightforward to check that such a circuit exists.

Define morphisms

$$\iota_1 : \mathcal{C} \to (\mathcal{C} \oplus \mathcal{D}) \tag{237}$$

$$\iota_2 : \mathcal{D} \to (\mathcal{D} \oplus \mathcal{D}) \tag{238}$$

as follows. For all $X \in \mathbb{F}^{[n_I] \times [r]}$ and $Y \in \mathbb{F}^{[n_O] \times [r]}$,

$$\iota_1([XY]) = [Y0^r], \tag{239}$$

$$\iota_2([XY]) = [Y1^r]. \tag{240}$$

It is straightforward to check that these are morphisms.

Theorem 4. *$\mathcal{C} \oplus \mathcal{D}$, ι_1, and ι_2 are a coproduct of $\mathcal{C}$ and $\mathcal{D}$.*

Proof. Let $\mathcal{X}$ be an object. Let $f : \mathcal{C} \to \mathcal{X}$ and $g : \mathcal{D} \to \mathcal{X}$ be morphisms. Define the morphism $f \oplus g : (\mathcal{C} \oplus \mathcal{D}) \to \mathcal{X}$ as follows. For all $X \in \mathbb{F}^{[n_I] \times [r]}$ and $Y \in \mathbb{F}^{[n_O] \times [r]}$ and $\vec{s} \in \mathbb{F}^r$,

$$(f \oplus g)([XY\vec{s}]) = \begin{cases} f([XY]) & \text{if } \vec{s} = 0^r, \\ g([XY]) & \text{otherwise.} \end{cases} \tag{241}$$

It is straightforward to check that this is a morphism. The diagram

$$\begin{tikzcd} & \mathcal{X} & \\ \mathcal{C} \arrow[ur, "f"] \arrow[r, "\iota_1"] & \mathcal{C} \oplus \mathcal{D} \arrow[u, "f \oplus g"] & \mathcal{D} \arrow[ul, "g"'] \arrow[l, "\iota_2"] \end{tikzcd} \tag{242}$$

commutes because morphism equality is maximal, and $f \oplus g$ is unique for the same reason.

5.1.4 Negations ($\perp^{\mathcal{C}}$)

Let $\mathbf{Circ}(\mathbb{F}, n_I, r)$ be a category of composable circuits. Let $\mathcal{C}$ be an object of this category. Find an object $\perp^{\mathcal{C}}$ with the following characteristics.

1. The numbers of advice columns are as follows:

$$n_{\perp^{\mathcal{C}},O} = n_{\mathcal{C},O} \tag{243}$$

$$n_{\perp^{\mathcal{C}},D} = n_{\mathcal{C},D} + 1 \tag{244}$$

   The advice columns of $\mathcal{C}$ are present. The additional deterministic advice column, denoted $\vec{t}$, is the truth value column. The truth value column of $\mathcal{C}$ is present and denoted $\vec{u}$.

2. All of the constraints of $\mathcal{C}$ are present except for $\vec{u} = 1^r$.

3. The following constraints are present:

   (a) For all $i \in [r]$, $t_i \in \{0, 1\}$.

   (b) $\vec{t} = 1^r - \vec{u}$.

4. The deterministic advice computer, $f_{\perp^{\mathcal{C}}}$, is defined in terms of the deterministic advice computer $f_{\mathcal{C}}$ of $\mathcal{C}$, as follows. For all $X \in \mathbb{F}^{[n_I + n_{\mathcal{C},O}] \times [r]}$,

$$f_{\perp^{\mathcal{C}}}(X) = [f_{\mathcal{C}}(X)(1^r - \vec{u})], \tag{245}$$

   where $\vec{u}$ denotes the value of the truth value column in $f_{\mathcal{C}}(X)$.

$\perp^{\mathcal{C}}$ is a circuit which is satisfied exactly in the cases when $\mathcal{C}$ is not satisfied.

It is unclear if this negation construction can be characterized by a universal property. A first thought is that it would be an exponential object with domain $\mathcal{C}$ and codomain $\perp$. However, this thought doesn't seem to pan out. The morphism

$$\mathrm{eval} : (\perp^{\mathcal{C}} \times \mathcal{C}) \to \perp \tag{246}$$

required for an exponential object would necessarily be extensionally equal to const(()). Here () denotes the empty tuple $() \in \mathbb{F}^0$, since $\perp$ has no nondeterministic advice. For this to be a morphism, it would have to be the case that $\perp^{\mathcal{C}} \times \mathcal{C}$ is not satisfiable. If $\mathcal{C}$ has no nondeterministic advice, then $\perp^{\mathcal{C}} \times \mathcal{C}$ is not satisfiable and eval is a morphism, and the universal property of exponential objects holds. This is not necessarily the case if $\mathcal{C}$ has nondeterministic advice, however. In nondeterministic advice for $\perp^{\mathcal{C}} \times \mathcal{C}$, $\perp^{\mathcal{C}}$ and $\mathcal{C}$ may receive different nondeterministic advice, making them both true at the same time.

5.2 Functorial arithmetization

The categories of composable arithmetic circuits yield a simple arithmetization of Σ^1_1 formulas which can be described as a functor from the category of Σ^1_1 formulas to a category of composable circuits.

The first step in the arithmetization is to eliminate quantifiers. This can be done because all quantifiers are bounded. Quantifier elimination is defined over all Σ^1_1 formulas by the following recursive clauses.

$$\mathrm{elim}(\tau_1 = \tau_2) = (\tau_1 = \tau_2) \tag{247}$$

$$\mathrm{elim}(\phi \wedge \psi) = \mathrm{elim}(\phi) \wedge \mathrm{elim}(\psi) \tag{248}$$

$$\mathrm{elim}(\phi \vee \psi) = \mathrm{elim}(\phi) \vee \mathrm{elim}(\psi) \tag{249}$$

$$\mathrm{elim}(\neg\phi) = \neg\mathrm{elim}(\phi) \tag{250}$$

$$\mathrm{elim}(\exists < 0.\ \phi) = \perp \tag{251}$$

$$\mathrm{elim}(\exists < (\beta + 1).\ \phi) = \mathrm{elim}(\phi[x_1 \mapsto \beta]) \vee \mathrm{elim}(\exists < \beta.\ \phi) \tag{252}$$

$$\mathrm{elim}(\forall < 0.\ \phi) = \neg\perp \tag{253}$$

$$\mathrm{elim}(\forall < (\beta + 1).\ \phi) = \mathrm{elim}(\phi[x_1 \mapsto \beta]) \wedge \mathrm{elim}(\forall < \beta.\ \phi) \tag{254}$$

$$\mathrm{elim}(\exists_f < \gamma(< \beta_1, ..., < \beta_n).\ \phi) = \bigvee_{f \in [\gamma]^{[\beta_1] \times \cdots \times [\beta_n]}} \mathrm{elim}(\phi[f_1 \mapsto f]) \tag{255}$$

The notation $\phi[f_1 \mapsto f]$ is defined by

$$\phi[f_1 \mapsto f] = \bigvee_{\vec{x} \in [\beta_1] \times \cdots \times [\beta_n]} \left(\bigwedge_{i \in [n]} \tau_i = x_i \right) \wedge \phi[f_1(\tau_1, ..., \tau_n) \mapsto f(\vec{x})], \tag{256}$$

where $\phi[f_1(\tau_1, ..., \tau_n) \mapsto f(\vec{x})]$ denotes the formula resulting from replacing all occurrences of the term $f_1(\tau_1, ..., \tau_n)$ in ϕ with the constant $f(\vec{x})$, and $f_1(\tau_1, ..., \tau_n)$ ranges over all function application terms occurring in ϕ which have f_1 as the function application head, so that potentially many substitutions are implied by this notation.

It is straightforward to check that quantifier elimination is semantics preserving, meaning for all models M, $\delta_M(\phi) = \delta_M(\text{elim}(\phi))$. This means in other words that ϕ and $\text{elim}(\phi)$ define the same relation. Therefore, any arithmetization of the relation defined by $\text{elim}(\phi)$ is an arithmetization of the relation defined by ϕ.

The codomain of elim is the set of quantifier-free second-order formulas over the language of rings. These formulas may contain free first-order and second-order variables. Because there are no quantifiers, all variables are free. Call this set QF. Call the elements of this set QF formulas.

