IMPERIAL

SNARKs for First Order Logic

Murdoch Gabbay & Andrew Mendelsohn
09/05/2026

Introduction

- Highly efficient SNARKs exist to prove knowledge of data satisfying relations.
- These proof systems take inputs of a specific form (e.g. R1CS), usually over finite fields.
- But many relations we may wish to prove are not naturally defined using finite field arithmetic.
- Converting initial relations into equivalent relations over finite fields can be costly.
- Often the relations we want to prove are mathematical (e.g. cryptographic relations).

Our Work: Summary

We design an arithmetisation from first order logic to a SNARK-friendly constraint system.

What does this mean?

Our Work: Summary

We design an arithmetisation from first order logic to a SNARK-friendly constraint system.

What does this mean?

First Order Logic: mathematical expressions of the form $\forall X.P$, $\exists Y.P \wedge Q$, $\forall X \exists Y.P \vee Q$, etc.

Our Work: Summary

We design an arithmetisation from first order logic to a SNARK-friendly constraint system.

What does this mean?

First Order Logic: mathematical expressions of the form $\forall X.P$, $\exists Y.P \wedge Q$, $\forall X \exists Y.P \vee Q$, etc.

Arithmetisation: reformulate a mathematical problem as a polynomial root finding problem.

Our Work: Summary

We design an arithmetisation from first order logic to a SNARK-friendly constraint system.

What does this mean?

First Order Logic: mathematical expressions of the form $\forall X.P$, $\exists Y.P \wedge Q$, $\forall X \exists Y.P \vee Q$, etc.

Arithmetisation: reformulate a mathematical problem as a polynomial root finding problem.

SNARKs: succinct non-interactive arguments of knowledge, proving knowledge of witnesses to instances.

Our Work: Summary

We design an arithmetisation from first order logic to a SNARK-friendly constraint system.

What does this mean?

First Order Logic: mathematical expressions of the form $\forall X.P$, $\exists Y.P \wedge Q$, $\forall X \exists Y.P \vee Q$, etc.

Arithmetisation: reformulate a mathematical problem as a polynomial root finding problem.

SNARKs: succinct non-interactive arguments of knowledge, proving knowledge of witnesses to instances.

Constraint System: any given SNARK requires inputs to have a certain form. The inputs define conditions the instance-witness pair must satisty. These conditions comprise a system of constraints.

Our Work: Summary

We design an arithmetisation from first order logic to a SNARK-friendly constraint system.

What does this mean?

First Order Logic: mathematical expressions of the form $\forall X.P$, $\exists Y.P \wedge Q$, $\forall X \exists Y.P \vee Q$, etc.

Arithmetisation: reformulate a mathematical problem as a polynomial root finding problem.

SNARKs: succinct non-interactive arguments of knowledge, proving knowledge of witnesses to instances.

Constraint System: any given SNARK requires inputs to have a certain form. The inputs define conditions the instance-witness pair must satisty. These conditions comprise a system of constraints.

This allows us to create succinct proofs of logical validity.

Preliminary Definitions: Bits and Bobs

Bits:
- Given bit string $b = (b_1, b_2, \ldots, b_\mu)$, define $b2int(b) = \sum_{1 \le \nu \le \mu} b_\nu \cdot 2^{\nu-1}$
- Given $i \in \mathbb{N}$, write $\mathfrak{b}_\mathsf{b}(i)$ for the binary decomposition of i.
  Note $i = b2int(\mathfrak{b}_\mathsf{b}(i))$.

Preliminary Definitions: Bits and Bobs

Bits:
- Given bit string $b = (b_1, b_2, \ldots, b_\mu)$, define $b2int(b) = \sum_{1 \le \nu \le \mu} b_\nu \cdot 2^{\nu-1}$
- Given $i \in \mathbb{N}$, write $\mathbb{0}_\mathsf{b}(i)$ for the binary decomposition of i.
  Note $i = b2int(\mathbb{0}_\mathsf{b}(i))$.

Multilinear polynomials: multivariate polynomials $f(\mathbf{X}) = \sum_{i=1}^{<\infty} a_i \prod_{j=1}^{\mu} X_j^{e_{ij}}$ in $\mathbf{X} = (X_1, \ldots, X_\mu)$

such that $0 \le e_{ij} \le 1$ for all i, j.