QF formulas form a category, where the objects are all QF formulas, and there is at most one morphism in each hom-set, and there is a morphism in $\text{hom}(\phi, \psi)$ iff for all models M, $\delta_M(\phi) \leq \delta_M(\psi)$ (or in other words, ϕ logically entails ψ). Refer to this category as QF.

QF has all products and coproducts. $\phi \wedge \psi$ is the product of ϕ and ψ. $\phi \vee \psi$ is the coproduct of ϕ and ψ. The universal properties of products and coproducts correspond to the conjunction and disjunction introduction and elimination rules of logic, which are as follows:

$$\frac{\phi, \psi}{\phi \wedge \psi} \quad \frac{\phi \wedge \psi}{\phi} \quad \frac{\phi \wedge \psi}{\psi} \tag{257}$$

$$\frac{\phi}{\phi \vee \psi} \quad \frac{\psi}{\phi \vee \psi} \quad \frac{\phi \vdash \zeta, \psi \vdash \zeta}{\phi \wedge \psi \vdash \zeta} \tag{258}$$

In these rules, $\frac{A}{B}$ means that the set of premises A entail the conclusion B, and the premise $A \vdash B$ means that the set of premises A entails the conclusion B.

These rules can be phrased differently, as statements about morphism existence in QF:

1. If there are morphisms $\zeta \to \phi$ and $\zeta \to \psi$, then there is a morphism $\zeta \to (\phi \wedge \psi)$.
2. There are morphisms $(\phi \wedge \psi) \to \phi$ and $(\phi \wedge \psi) \to \psi$.
3. There are morphisms $\phi \to (\phi \vee \psi)$ and $\psi \to (\phi \vee \psi)$.
4. If there are morphisms $\phi \to \zeta$ and $\psi \to \zeta$, then there is a morphism $(\phi \vee \psi) \to \zeta$.

These rules may be quickly recognized as the universal properties of product and coproduct as applied to QF.

It is also straightforward to see that $\bot$ is the initial object of QF, and $\neg\bot$ is the terminal object. $\bot$ being the initial object corresponds to the following rule:

$$\frac{\bot}{\phi} \tag{259}$$

The negation introduction and elimination rules of logic are as follows:

$$\frac{\phi \vdash \bot}{\neg\phi} \quad \frac{\phi \wedge \neg\phi}{\bot} \tag{260}$$

As statements about morphism existence, these rules say:

1. If there is a morphism $(\phi \wedge \psi) \to \bot$, then there is a morphism $\psi \to \neg\phi$.

2. There is a morphism $(\phi \wedge \neg\phi) \to \bot$.

These rules may be quickly recognized as the universal property of the exponential object $\bot^\phi$ as applied to Σ_1^1.

To say categorically that ϕ is a tautology, we can say in other words that there is a morphism $\neg\bot \to \phi$. For example, there is a morphism $\neg\bot \to (\phi \vee \neg\phi)$, which fact corresponds to the following rule of logic:

$$\frac{}{\phi \vee \neg\phi} \tag{261}$$

Every tautology is itself a terminal object, as it is straightforward to see.

The following formulas, which state axioms for equality, are tautologies in this category, for all de Bruijn indices a, b, c:

1. $x_a = x_a$.

2. $\neg(x_a = x_b) \vee x_b = x_a$.

3. $\neg(x_a = x_b \wedge x_b = x_c) \vee x_a = x_c$.

4. For all formulas ϕ and terms τ:
$$x_a = \tau \to \phi[x_a \mapsto \tau]. \tag{262}$$

The following formulas, corresponding to the ring axioms, are tautologies in this category, for all de Bruijn indices a, b, c:

1. $(a + b) + c = a + (b + c)$.

2. $a + b = b + a$.

3. $a + 0 = a$.

4. $a + (-1 \cdot a) = 0$.

5. $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

6. $a \cdot 1 = a$.

7. $1 \cdot a = a$.

8. $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.

9. $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$.

All morphisms in QF can be generated by applications of function composition and the universal properties of products, coproducts, and exponential objects $\bot^\phi$, starting from the identity morphisms, the initial morphisms from $\bot$, and the tautologies enumerated above (law of excluded middle, equality axioms, and ring axioms). This statement merits some further explanation.

The applications of the universal property of products give us the projection morphisms $\pi_1 : (\phi \wedge \psi) \to \phi$ and $\pi_2 : (\phi \wedge \psi) \to \psi$, and for each pair of morphisms $f : \zeta \to \phi$ and $g : \zeta \to \psi$, the morphism $f \times g : \zeta \to (\phi \wedge \psi)$.

The applications of the universal property of coproducts give us the injection morphisms $\iota_1 : \phi \to (\phi \vee \psi)$ and $\iota_2 : \psi \to (\phi \vee \psi)$, and for each pair of morphisms $f : (\phi \wedge \eta) \to \zeta$ and $g : (\psi \wedge \eta) \to \zeta$, the morphism $f \oplus g : (\phi \vee \psi) \wedge \eta \to \zeta$.

The applications of the universal property of exponential objects $\bot^\phi$ give us the evaluation morphisms $\mathrm{eval} : (\neg\phi \wedge \phi) \to \bot$ and for each morphism $g : (\psi \wedge \phi) \to \bot$ the morphism $\lambda g : \psi \to \neg\phi$.

These universal properties, together with function composition, identity morphisms, initial morphisms, and basic tautologies, can be used to freely generate a category of QF formulas, here called QFP ("QF with proofs"), where the morphisms are all and only the morphisms generated by the following clauses:

1. For all QF formulas ϕ, $\mathrm{id}_\phi \in \hom(\phi, \phi)$.
2. For all QF formulas ϕ, $\mathrm{init}_\phi \in \hom(\bot, \phi)$.
3. For all basic tautologies ϕ, $\mathrm{axiom}_\phi \in \hom(\neg\bot, \phi)$. The basic tautologies are the ring axioms, the equality axioms, and each instance of the law of the excluded middle.
4. For all QF formulas ϕ, ψ, $\pi_{1,\phi,\psi} \in \hom(\phi \wedge \psi, \phi)$, and $\pi_{2,\phi,\psi} \in \hom(\phi \wedge \psi, \psi)$. (The indices may be elided when context makes it clear.)
5. For all QF formulas ϕ, ψ, ζ and morphisms $g \in \hom(\zeta, \phi)$ and $h \in \hom(\zeta, \psi)$, $g \times h \in \hom(\zeta, \phi \wedge \psi)$.
6. For all QF formulas ϕ, ψ, $\iota_{1,\phi,\psi} \in \hom(\phi, \phi \vee \psi)$, and $\iota_{2,\phi,\psi} \in \hom(\psi, \phi \vee \psi)$.
7. For all QF formulas ϕ, ψ, ζ, η and morphisms $g \in \hom(\phi \wedge \eta, \zeta)$ and $h \in \hom(\psi \wedge \eta, \zeta)$, $g \oplus h \in \hom((\phi \vee \psi) \wedge \eta, \zeta)$.
8. For all QF formulas ϕ, $\mathrm{eval}_\phi \in \hom(\neg\phi \wedge \phi, \bot)$.
9. For all QF formulas ϕ, ψ and morphisms $g \in \hom(\psi \wedge \phi, \bot)$, $\lambda g \in \hom(\psi, \neg\phi)$.
10. For all QF formulas ϕ, ψ, ζ, and morphisms $f \in \hom(\phi, \psi)$ and $g \in \hom(\psi, \zeta)$, $g \circ f \in \hom(\phi, \zeta)$.

A morphism in QFP is generated by a finite number of applications of $\times, \oplus, \circ$, and λ to the primitive morphisms id_ϕ, init_ϕ, axiom_ϕ, $\pi_{i,\phi,\psi}$, $\iota_{i,\phi,\psi}$, and $\mathrm{eval}(\phi)$. Here are some examples:

$$\mathrm{id}_{\neg\bot} \times \mathrm{axiom}_{x_1 = x_1} : \neg\bot \to (\neg\bot \wedge x_1 = x_1). \tag{263}$$

$$\pi_1 \circ \pi_1 : ((x_1 = x2 \wedge x_1 = x_3) \wedge x_1 = x_4) \to x_1 = x_2. \tag{264}$$

For any QF formulas ϕ, ψ, ζ,

$$(\iota_1 \circ \pi_1) \oplus (\iota_2 \circ \pi_1) : (\phi \vee \psi) \wedge \zeta \to \phi \vee \psi. \tag{265}$$

For any QF formula ϕ,

$$\lambda\pi_{2,\phi,\bot} : \phi \to \neg\bot. \tag{266}$$

$$(\pi_{1,\phi,\phi} \oplus (\mathrm{init}_\phi \circ \mathrm{eval}_\phi)) \circ ((\mathrm{axiom}_{\phi\vee\neg\phi} \circ \lambda\pi_{2,\neg\neg\phi,\bot}) \times \mathrm{id}_{\neg\neg\phi}) : \neg\neg\phi \to \phi. \tag{267}$$

In QFP, morphism equality is the maximal equality relation; i.e., all morphisms in a hom-set are considered as formally equal for the purposes of categorical axioms and definitions.

It is straightforward to check that QFP has products, coproducts, and exponential objects of the form $\bot^\phi$, which are defined as expected, and that $\bot$ is an initial object and $\neg\bot$ is a terminal object.

Here is the more precise statement of the notion that the morphisms in QF are generated by the mentioned primitives and constructions.

Theorem 5. *There is a functor QF → QFP. This functor preserves products, coproducts, and exponential objects of the form $\bot^\phi$.*

This theorem is a form of completeness theorem. Thinking of the morphism construction rules in QFP as rules of a proof system, this theorem states that these categorical constructions are sufficient to prove every entailment which is semantically true.

Proof. This proof relies on the following lemma.

Lemma 1. *Let ϕ be a QF formula. Suppose that hom$(\phi, \bot) = \emptyset$ in QFP. Then there is a model M such that $\delta_M(\phi) = 1$.*

First let us see how the lemma implies the theorem. All that is required to prove that there is a functor QF → QFP is to prove that whenever there is a morphism $\phi \to \psi$ in QF, there is a morphism $\phi \to \psi$ in QFP. Preservation of products, coproducts, and exponentials $\bot^\phi$ follows straightforwardly due to the maximal morphism equality of both categories. For a proof by contrapositive, suppose that there is no morphism $\phi \to \psi$ in QFP. Then there is no morphism $(\phi \wedge \neg\psi) \to \bot$ in QFP, since if there was such a morphism, call it f, then $\lambda f : \phi \to \neg\neg\psi$ would be a morphism in QFP, and then there would be a morphism $\phi \to \psi$ by composing λf with the morphism $\neg\neg\psi \to \psi$ defined in (267). Then, applying the lemma, there is a model M such that $\delta_M(\phi \wedge \neg\psi) = 1$. This means $\delta_M(\phi) = 1$ and $\delta_M(\psi) = 0$. This means that there is no morphism $\phi \to \psi$ in QF, because M provides a counterexample to the statement that for all models M, $\delta_M(\phi) \leq \delta_M(\psi)$. Thus, the lemma implies the theorem.

Proof of Lemma 1. This proof is in two steps. First, construct a complete, consistent theory which satisfies the provided formula. Second, construct a model which satisfies the theory. A "theory," by definition, is a set of formulas wherein all second order function variables have consistent arities across all formulas. A theory T is "consistent," by definition, iff for all finite $U \subseteq T$, hom$(\bigwedge U, \bot) = \emptyset$. Here $\bigwedge U$ denotes a formula which is a conjunction of the elements of U (in some arbitrary order). A theory T is "complete," by definition, iff for all formulas ϕ, either $\phi \in T$ or $\neg\phi \in T$. A model M satisfies a theory T, by definition, iff for all $\phi \in M$, $\delta_M(\phi) = 1$. These steps rely on the following lemmas.

Lemma 2. *Let T be a consistent theory. There is a complete, consistent theory $T' \supseteq T$.*

Lemma 3. *Let T be a complete, consistent theory. There is a model M such that M satisfies T.*

These lemmas imply Lemma 1 by the following steps. Let ϕ be a formula such that hom$(\phi, \bot) = \emptyset$ in QFP. Then $\{\phi\}$ is consistent. Let $T \supseteq \{\phi\}$ be a complete, consistent theory. Let M be a model satisfying T. Then $\delta_M(\phi) = 1$.

Proof of Lemma 2. Let T be a consistent theory. If T is complete, then we can let $T' = T$ and be done. So suppose T is not complete. Choose a formula ϕ such that $\phi \notin T$ and $\neg\phi \notin T$. By Lemma 4, either $T \cup \{\phi\}$ is consistent or $T \cup \{\neg\phi\}$ is consistent. So choose either $T_1 = T \cup \{\phi\}$ or $T_1 = T \cup \{\neg\phi\}$, whichever one is consistent (or pick one arbitrarily if they are both consistent). Construct an infinite sequence

$$T \supseteq T_1 \supseteq T_2 \supseteq \cdots \tag{268}$$

by repeating this procedure. Precisely, this works as follows. Let $\phi_1, \phi_2, ...$ be an enumeration of all formulas. Given T_i, if T_i is complete then let $T_{i+1} = T_i$, or if T_i is not complete then let $j \in \mathbb{Z}^+$ be the least j such that $\phi_j \notin T_i$ and $\neg\phi_j \notin T_i$, and let $T_{i+1} = T \cup \{\phi_j\}$ if $T \cup \{\phi_j\}$ is consistent, or let $T_{i+1} = T \cup \{\neg\phi_j\}$ otherwise. By Lemma 4, T_{i+1} is consistent. Let $T' = \bigcup_{i \in \mathbb{N}} T_i$. Evidently T' is consistent, because each finite subset of T' is contained in some T_i and each T_i is consistent. Evidently $T' \supseteq T$.

T' is complete, by the following reasoning. Let ψ be a formula. For some $j \in \mathbb{Z}^+$, $\psi = \phi_j$. By induction on j, there is some $i \in \mathbb{Z}^+$ such that for all $j' \leq j$, $\phi_{j'} \in T_i$ or $\neg\phi_{j'} \in T_i$. When $j = 1$, this is true because there is no $j' < j$ and therefore either ϕ_1 or $\neg\phi_1$ had to be added at T_1 if one of them was not already present. When $j = k + 1$, applying the inductive hypothesis gives that for all $j' < j$, $\phi_{j'} \in T_i$ or $\neg\phi_{j'} \in T_i$ for some i, and then T_{i+1} must contain ϕ_j or $\neg\phi_j$ because if one of them is not already present in T_i then one of them must be added in T_{i+1}. Therefore, the inductive argument is done. Therefore, T' is complete. □

Lemma 4. *Let T be a consistent theory. Let ϕ be a formula. Either $T \cup \{\phi\}$ is consistent or $T \cup \{\neg\phi\}$ is consistent.*

Proof. Suppose that $T \cup \{\phi\}$ is not consistent and $T \cup \{\neg\phi\}$ is not consistent. Want to show a contradiction, that T is not consistent. Let $U \subseteq T \cup \{\neg\phi\}$ be a finite subset of $T \cup \{\neg\phi\}$ such that $\hom(\bigwedge U, \perp) \neq \emptyset$ in QFP. Since T is consistent, $\neg\phi \in U$. Choose a conjunction of the formulas in U and call it ψ. Let $f \in \hom(\psi \wedge \neg\phi, \perp)$ be a morphism in QFP. Then $\lambda f : \psi \to \neg\neg\phi$. Let $g : \neg\neg\phi \to \phi$ be a morphism in QFP as defined in (267). Then $g \circ \lambda f : \psi \to \phi$. Since $T \cup \{\phi\}$ is not consistent, there is a finite $V \subseteq T \cup \{\phi\}$ such that $\hom(\bigwedge V, \perp) \neq \emptyset$ in QFP. Since T is consistent, $\phi \in V$. Choose a conjunction of the formulas in $V \cap T$ and call it ζ. Let $h : \zeta \wedge \phi \to \perp$ be a morphism. Then

$$h \circ (\mathrm{id}_\zeta \times (g \circ \lambda f)) : \zeta \wedge \psi \to \perp \tag{269}$$

is a morphism in QFP. Since ζ and ψ are conjunctions of formulas in T, this contradicts the fact that T is consistent. □

Proof of Lemma 3. Let T be a complete theory. Define an equivalence relation over all terms as follows: $\tau_1 \equiv \tau_2$ iff there exists a formula $\psi \in T$ such that there is a morphism $\psi \to \tau_1 = \tau_2$ in QFP. This is an equivalence relation because the equality axioms hold over all first-order variables.