Multilinear extensions: given $f(\mathbf{X}) : \{0,1\}^\mu \to \mathcal{R}$, define $\tilde{f}$, the MLE of f:

$$\tilde{f}(\mathbf{X}) = \sum_{y \in \{0,1\}^\mu} f(y) \cdot \mathsf{eq}(y, \mathbf{X}), \quad \mathsf{eq}(y, \mathbf{X}) = \prod_{i=1}^{\mu} (y_i X_i + (1-y_i)(1-X_i))$$

$\tilde{f}$ is the unique multilinear polynomial: $f(y) = \tilde{f}(y)$ for every $y \in \{0,1\}^\mu$.

Preliminary Definitions: Bits and Bobs

Bits: • Given bit string $b = (b_1, b_2, \ldots, b_\mu)$, define $b2int(b) = \sum_{1 \le \nu \le \mu} b_\nu \cdot 2^{\nu-1}$

• Given $i \in \mathbb{N}$, write $\mathbb{0}_\mathsf{b}(i)$ for the binary decomposition of i.
Note $i = b2int(\mathbb{0}_\mathsf{b}(i))$.

Multilinear polynomials: multivariate polynomials $f(\mathbf{X}) = \sum_{i=1}^{<\infty} a_i \prod_{j=1}^{\mu} X_j^{e_{ij}}$ in $\mathbf{X} = (X_1, \ldots, X_\mu)$

such that $0 \le e_{ij} \le 1$ for all i, j.

Multilinear extensions: given $f(\mathbf{X}) : \{0,1\}^\mu \to \mathcal{R}$, define $\tilde{f}$, the MLE of f:

$$\tilde{f}(\mathbf{X}) = \sum_{y \in \{0,1\}^\mu} f(y) \cdot \mathsf{eq}(y, \mathbf{X}), \quad \mathsf{eq}(y, \mathbf{X}) = \prod_{i=1}^{\mu} (y_i X_i + (1-y_i)(1-X_i))$$

$\tilde{f}$ is the unique multilinear polynomial: $f(y) = \tilde{f}(y)$ for every $y \in \{0,1\}^\mu$.

Interpolation:

- *Lagrange*: $P(x_1, \ldots, x_k)(x)$, the unique degree $k-1$ univariate polynomial interpolating k points.
- *Multilinear*: given $\mathbf{v} = (\mathbf{v}_0, \ldots, \mathbf{v}_{2^\mu - 1})$, let $f_\mathbf{v} : \{0,1\}^\mu \to \mathcal{R},\ x \mapsto \mathbf{v}_{b2int(x)}$.
E.g. $f_\mathbf{v}(101) = \mathbf{v}_4$. Then the MLE of $\mathbf{v}$ is $\tilde{f}_\mathbf{v}$.

Preliminary Definitions: Zinc

Zinc: a SNARK for integer multilinear relations.

Preliminary Definitions: Zinc

Zinc: a SNARK for integer multilinear relations.

Parameters: $\mathrm{pp} = (k, \mu, \partial) \in \mathbb{N}^3$.

$\mathcal{Q}$: a set of multivariate polynomials with coefficients in $\mathbb{Z}$ and $\mathbf{X} = (X_1, \ldots, X_\mu)$.

Preliminary Definitions: Zinc

Zinc: a SNARK for integer multilinear relations.

Parameters: $\mathrm{pp} = (k, \mu, \partial) \in \mathbb{N}^3$.

$\mathcal{Q}$: a set of multivariate polynomials with coefficients in $\mathbb{Z}$ and $\mathbf{X} = (X_1, \ldots, X_\mu)$.

An algebraic indexed relation (**AIR**) is a set $\mathrm{REL}_{\mathrm{pp},\mathcal{Q}}$ of pairs $(\mathbb{x}, \mathbb{w})$ such that:

1. $\mathbb{w} = (f_1, \ldots, f_k)$, a k-tuple of multilinear polynomials with ∂-bounded coefficients.
2. $\mathbb{x} = ([f_1], \ldots, [f_k])$, where the $[f_i]$ are commitments to the f_i.
3. $\mathcal{Q}$ contains polynomials Q which equal zero when evaluated on

$$(f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))_{\mathbf{x} \in \{0,1\}^\mu}$$

Preliminary Definitions: Zinc

Zinc: a SNARK for integer multilinear relations.