Let R be the set of equivalence classes of $\equiv$. Define the operations $\cdot, +$ to be the unique operations consistent with T. In other words, letting $[\![\tau]\!]$ denote the equivalence class of τ, if T contains $\tau_i \cdot \tau_j = \tau_k$ and $\tau_{i'} \equiv \tau_i$ and $x_{j'} \equiv x_j$ and $x_{k'} \equiv x_k$, then $\tau_{i'} \cdot \tau_{j'} = \tau_{k'}$ is in T. This means that this definition is coherent:

$$[\![\tau_i]\!] \cdot [\![\tau_j]\!] = [\![\tau_k]\!] \tag{270}$$

Similarly, this definition is coherent:

$$[\![\tau_i]\!] + [\![\tau_j]\!] = [\![\tau_k]\!] \tag{271}$$

Let $<$ denote any strict total ordering relation over R. The choice of $<$ does not matter because we are only using the model we are building to evaluate quantifier-free formulas, and their truth values do not depend on the choice of $<$.

Define a function $F : \mathbb{Z}^+ \to R$ by: $F(i) = [\![x_i]\!]$. Define a function $S : \mathbb{Z}^+ \to \sum_{i \in \mathbb{Z}^+} (R^i \to R)$ as follows. Let $i \in \mathbb{Z}^+$. Let n be the arity of f_i in the formulas in T. For all $\vec{x} \in R^i$, define

$$S(i)(\vec{x}) = [\![f_i(\tau_1, ..., \tau_n)]\!] \tag{272}$$

where $\vec{x} = ([\![\tau_1]\!], ..., [\![\tau_n]\!])$. This definition can be seen to be coherent by applying the substitution axiom for equality, which holds in T. Now let

$$M = (R, \cdot, +, [\![0]\!], [\![1]\!], [\![-1]\!], <, F, S). \tag{273}$$

It is straightforward to check that M is a model and that M satisfies T. □

That completes the proof of Theorem 5. □

The next step in defining functorial arithmetization is to define a functor from a subcategory of QFP to $\mathbf{Circ}(\mathbb{F}, n_I, r)$. The subcategory of QFP is a full subcategory, meaning the set of objects is restricted but all morphisms between the restricted set of objects are included in the subcategory. QFP needs to be restricted to limit the numbers of free variables, so that they can fit into the number of instance columns n_I.

A "free variable signature," by definition, is an element of

$$\mathbb{S} = \mathbb{N} \times \sum_{n \in \mathbb{N}} (\mathbb{Z}^+)^n. \tag{274}$$

A free variable signature is a Cartesian pair where the first element specifies the number of free first-order variables, and the second element specifies the number of free second-order variables and the arity of each.

For all $(n, (k_1, ..., k_m)) \in \mathbb{S}$, let $\mathrm{QF}(n, \vec{k})$ and $\mathrm{QFP}(n, \vec{k})$ respectively denote the full subcategories of QF and QFP which include in their objects a formula ϕ iff

$$\mathrm{free}(\phi) \subseteq \{x_1, ..., x_n\} \cup \{f_1, ..., f_m\} \tag{275}$$

and for each $i \in [m]$, the arity of f_i in ϕ is k_i if f_i occurs in ϕ.

It is straightforward to see that the functor QF $\to$ QFP can be restricted to a functor $\mathrm{QF}(n, \vec{k}) \to \mathrm{QFP}(n, \vec{k})$ for all $(n, \vec{k}) \in \mathbb{S}$.

Theorem 6. *Let $s = (n, (k_1, ..., k_m)) \in \mathbb{S}$. Let $\mathbb{F}$ be a finite field. Let $n_I = n + \sum_{i \in [m]} (k_i + 1)$. Let $r \geq |\mathbb{F}| + 1$. There is a functor $QFP(s) \to \mathbf{Circ}(\mathbb{F}, n_I, r)$. This functor preserves products, coproducts, and exponentials of the form $\bot^\phi$.*

Proof. Call the functor F. F is defined on objects by recursion on the structure of QF formulas, as follows.

$$F(\bot) = \bot. \tag{276}$$

$$F(\phi \wedge \psi) = F(\phi) \times F(\psi). \tag{277}$$

$$F(\phi \vee \psi) = F(\phi) \oplus F(\psi). \tag{278}$$

$$F(\neg\phi) = \bot^{F(\phi)}. \tag{279}$$

The only case that remains to be defined is the base case $F(\tau_1 = \tau_2)$. Let $\tau_1 = \tau_2$ be a formula in $\mathrm{QFP}(s)$. Find a circuit $\mathcal{C}$ in $\mathbf{Circ}(\mathbb{F}, n_I, r)$ with the following characteristics:

1. There are no nondeterministic advice columns.

2. For each subterm $f_i(v_1, ..., v_{k_i})$ in $\tau_1 = \tau_2$, there are $n+1$ deterministic advice columns. These columns are determined by the following constraints. In these constraints, $\vec{x}_i$ denotes the instance columns representing the first-order variable x_i. $\vec{y}_i$ denotes the (instance) output column for f_i. $\vec{z}_{i,j}$ denotes the jth (instance) input column for f_i. $\vec{u}$ denotes the (advice) output column for $f_i(v_1, ..., v_{k_i})$. $\vec{v}_j$ denotes the value of v_j. These constraints are defined in terms of a partial function val from terms to polynomials over column vectors, defined recursively as follows:

$$\mathrm{val}(0) = 0 \tag{280}$$

$$\mathrm{val}(1) = 1 \tag{281}$$

$$\mathrm{val}(-1) = -1 \tag{282}$$

Here -1 denotes the additive inverse of $1 \in \mathbb{F}$.

$$\mathrm{val}(\rho + \sigma) = \mathrm{val}(\rho) + \mathrm{val}(\sigma). \tag{283}$$

$$\mathrm{val}(\rho \cdot \sigma) = \mathrm{val}(\rho) \cdot \mathrm{val}(\sigma). \tag{284}$$

$$\mathrm{val}(f_i(\tau_1, ..., \tau_{k_i})) = \vec{u}. \tag{285}$$

Since $\tau_1, ..., \tau_{k_i}$ may contain function application terms, val must be defined in this manner over all subterms of $\tau_1 = \tau_2$, which could be said more explicitly if we had distinct names for all of the deterministic advice columns in play. The constraints are as follows.

(a) For all $j \in [k_i]$,
$$\vec{v}_j = \mathrm{val}(\tau_j). \tag{286}$$

(b) For all $j \in [r]$, $(u_j, v_{1,j}, ..., v_{k_i,j}) \in (\vec{y}_i, \vec{z}_1, ..., \vec{z}_{k_i})$.

3. There is a deterministic advice column, the truth value column, denoted $\vec{t}$. The following constraints are present.

(a) $\vec{t} = 1^r$.

(b) For all $i \in [r]$,
$$(t_i, \mathrm{val}(\tau_1) - \mathrm{val}(\tau_2)) \in \{(1, 0)\} \cup \{(0, x) \mid x \in \mathbb{F} \setminus \{0\}\}. \tag{287}$$

4. There are additional advice columns and constraints which ensure that each provided lookup table in the instance and advice columns defines a function, by ensuring that the input rows are lexicographically ordered and adjacent rows with the same input values have the same output values. This requires deterministic advice columns giving the deltas between each pair of adjacent input cells in the same column and the truth values of the statements that those deltas are positive. This is done similarly to the process laid out in Section 4, except that checking that the deltas are non-negative is done directly using a lookup table instead of using bytewise decompositions.

It is straightforward to check that there is such a circuit $\mathcal{C}$. That completes the definition of F on objects. F is defined on morphisms by recursion on their structure, as follows.

1. $F(\mathrm{id}_\phi) = \mathrm{id}_{F(\phi)}$.

2. $F(\mathrm{init}_\phi) = \mathrm{const}(0^{[n_{F(\phi),O}] \times [r]})$. Here $n_{F(\phi),O}$ denotes the number of nondeterministic advice columns of $F(\phi)$.