Parameters: $\mathrm{pp} = (k, \mu, \partial) \in \mathbb{N}^3$.

$\mathcal{Q}$: a set of multivariate polynomials with coefficients in $\mathbb{Z}$ and $\mathbf{X} = (X_1, \ldots, X_\mu)$.

An algebraic indexed relation (**AIR**) is a set $\mathrm{REL}_{\mathrm{pp},\mathcal{Q}}$ of pairs $(\mathbb{x}, \mathbb{w})$ such that:

1. $\mathbb{w} = (f_1, \ldots, f_k)$, a k-tuple of multilinear polynomials with ∂-bounded coefficients.
2. $\mathbb{x} = ([f_1], \ldots, [f_k])$, where the $[f_i]$ are commitments to the f_i.
3. $\mathcal{Q}$ contains polynomials Q which equal zero when evaluated on

$$\left(f_1(\mathbf{x}),\ \ldots,\ f_k(\mathbf{x}) \right)_{\mathbf{x} \in \{0,1\}^\mu}$$

$$\mathrm{REL}_{\mathrm{pp},\mathcal{Q}} = \left\{ (\mathbb{x}, \mathbb{w}) \;\middle|\; \begin{array}{l} \mathrm{pp} = (k, \mu, \partial), \\ \mathbb{x} = ([f_1], \ldots, [f_k]), \\ \mathbb{w} = (f_1, \ldots, f_k), \\ Q\left((f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))_{\mathbf{x} \in \{0,1\}^\mu} \right) \\ \qquad = 0, \quad \text{for all } Q \in \mathcal{Q} \end{array} \right\}$$

Preliminary Definitions: FOL

Fix some formal symbols:

1. A matrix variable symbol $\mathtt{C}$ with an attached integer arity $ar(\mathtt{C}) \geq 1$.
2. A single index variable symbol X.

$$\begin{array}{ll} \text{Term} & t ::= q \mid t + t \mid t * t \mid X \mid \mathtt{C}_i(X) \mid \mathtt{C}_i\mathtt{C}_j(X) \\ \text{Pred} & \phi, \psi ::= t = t \mid \phi \wedge \psi \mid \phi \vee \psi \\ \text{EP} & P ::= q \mid P{+}P \mid P{*}P \mid X \mid \mathtt{C}_i(X) \mid \mathtt{C}_i\mathtt{C}_j(X) \\ & \qquad (i, j \in [ar(\mathtt{C})], q \in \mathbb{Z}) \end{array}$$

Figure: Terms, predicates, and enriched polynomials; we ignore range checks!

Preliminary Definitions: FOL

Fix some formal symbols:

1. A matrix variable symbol $\mathtt{C}$ with an attached integer arity $ar(\mathtt{C}) \geq 1$.
2. A single index variable symbol X.

$$\begin{array}{ll} \text{Term} & t ::= q \mid t + t \mid t * t \mid X \mid \mathtt{C}_i(X) \mid \mathtt{C}_i\mathtt{C}_j(X) \\ \text{Pred} & \phi, \psi ::= t = t \mid \phi \wedge \psi \mid \phi \vee \psi \\ \text{EP} & P ::= q \mid P + P \mid P * P \mid X \mid \mathtt{C}_i(X) \mid \mathtt{C}_i\mathtt{C}_j(X) \\ & \qquad (i, j \in [ar(\mathtt{C})], q \in \mathbb{Z}) \end{array}$$

Figure: Terms, predicates, and enriched polynomials; we ignore range checks!

Example: The standard inductive definition of a^b for $a, b \in \mathbb{N}$ is a function $\mathrm{pow}(x, y)$ satisfying:

base case	$\mathrm{pow}(a, 0) = 1$
inductive step	$\mathrm{pow}(a, b+1) = a * \mathrm{pow}(a, b)$

This may be written as

$$\forall a.(\mathrm{pow}(a, 0) = 1) \wedge \forall a, b.(\mathrm{pow}(a, b+1) = a * \mathrm{pow}(a, b))$$

Preliminary Definitions: FOL

Example: a^b, i.e. $\mathrm{pow}(x, y)$: $\begin{array}{l}\mathrm{pow}(a, 0) = 1 \\ \mathrm{pow}(a, b+1) = a * \mathrm{pow}(a, b),\end{array}$

i.e. $\forall a.(\mathrm{pow}(a, 0) = 1) \wedge \forall a, b.(\mathrm{pow}(a, b+1) = a * \mathrm{pow}(a, b))$

In our FOL syntax: let $\mathtt{pow}$ be an arity 4 matrix variable symbol with rows $\mathtt{pow}_i$. Set

$$\begin{aligned}\mathrm{baseCase}(X) &= (\mathtt{pow}_2(X) = 0) \wedge (\mathtt{pow}_3(X) = 1), \\ \mathrm{indStep}(X) &= \mathtt{pow}_1(X) = \mathtt{pow}_1(\mathtt{pow}_4(X)) \quad \wedge \\ &\quad\ \mathtt{pow}_2(X) = \mathtt{pow}_2(\mathtt{pow}_4(X)) + 1 \quad \wedge \\ &\quad\ \mathtt{pow}_3(X) = \mathtt{pow}_1(X) \cdot \mathtt{pow}_3(\mathtt{pow}_4(X))\end{aligned}$$

Preliminary Definitions: FOL

Example: a^b, i.e. $\mathrm{pow}(x, y)$: $\begin{array}{l}\mathrm{pow}(a, 0) = 1 \\ \mathrm{pow}(a, b+1) = a * \mathrm{pow}(a, b),\end{array}$

i.e. $\forall a.(\mathrm{pow}(a, 0) = 1) \wedge \forall a, b.(\mathrm{pow}(a, b+1) = a * \mathrm{pow}(a, b))$

In our FOL syntax: let $\mathtt{pow}$ be an arity 4 matrix variable symbol with rows $\mathtt{pow}_i$. Set

$$\begin{aligned}\mathrm{baseCase}(X) &= (\mathtt{pow}_2(X) = 0) \wedge (\mathtt{pow}_3(X) = 1), \\ \mathrm{indStep}(X) &= \mathtt{pow}_1(X) = \mathtt{pow}_1(\mathtt{pow}_4(X)) \quad \wedge \\ &\quad\ \mathtt{pow}_2(X) = \mathtt{pow}_2(\mathtt{pow}_4(X)) + 1 \quad \wedge \\ &\quad\ \mathtt{pow}_3(X) = \mathtt{pow}_1(X) \cdot \mathtt{pow}_3(\mathtt{pow}_4(X))\end{aligned}$$

The $\mathtt{pow}_i(X)$ are enriched polynomials: $\mathtt{pow}_1$ corresponds to bases, $\mathtt{pow}_2$ to exponents, $\mathtt{pow}_3$ to the function output, and $\mathtt{pow}_4$ to pointers.

Preliminary Definitions: FOL

Example: a^b, i.e. $\mathrm{pow}(x, y)$: $\begin{array}{l}\mathrm{pow}(a, 0) = 1 \\ \mathrm{pow}(a, b+1) = a * \mathrm{pow}(a, b),\end{array}$

i.e. $\forall a.(\mathrm{pow}(a, 0) = 1) \wedge \forall a, b.(\mathrm{pow}(a, b+1) = a * \mathrm{pow}(a, b))$

In our FOL syntax: let $\mathtt{pow}$ be an arity 4 matrix variable symbol with rows $\mathtt{pow}_i$. Set

$$\begin{aligned}\mathrm{baseCase}(X) &= (\mathtt{pow}_2(X) = 0) \wedge (\mathtt{pow}_3(X) = 1), \\ \mathrm{indStep}(X) &= \mathtt{pow}_1(X) = \mathtt{pow}_1(\mathtt{pow}_4(X)) \quad \wedge \\ &\qquad \mathtt{pow}_2(X) = \mathtt{pow}_2(\mathtt{pow}_4(X)) + 1 \quad \wedge \\ &\qquad \mathtt{pow}_3(X) = \mathtt{pow}_1(X) \cdot \mathtt{pow}_3(\mathtt{pow}_4(X))\end{aligned}$$

The $\mathtt{pow}_i(X)$ are enriched polynomials: $\mathtt{pow}_1$ corresponds to bases, $\mathtt{pow}_2$ to exponents, $\mathtt{pow}_3$ to the function output, and $\mathtt{pow}_4$ to pointers.

Then

$$\phi_{\mathrm{pow}} = \mathrm{baseCase}(X) \vee \mathrm{indStep}(X)$$

also encodes the power function.