3. $F(\mathrm{axiom}_\phi) = \mathrm{const}(0^{[n_{F(\phi),O}] \times [r]})$. For this to define a morphism, it suffices to know that $F(\phi)$ is satisfiable for all possible combinations of instance and nondeterministic advice values. This is straightforward to check for all basic tautologies ϕ.

4. $F(\pi_{1,\phi,\psi}) = (\pi_1 : F(\phi \wedge \psi) \to F(\phi))$.

5. $F(\pi_{2,\phi,\psi}) = (\pi_2 : F(\phi \wedge \psi) \to F(\psi))$.

6. $F(f \times g) = F(f) \times F(g)$.

7. $F(\iota_{1,\phi,\psi}) = (\iota_1 : F(\phi) \to F(\phi \vee \psi))$.

8. $F(\iota_{2,\phi,\psi}) = (\iota_2 : F(\psi) \to F(\phi \vee \psi))$.

9. $F(f \oplus g) = F(f) \oplus F(g)$.

10. $F(\mathrm{eval}_\phi) = \mathrm{const}(())$. This is a morphism $F(\neg\phi \wedge \phi) \to \bot$. It is a morphism because the circuit $F(\neg\phi \wedge \phi)$ is unsatisfiable.

11. $F(\lambda g) = \lambda F(g)$.

12. $F(g \circ f) = F(g) \circ F(f)$.

It is straightforward to see that F is a functor which preserves products, coproducts, and exponentials of the form $\bot^{\phi}$. □

By composing the functors from Theorem 5 and Theorem 6, we obtain a functor $\mathrm{QF}(s) \to \mathrm{Circ}(\mathbb{F}, n_I, r)$. This completes the definition of functorial arithmetization, but not yet the proof that it is correct. The final step is to define denotation functors from each of the three categories involved and prove that the functors preserve denotations. The denotations of objects are relations (represented as sets) and the denotations of morphisms are functions. The overall goal is to define functors F, G, H such that the following diagram commutes, where K is the functor from Theorem 5 and J is the functor from Theorem 6:

$$\begin{tikzcd}
\mathrm{QF}(s) \arrow[r, "J"] \arrow[dr, "F"'] & \mathrm{QFP}(s) \arrow[r, "K"] \arrow[d, "G"] & \mathrm{Circ}(\mathbb{F}, n_I, r) \arrow[dl, "H"] \\
& \mathrm{Set} &
\end{tikzcd} \tag{288}$$

The commutativity of this diagram will straightforwardly imply the correctness of the arithmetization.

Let $s = (n, (k_1, ..., k_m)) \in \mathbb{S}$. Let $\mathbb{F}$ be a finite field. Let $r = |\mathbb{F}| + 1$. Let

$$n_I = n + \sum_{i \in [m]} k_i + 1. \tag{289}$$

The denotation functor $F : \mathrm{QF}(s) \to \mathrm{Set}$ is defined as follows. Let ϕ be a formula in $\mathrm{QF}(s)$. $F(\phi)$ is a relation, a subset of

$$\mathbb{D} = \mathbb{F}^n \times \prod_{i \in [m]} \mathbb{F}^{\mathbb{F}^{k_i}}, \tag{290}$$

specifically:

$$F(\phi) = \{(G, S) \in \mathbb{D} \mid \delta_{M(G,S)}(\phi) = 1\}, \tag{291}$$

where by definition

$$M(G, S) = (\mathbb{F}, \cdot_{\mathbb{F}}, +_{\mathbb{F}}, 0_{\mathbb{F}}, 1_{\mathbb{F}}, -1_{\mathbb{F}}, <_{\mathbb{F}}, G, S). \tag{292}$$

Here $\cdot_{\mathbb{F}}$ denotes the multiplication operation of $\mathbb{F}$, and so forth. $<_{\mathbb{F}}$ denotes the usual ordering on $\mathbb{F}$, namely $0, 1, 1+1, 1+1+1, ...$ (but it could be any arbitrary strict total ordering, since it is unused for evaluating quantifier-free formulas).

Let $f : \phi \to \psi$ be a morphism in QF. Define $F(f)$ to be the embedding $x \mapsto x$. This works because the existence of f implies that in every model where ϕ is true, ψ is also true. It is straightforward to check that identities and composition are preserved, making F a functor.

The denotation functor $G : \mathrm{QFP}(s) \to \mathrm{Set}$ is defined the same way as $F : \mathrm{QF}(s) \to \mathrm{Set}$.

The denotation functor $H : \mathbf{Circ}(\mathbb{F}, n_I, r) \to \mathrm{Set}$ is defined on objects by:

$$H(\mathcal{C}) = \{(G, S) \in \mathbb{D} \mid (\mathrm{inst}(G, S), W) \text{ satisfies } \mathcal{C} \text{ for some } W \in \mathbb{F}^{[n_{\mathcal{C},O}] \times [r]}\}, \tag{293}$$

where

$$\mathrm{inst}((x_1, ..., x_n), (f_1, ..., f_m)) = (x_1^r \ \cdots \ x_n^r \ \mathrm{tab}(f_1) \ \cdots \ \mathrm{tab}(f_m)), \tag{294}$$

where x_i^r denotes the column vector with r rows equal to x_i at each row, and f_i denotes the lookup table of f_i, i.e.,

$$\mathrm{tab}(f_i) = \{(f(z_1, ..., z_{k_i}) \ z_1 \ \cdots \ z_{k_i})\}. \tag{295}$$

This is a slight abuse of notation in that it denotes a matrix as the set of its rows (without specifying the order, which can be any arbitrary order).

Let $f : \mathcal{C} \to \mathcal{D}$ be a morphism in $\mathbf{Circ}(\mathbb{F}, n_I, r)$. f is a function

$$f : \mathbb{F}^{[n_I+n_{\mathcal{C},O}]\times[r]} \to \mathbb{F}^{[n_I+n_{\mathcal{D},O}]\times[r]}. \tag{296}$$

We can therefore define $H(f)$ as f restricted to $H(\mathcal{C})$, because f is already a function $\mathbb{D} \to \mathbb{D}$, and $H(\mathcal{C}) \subseteq \mathbb{D}$, $H(\mathcal{D}) \subseteq \mathbb{D}$, and f maps tables satisfying $\mathcal{C}$ to tables satisfying $\mathcal{D}$. Evidently G preserves identities and function composition, and therefore H is a functor.

It is straightforward to check, using induction on the structure of formulas, that the diagram (288) commutes. That completes the construction of functorial arithmetization for QF formulas. Here is the statement that the construction is correct.

Theorem 7. *Let $\mathbb{F}$ be a finite field. Let $s = (n, (k_1, ..., k_m)) \in \mathbb{S}$. Let $n_I = n + \sum_{i\in[m]} k_i + 1$. Let*

$$\mathbb{D} = \mathbb{F}^n \times \prod_{i\in[m]} \mathbb{F}^{\mathbb{F}^{k_i}}. \tag{297}$$

Let

$$r \geq \max(\{|\mathbb{F}| + 1\} \cup \{|\mathbb{F}^{\mathbb{F}^{k_i}}| \mid i \in [m]\}). \tag{298}$$

Let F be the composition of the functors from Theorem 5 and Theorem 6:

$$F : QF(s) \to \mathbf{Circ}(\mathbb{F}, n_I, r). \tag{299}$$

Define a function $f : \mathbb{D} \to \mathbb{F}^{[n_I]\times[r]}$ by

$$f((x_1 \ \cdots \ x_n), (g_1 \ \cdots \ g_m)) = (x_1^r \ \cdots \ x_n^r \ g_1 \ \cdots \ g_m). \tag{300}$$

Here x_i^r denotes a column vector with r rows and x_i as the value at each row. g_i denotes the tabular representation of the function g_i.

Let ϕ be a formula in $QF(s)$. $(F(\phi), f)$ is an arithmetization of the relation

$$\{(G, S) \in \mathbb{D} \mid \delta_{M(G,S)}(\phi) = 1\}, \tag{301}$$

where

$$M(G, S) = (\mathbb{F}, \cdot_{\mathbb{F}}, +_{\mathbb{F}}, 0_{\mathbb{F}}, 1_{\mathbb{F}}, -1_{\mathbb{F}}, <_{\mathbb{F}}, G, S). \tag{302}$$

This is a corollary of the statement that (288) commutes.