Our Work: the Logical Side

An Interpretation $\varsigma : \mathtt{C} \mapsto \varsigma(\mathtt{C}) \in \mathcal{M}_{ar(\mathtt{C}) \times len(\varsigma(\mathtt{C}))}(\mathbb{N})$ maps $\mathtt{C}$ to an $ar(\mathtt{C}) \times len(\varsigma(\mathtt{C}))$ matrix.

Our Work: the Logical Side

An Interpretation $\varsigma : \mathtt{C} \mapsto \varsigma(\mathtt{C}) \in \mathcal{M}_{ar(\mathtt{C}) \times len(\varsigma(\mathtt{C}))}(\mathbb{N})$ maps $\mathtt{C}$ to an $ar(\mathtt{C}) \times len(\varsigma(\mathtt{C}))$ matrix.
Define a semantics $\langle t \rangle^x_\varsigma \in \mathbb{Z}$, $\langle \phi \rangle^x_\varsigma \in \mathbb{N}$:

$$\langle q \rangle^x_\varsigma = q \qquad \langle t + t' \rangle^x_\varsigma = \langle t \rangle^x_\varsigma + \langle t' \rangle^x_\varsigma \qquad \langle t * t' \rangle^x_\varsigma = \langle t \rangle^x_\varsigma * \langle t' \rangle^x_\varsigma$$

$$\langle X \rangle^x_\varsigma = x \qquad \langle \mathtt{C}_i(X) \rangle^x_\varsigma = \varsigma(\mathtt{C})_{\mathbf{@}i,x} \qquad \langle \mathtt{C}_i\mathtt{C}_j(X) \rangle^x_\varsigma = \varsigma(\mathtt{C})_{\mathbf{@}i,\varsigma(\mathtt{C})_{\mathbf{@}j,x}}$$

$$\langle t = t' \rangle^x_\varsigma = (\langle t \rangle^x_\varsigma - \langle t' \rangle^x_\varsigma)^2 \qquad \langle \phi \wedge \phi' \rangle^x_\varsigma = \langle \phi \rangle^x_\varsigma + \langle \phi' \rangle^x_\varsigma \qquad \langle \phi \vee \phi' \rangle^x_\varsigma = \langle \phi \rangle^x_\varsigma * \langle \phi' \rangle^x_\varsigma$$

Write: $x \vDash_\varsigma \phi$ when $\langle \phi \rangle^x_\varsigma = 0$.

Our Work: the Logical Side

An Interpretation $\varsigma : \mathtt{C} \mapsto \varsigma(\mathtt{C}) \in \mathcal{M}_{ar(\mathtt{C}) \times len(\varsigma(\mathtt{C}))}(\mathbb{N})$ maps $\mathtt{C}$ to an $ar(\mathtt{C}) \times len(\varsigma(\mathtt{C}))$ matrix.
Define a semantics $\langle t \rangle^x_\varsigma \in \mathbb{Z}$, $\langle \phi \rangle^x_\varsigma \in \mathbb{N}$:

$$\langle q \rangle^x_\varsigma = q \qquad \langle t + t' \rangle^x_\varsigma = \langle t \rangle^x_\varsigma + \langle t' \rangle^x_\varsigma \qquad \langle t * t' \rangle^x_\varsigma = \langle t \rangle^x_\varsigma * \langle t' \rangle^x_\varsigma$$

$$\langle X \rangle^x_\varsigma = x \qquad \langle \mathtt{C}_i(X) \rangle^x_\varsigma = \varsigma(\mathtt{C})_{@i,x} \qquad \langle \mathtt{C}_i\mathtt{C}_j(X) \rangle^x_\varsigma = \varsigma(\mathtt{C})_{@i,\varsigma(\mathtt{C})_{@j,x}}$$

$$\langle t = t' \rangle^x_\varsigma = (\langle t \rangle^x_\varsigma - \langle t' \rangle^x_\varsigma)^2 \qquad \langle \phi \wedge \phi' \rangle^x_\varsigma = \langle \phi \rangle^x_\varsigma + \langle \phi' \rangle^x_\varsigma \qquad \langle \phi \vee \phi' \rangle^x_\varsigma = \langle \phi \rangle^x_\varsigma * \langle \phi' \rangle^x_\varsigma$$

Write: $x \vDash_\varsigma \phi$ when $\langle \phi \rangle^x_\varsigma = 0$.