5.3 Byte decompositions

The functorial arithmetization given so far is relatively simple to define and prove correct, but the resulting circuits are not efficient; they have far more rows and columns than necessary. The remaining steps of the proof bridge the gap between the functorial arithmetization of QF formulas and the arithmetization of Σ_1^1 formulas defined explicitly in Section 4, proving that these are equivalent. The step covered in this section is reducing the required number of rows in the circuit by using byte decompositions to reduce the sizes of lookup tables. This step also reduces the domain of the circuit, so that rather than being correct for all instance values in $\mathbb{F}$, the circuit is only correct for all instance values in $[2^{NB}]$ for B the number of bits per byte and N the number of bytes per word. This step increases the number of columns, which will be decreased in the next step after this, in Subsection 5.4.

Let B and N be positive integers. By definition, an "(N, B)-composable circuit" is a tuple

$$\mathcal{C} = (\mathbb{F}, \vec{c}, d, \vec{p}, \vec{\ell}, r, \vec{e}, X), \tag{303}$$

where:

1. Conditions 1-9 and 12 of the definition of composable circuits are satisified.
2. $f : [2^{NB}]^{[n_I+n_O]\times[r]} \to [2^{NB}]^{[n_D]\times[r]}$ is a function, where for all $X \in \{I, O, D\}$,
$$n_X = |\{i \in [n] \mid c_i = X\}|. \tag{304}$$
f is called the deterministic advice computer.
3. For all $Y \in [2^{NB}]^{[n_I+n_O]\times[r]}$, if there exists $Z \in \mathbb{F}^{[n_D]\times[r]}$ such that $[XYZ]$ satisfies $\mathcal{C}$, then $[XYf([XY])]$ satisfies $\mathcal{C}$, and $f([XY])$ is the unique such Z.
4. If the constraint $\vec{t} = 1^r$ is removed from $\mathcal{C}$, the resulting circuit $\mathcal{D}$ is uniquely satisfiable for all combinations of instance and nondeterministic advice values in $[2^{NB}]$. In other words, for all $Y \in [2^{NB}]^{[n_I+n_O]\times[r]}$, $[XYf([XY])]$ satisfies $\mathcal{D}$, and $f([XY])$ is the unique Z such that $[XYZ]$ satisfies $\mathcal{D}$.

This is the same as the definition of composable circuits except that the coherence conditions do not apply over all possible instance values, but only those in $2^[NB]$.

(N, B)-composable circuits also form categories. Let $\mathbb{F}$ be a finite field such that $|\mathbb{F}| > 2^{NB+1}$. Let r be a positive integer, $r \geq 2^B$. Let n_I be a positive integer. Let $\mathbf{Circ}(N, B, \mathbb{F}, n_I, r)$ denote a category. The objects are (N, B)-composable circuits with field $\mathbb{F}$, row count r, and n_I instance columns. A morphism $g : \mathcal{C} \to \mathcal{D}$ is a function
$$g : [2^{NB}]^{[n_I+n_{\mathcal{C},O}]\times[r]} \to [2^{NB}]^{[n_{\mathcal{D},O}]\times[r]}, \tag{305}$$
where $n_{\mathcal{X},O}$ denotes the number of nondeterministic advice columns of the circuit $\mathcal{X}$, such that for all $X \in [2^{NB}]^{[n_I]\times[r]}$ and $Y \in [2^{NB}]^{[n_{\mathcal{C},O}]\times[r]}$, if $[XYf_{\mathcal{C}}([XY])]$ satisfies $\mathcal{C}$, then $[Xg([XY])f_{\mathcal{D}}([Xg([XY])])]$ satisfies $\mathcal{D}$. Here $f_{\mathcal{X}}$ denotes the deterministic advice computer of $\mathcal{X}$.

Equality and composition of morphisms, and identity morphisms, are defined the same as for categories of composable circuits. Initial objects, products, and exponential objects of the form $\bot^\phi$ are defined the same as for categories of composable circuits. The definition of coproducts is different.

5.3.1 Coproducts ($\oplus$)

Let $\mathbf{Circ}(N, B, \mathbb{F}, n_I, r)$ be a category of (N, B)-composable circuits. Let $\mathcal{C}$ and $\mathcal{D}$ be objects of this category. Without loss of generality, assume that $\mathcal{C}$ and $\mathcal{D}$ have the same numbers of nondeterministic advice columns. Find a circuit $\mathcal{C} \oplus \mathcal{D}$ with the following characteristics:

1. $\mathcal{C} \oplus \mathcal{D}$ has the following numbers of advice columns:
$$n_{\mathcal{C}\oplus\mathcal{D},O} = n_O + 1 \tag{306}$$
$$n_{\mathcal{C}\oplus\mathcal{D},D} = n_{\mathcal{C},D} + n_{\mathcal{D},D} + 3 + 2N \tag{307}$$
The nondeterministic advice columns of $\mathcal{C}$ and $\mathcal{D}$ are overlapped, or in other words they share nondeterministic advice. There is an additional nondeterministic advice column, denoted $\vec{s}$, which selects which of the two circuits is applied at each row. The deterministic advice columns of $\mathcal{C}$ and $\mathcal{D}$ are present and separate. Their truth value columns are present and denoted $\vec{u}$ and $\vec{v}$, respectively. There are $3 + 2N$ additional deterministic advice columns. N of them are the elements of the byte decomposition of the selector column $\vec{s}$, denoted $\vec{b}_1, ..., \vec{b}_N$. N of them are related truth value columns. $\vec{c}_1$ is the truth value column for the statements $b_{1,i} \in \{0, 1\}$. For $i \in [N] \setminus \{1\}$, $\vec{c}_i$ is the truth value for the statement $b_{i,j} = 0$. The other three are the truth value column, denoted $\vec{t}$, the $s_i \in \{0, 1\}$ truth value column, denoted $\vec{w}$, and the $s_i = s_{i-1}$ truth value column, denoted $\vec{x}$.
2. All of the constraints of $\mathcal{C}$ and $\mathcal{D}$ are present, except for $\vec{u} = 1^r$ and $\vec{v} = 1^r$.

3. The following constraints are present:

   (a) For all $i \in [r]$, $t_i \in \{0, 1\}$.

   (b) $\vec{t} = 1^r$.

   (c) $\vec{t} = (\vec{s} \cdot \vec{u} + (1^r - \vec{s}) \cdot \vec{v}) \cdot \vec{w} \cdot \vec{x}$, where $\cdot$ is the Hadamard product.

   (d) $\vec{s} = \sum_{i=1}^{N} 2^{W(i-1)} \cdot \vec{b}_i$.

   (e) For all $i \in [r]$,
   $$(b_{1,i}, c_{1,i}) \in \{(0, 1), (1, 1)\} \cup \{(x, 0) \mid x \in [2^W] \setminus \{0, 1\}\}. \tag{308}$$

   (f) For all $\ell \in [N] \setminus \{1\}, i \in [r]$,
   $$(b_{\ell,i}, c_{\ell,i}) \in \{(0, 1)\} \cup \{(x, 0) \mid x \in [2^W] \setminus \{0\}\}. \tag{309}$$

   (g) $\vec{w} = \prod_{\ell=1}^{N} \vec{c}_\ell$.

   (h) For all $i \in [r]$,
   $$(x_i, 1 - (b_{1,i} - b_{1,i-1})) \in \{(1, 0)\} \cup \{(0, x) \mid x \in [2^W] \setminus \{0\}\}. \tag{310}$$

4. The deterministic advice computer, $f_{\mathcal{C} \oplus \mathcal{D}}$, is defined as follows, in terms of the deterministic advice computers $f_{\mathcal{C}}$ of $\mathcal{C}$ and $f_{\mathcal{D}}$ of $\mathcal{D}$. For all $X \in [2^{NW}]^{[n_I] \times [r]}$, $Y \in [2^{NW}]^{[n_O] \times [r]}$, $\vec{s} \in [2^{NW}]^r$,
$$f_{\mathcal{C} \oplus \mathcal{D}}([XY\vec{s}]) = \begin{cases} [f_{\mathcal{C}}([XY])\vec{v}\vec{w}\vec{x}] & \text{if } s_1 = 0, \\ [f_{\mathcal{D}}([XY])\vec{u}\vec{w}\vec{x}] & \text{otherwise,} \end{cases} \tag{311}$$
where $\vec{v}$ denotes the value of the truth value column in $f_{\mathcal{C}}([XY])$, and $\vec{u}$ denotes the value of the truth value column in $f_{\mathcal{D}}([XY])$, and $\vec{w}$ and $\vec{x}$ are the dependent advice columns defined, for all $i \in [r]$, by
$$w_i = \begin{cases} 1 & \text{if } s_i \in \{0, 1\}, \\ 0 & \text{otherwise,} \end{cases} \tag{312}$$
$$x_i = \begin{cases} 1 & \text{if } s_i = s_{i-1}, \\ 0 & \text{otherwise.} \end{cases} \tag{313}$$

5.3.2 Functorial arithmetization

Theorem 8. *Let $s = (n, (k_1, ..., k_m)) \in \mathbb{S}$. Let $n_I = n + \sum_{i=1}^{m}(1 + k_i)$. Let N, B be positive integers. Let $\mathbb{F}$ be a finite field such that $|\mathbb{F}| > 2^{NB+1}$. Let $r \geq 2^B$ be an integer. There is a functor $QFP(s) \to$ $\mathbf{Circ}(N, W, \mathbb{F}, n_I, r)$. This functor preserves products, coproducts, and exponentials of the form $\bot^\phi$.*

Proof. Call the functor F. F is defined on objects by recursion on the structure of QF formulas, as follows.

$$F(\bot) = \bot. \tag{314}$$
$$F(\phi \wedge \psi) = F(\phi) \times F(\psi). \tag{315}$$
$$F(\phi \vee \psi) = F(\phi) \oplus F(\psi). \tag{316}$$
$$F(\neg\phi) = \bot^{F(\phi)}. \tag{317}$$

The only case that remains to be defined is the base case $F(\tau_1 = \tau_2)$. Let $\tau_1 = \tau_2$ be a formula in QFP(s). Find a circuit $\mathcal{C}$ in $\mathbf{Circ}(N, B, \mathbb{F}, n_I, r)$ with the following characteristics:

1. There are no nondeterministic advice columns.

2. For each subterm $f_i(v_1, ..., v_{k_i})$ in $\tau_1 = \tau_2$, there are $n{+}1$ deterministic advice columns. These columns are determined by the following constraints. In these constraints, $\vec{x}_i$ denotes the instance columns representing the first-order variable x_i. $\vec{y}_i$ denotes the (instance) output column for f_i. $\vec{z}_{i,j}$ denotes the jth (instance) input column for f_i. $\vec{u}$ denotes the (advice) output column for $f_i(v_1, ..., v_{k_i})$. $\vec{v}_j$ denotes the value of v_j, a deterministic advice column.

   The constraints are defined in terms of a partial function val from terms to polynomials over column vectors, defined recursively as follows:
$$\text{val}(0) = 0 \tag{318}$$
$$\text{val}(1) = 1 \tag{319}$$
$$\text{val}(-1) = -1 \tag{320}$$
   Here -1 denotes the additive inverse of $1 \in \mathbb{F}$.
$$\text{val}(\rho + \sigma) = \text{val}(\rho) + \text{val}(\sigma). \tag{321}$$
$$\text{val}(\rho \cdot \sigma) = \text{val}(\rho) \cdot \text{val}(\sigma). \tag{322}$$
$$\text{val}(f_i(\rho_1, ..., \rho_{k_i})) = \vec{u}. \tag{323}$$
   Since $\rho_1, ..., \rho_{k_i}$ may contain function application terms, val must be defined in this manner over all subterms of $\tau_1 = \tau_2$, which could be said more explicitly if we had distinct names for all of the deterministic advice columns in play. The constraints are as follows.

   (a) For all $j \in [k_i]$,
$$\vec{v}_j = \text{val}(v_j). \tag{324}$$
   (b) For all $j \in [r]$,
$$(u_j, v_{1,j}, ..., v_{k_i,j}) \in (\vec{y}_i, \vec{z}_1, ..., \vec{z}_{k_i}). \tag{325}$$

3. For each $\ell \in [N]$, $\vec{b}_\ell$ denotes the deterministic advice column containing the ℓth byte of the byte decomposition of the values of $\tau_1 - \tau_2$ at each row. $\vec{b}^{+/-}$ denotes the deterministic advice column containing the sign of the byte decomposition of the values of $\tau_1 - \tau_2$ at each row. The following constraints apply.

   (a) For all $i \in [r], \ell \in [N]$, $b_{\ell,i} \in [2^W]$.
   (b) For all $i \in [r]$, $b_i^{+/-} \in \{0, 1\}$.
   (c) $\text{val}(\tau_1 - \tau_2) = (2 \cdot \vec{b}^{+/-}) \cdot \sum_{\ell=1}^{N} 2^{W\ell} \cdot \vec{b}_\ell$.

4. For each $\ell \in [N]$, $\vec{t}_\ell$ denotes the deterministic advice column containing the truth value at each row i of the statement $b_{\ell,i} = 0$. The following constraint applies. For all $i \in [r], \ell \in [N]$,
$$(b_{\ell,i}, t_{\ell,i}) \in \{(0, 1)\} \cup \{(x, 0) \mid x \in [2^W] \setminus \{0\}\} \tag{326}$$

5. There is a deterministic advice column, the truth value column, denoted $\vec{t}$. The following constraints are present.

   (a) $\vec{t} = 1^r$.
   (b) $\vec{t} = \prod_{i=1}^{N} \vec{t}_{\ell,i}$, where $\prod$ denotes a Hadamard product of column vectors.

6. There are additional advice columns and constraints which ensure that each provided lookup table in the instance and advice columns defines a function, by ensuring that the input rows are lexicographically ordered and adjacent rows with the same input values have the same output values. This requires deterministic advice columns giving the deltas between each pair of adjacent input cells in the same column and the truth values of the statements that those deltas are positive. The process is broken down in more detail in Section 4.

7. The deterministic advice computer is defined in a straightforward manner which computes the necessary values of all the deterministic advice columns as determined by the above constraints.

That completes the definition of the functor F on objects. The definition of F on morphisms is by recursion on the structure of morphisms. The definition is identical to the definition of F on morphisms given in the proof of Theorem 6. F preserves identity, composition, products, coproducts, and exponentials trivially by definition. □

The denotation functor $H : \mathbf{Circ}(N, W, \mathbb{F}, n_I, r) \to \text{Set}$ is defined on objects by:

$$H(\mathcal{C}) = \{(G, S) \in \mathbb{D} \mid (\text{inst}(G, S), Y) \text{ satisfies } \mathcal{C} \text{ for some } Y \in [2^{NW}]^{[n_{\mathcal{C},O}]\times[r]}\}, \tag{327}$$

where

$$\mathbb{D} = [2^{NW}]^n \times \prod_{i=1}^{m} [2^{NW}]^{[2^{NW}]^{k_i}} \tag{328}$$

and

$$\text{inst}((x_1, ..., x_n), (f_1, ..., f_m)) = (x_1^r \ \cdots \ x_n^r \ \text{tab}(f_1) \ \cdots \ \text{tab}(f_m)), \tag{329}$$

where x_i^r denotes the column vector with r rows equal to x_i at each row, and f_i denotes the lookup table of f_i, i.e.,

$$\text{tab}(f_i) = \{(f(z_1, ..., z_{k_i}) \ z_1 \ \cdots \ z_{k_i})\}. \tag{330}$$

This is a slight abuse of notation in that it denotes a matrix as the set of its rows (without specifying the order, which can be any arbitrary order).

Let $F : \text{QF}(s) \to \text{Set}$ be the denotation functor for QF(s). Let $G : \text{QFP}(s) \to \text{Set}$ be the denotation functor for QFP(s). Let $J : \text{QF}(s) \to \text{QFP}(s)$ be the functor defined in Theorem 5. Let $K : \text{QFP}(s) \to \mathbf{Circ}(N, W, \mathbb{F}, n_I, r)$ be the functor defined in Theorem 8. It is straightforward to check using induction on the structure of formulas that the following diagram commutes:

$$\begin{tikzcd} \text{QF}(s) \arrow[r, "J"] \arrow[dr, "F"'] & \text{QFP}(s) \arrow[r, "K"] \arrow[d, "G"] & \mathbf{Circ}(N, W, \mathbb{F}, n_I, r) \arrow[dl, "H"] \\ & \text{Set} & \end{tikzcd} \tag{331}$$

From this it follows as a corollary that:

Theorem 9. *Let N and B be positive integers. Let $\mathbb{F}$ be a finite field, such that $|\mathbb{F}| > 2^{NB+1}$. Let $s = (n, (k_1, ..., k_m)) \in \mathbb{S}$. Let $n_I = n + \sum_{i\in[m]} k_i + 1$. Let*

$$\mathbb{D} = [2^{NB}]^n \times \prod_{i\in[m]} [2^{NB}]^{[2^{NB}]^{k_i}}. \tag{332}$$

Let $r \geq 2^B$. Let F be the composition of the functors from Theorem 5 and Theorem 8:

$$F : QF(s) \to \mathbf{Circ}(N, B, \mathbb{F}, n_I, r). \tag{333}$$

Define a function $f : \mathbb{D} \to \mathbb{F}^{[n_I] \times [r]}$ by

$$f((x_1 \ \cdots \ x_n), (g_1 \ \cdots \ g_m)) = (x_1^r \ \cdots \ x_n^r \ g_1 \ \cdots \ g_m). \tag{334}$$

Here x_i^r denotes a column vector with r rows and x_i as the value at each row. g_i denotes the tabular representation of the function g_i.

Let ϕ be a formula in $QF(s)$. $(F(\phi), f)$ is an arithmetization of the relation

$$\{(G, S) \in \mathbb{D} \mid \delta_{M(G,S)}(\phi) = 1 \textit{ and for all } f \in \textit{im}(S),\ |f| \leq r\}, \tag{335}$$

where

$$M(G, S) = (\mathbb{F}, \cdot_{\mathbb{F}}, +_{\mathbb{F}}, 0_{\mathbb{F}}, 1_{\mathbb{F}}, -1_{\mathbb{F}}, <_{\mathbb{F}}, G, S), \tag{336}$$

and $im(f)$ denotes the image of the (partial) function f.

5.4 Quantified formulas

Overall, the goal is to prove:

Theorem 10. *Let n and m be non-negative integers. Let $k_1, ..., k_m$ be positive integers. Let ϕ be a Σ_1^1 formula with n free first-order variables $x_1, ..., x_n$ and m free second-order variables $f_1, ..., f_m$, where f_i has arity k_i for each $i \in [m]$. Let $n_I = \sum_{i \in m} k_i + 1$.*

Let N and B be positive integers. Let

$$\mathbb{D} = [2^{NB}]^n \times \prod_{i \in [m]} [2^{NB}]^{[2^{NB}]^{k_i}}. \tag{337}$$

Let $\mathcal{C}$ be a circuit arithmetizing ϕ as defined in Section 4, using byte decompositions of N bits per byte and B bytes per word. Note that n_I is equal to the number of instance columns of $\mathcal{C}$. Let r be the number of rows $\mathcal{C}$. Define a function $f : \mathbb{D} \to \mathbb{F}^{[n_I] \times [r]}$ by

$$f((x_1 \ \cdots \ x_n), (g_1 \ \cdots \ g_m)) = (x_1^r \ \cdots \ x_n^r \ g_1 \ \cdots \ g_m). \tag{338}$$

Here x_i^r denotes a column vector with r rows and x_i as the value at each row. g_i denotes the tabular representation of the function g_i.

Then $(\mathcal{C}, f)$ is an arithmetization of the relation

$$\{(G, S) \in \mathbb{D} \mid \delta_{M(G,S)}(\phi) = 1 \textit{ and for all } f \in \textit{im}(S),\ |f| \leq r\}, \tag{339}$$

where

$$M(G, S) = (\mathbb{Z}, \cdot, +, 0, 1, -1, <, G, S), \tag{340}$$

and $im(f)$ denotes the image of the (partial) function f.

This is a more precise restatement of Theorem 1.

Proof sketch. The reasons why this is true are straightforward but not simple. Lacking language to describe the reasons succinctly and rigorously, I have opted for a description of the reasons which is succinct but not rigorous. Work is underway to formalize this proof sketch, to resolve any doubts and uncover any errors which may remain in the construction.

The proof strategy so far has been to prove the correctness of an arithmetization which is simple to define and simple to prove correct (Subsection 5.2), but which has an astronomical number of rows and columns, and then to prove its equivalence to an arithmetization which has an astronomical number of columns but a small number of rows (Subsection 5.3). The desired goal is to prove these arithmetizations equivalent to the more efficient arithmetization given explicitly in Section 4, which is harder to describe but has a more reasonable number of columns as well as a reasonable number of rows.

The reason these circuits are equivalent is that the circuit of Subsection 5.3 can be turned into the circuit of Section 4 for the same formula (prior to applying quantifier elimination), by applying a series of semantics preserving circuit transformations. The transformations are semantics preserving in the sense that they do not change the set of instance values on which the circuit is satisfiable.

The first transformation undoes the quantifier elimination done on the input formula ϕ, by applying a transformation to the circuit matrix so that values spread across multiple columns in the input circuit are consolidated into rows in the output circuit. Quantifier elimination turns universally quantified formulas $\forall < \beta.\ \phi$ into conjunctions $\bigwedge_{i \in [\beta]} \phi[x_1 \mapsto i]$. Quantifier elimination turns existentially quantified formulas $\exists < \beta.\phi$ into disjunctions $\bigvee_{i \in [\beta]} \phi[x_1 \mapsto i]$. This results in a whole bunch of advice columns handling each of these conjuncts or disjuncts. These advice columns encode constraints which are encoded in the output circuit using multiple rows with the same constraints. Each existential quantifier generates one or more nondeterministic advice columns in the output circuit (one for a first-order quantifier, or more than one for a second-order quantifier). Each universal quantifier generates a fixed column. These universal quantifier fixed columns enumerate a falsifying set of values of the universally quantified variables. For each such combination, the nondeterministic existential advice columns provide witness values. These witness values are subject to equality constraints to ensure consistency with the quantifier ordering in ϕ. The transformation from many columns from the quantifier-eliminated formula, to fewer columns and more rows, correlates to a transformation of the circuit input matrices. The key steps are to define this transformation and prove that the circuit constraints are preserved by this transformation and its inverse. From this it follows that the circuit transformation is semantics preserving.

The next transformation further reduces the number of advice columns by consolidating truth value computations. It is not necessary to have advice columns for the truth values of all subformulas, but only for the truth values of atomic formulas (equality statements) and the truth value of the overall formula. All of the truth value computations for subformulas leading to the final truth value computation can be collapsed into a single constraint. This is another semantics preserving transformation of the circuit input matrix.

The next transformation further reduces the number of advice columns by discarding redundant advice columns. There will potentially be redundant deterministic advice columns which always contain the same values as each other, resulting from the base case of the functorial arithmetization of Subsection 5.3. These redundant advice columns help to check that each function lookup table defines a function. For each redundant advice column, only one of the redundant copies needs to be kept and the rest can be discarded. This is another semantics preserving transformation of the circuit input matrix.

That completes this admittedly sketchy definition and correctness proof sketch of arithmetizing Σ^1_1 formulas in Halo 2.

6 The other direction

This research has shown how arbitrary Σ^1_1 formulas can be arithmetized in Halo 2. The author conjectures that the other direction is also true: that arbitrary Halo 2 circuits can be expressed as Σ^1_1 formulas. In other words, for every circuit $\mathcal{C}$, there exists a Σ^1_1 formula ϕ such that the set of values for the free variables which satisfy ϕ is equal to the set of instance values for which $\mathcal{C}$ is satisfiable. This conjecture should be straightforward to define more precisely and then prove, by defining a Σ^1_1 formula which encodes all the constraints of a given Halo 2 circuit. If true, this conjecture together with Σ^1_1 arithmetization show that Σ^1_1 formulas have the same expressive power (i.e. can express the same relations) as Halo 2 circuits.

References

[1] The Electric Coin Company. *The halo2 Book.* 2021. `https://zcash.github.io/halo2/index.html`

[2] The Electric Coin Company. *halo2.* 2022. `https://github.com/zcash/halo2`

[3] Sean Bowe, Jack Grigg, and Daira Hopwood. *Recursive Proof Composition without a Trusted Setup.* IACR Cryptol. ePrint Arch., 2019, #1021. `https://eprint.iacr.org/2019/1021`