This semantics is sound and complete. We sum up the meaning of the semantics with slogans:

1. Slogan 1. Zero = true; non-zero = false.
2. Slogan 2. Equality = square of difference; conjunction = sum; disjunction = product.
3. Slogan 3. Composed enriched polynomials are pointers.
4. Slogan 4. Interpretations are witnesses to logical validity.

Recall: *Arithmetising Logic : Polynomial Semantics of FOL* **[Gab25]**

[Gab25] introduced a (less general) semantics using Lagrange interpolation.

Let ς be an interpretation on $\mathtt{C}$. Let $P(\varsigma(\mathtt{C})_i)(x)$ interpolate the ith row of $\varsigma(\mathtt{C})$.

[Gab25] gave a semantics with

$$\langle \mathtt{C}_i(X)\rangle^x_\varsigma = P(\varsigma(\mathtt{C})_i)(x) \qquad \langle \mathtt{C}_i\mathtt{C}_j(X)\rangle^x_\varsigma = P(\varsigma(\mathtt{C})_i)(P(\varsigma(\mathtt{C})_j)(x))$$

Recall: *Arithmetising Logic : Polynomial Semantics of FOL* **[Gab25]**

[Gab25] introduced a (less general) semantics using Lagrange interpolation.

Let ς be an interpretation on $\mathtt{C}$. Let $P(\varsigma(\mathtt{C})_i)(x)$ interpolate the ith row of $\varsigma(\mathtt{C})$.

[Gab25] gave a semantics with

$$\langle \mathtt{C}_i(X)\rangle^x_\varsigma = P(\varsigma(\mathtt{C})_i)(x) \qquad \langle \mathtt{C}_i\mathtt{C}_j(X)\rangle^x_\varsigma = P(\varsigma(\mathtt{C})_i)(P(\varsigma(\mathtt{C})_j)(x))$$

Example: the power function 2^2. A valid interpretation is

$$\begin{pmatrix} 2 & 2 & 2 \\ 0 & 1 & 2 \\ 1 & 2 & 4 \\ 1 & 1 & 2 \end{pmatrix}$$

Recall: *Arithmetising Logic : Polynomial Semantics of FOL* **[Gab25]**

[Gab25] introduced a (less general) semantics using Lagrange interpolation.

Let ς be an interpretation on $\mathtt{C}$. Let $P(\varsigma(\mathtt{C})_i)(x)$ interpolate the ith row of $\varsigma(\mathtt{C})$.

[Gab25] gave a semantics with

$$\langle \mathtt{C}_i(X)\rangle^x_\varsigma = P(\varsigma(\mathtt{C})_i)(x) \qquad \langle \mathtt{C}_i\mathtt{C}_j(X)\rangle^x_\varsigma = P(\varsigma(\mathtt{C})_i)(P(\varsigma(\mathtt{C})_j)(x))$$

Example: the power function 2^2. A valid interpretation is

$$\begin{pmatrix} 2 & 2 & 2 \\ 0 & 1 & 2 \\ 1 & 2 & 4 \\ 1 & 1 & 2 \end{pmatrix}$$

Recall a^b is $\phi_{\text{pow}} = \text{baseCase}(X) \vee \text{indStep}(X)$ with

$$\begin{aligned} \text{baseCase}(X) &= (\mathtt{pow}_2(X) = 0) \wedge (\mathtt{pow}_3(X) = 1), \\ \text{indStep}(X) &= \mathtt{pow}_1(X) = \mathtt{pow}_1(\mathtt{pow}_4(X)) \quad \wedge \\ &\quad\ \mathtt{pow}_2(X) = \mathtt{pow}_2(\mathtt{pow}_4(X)) + 1 \quad \wedge \\ &\quad\ \mathtt{pow}_3(X) = \mathtt{pow}_1(X) \cdot \mathtt{pow}_3(\mathtt{pow}_4(X)) \end{aligned}$$

Recall: *Arithmetising Logic : Polynomial Semantics of FOL* **[Gab25]**

Applying the semantics to $\mathrm{baseCase}(X) \vee \mathrm{indStep}(X)$:

$$\begin{aligned}\langle \phi_{\mathrm{pow}} \rangle_{\varsigma}^{x} = &\big(P(0,1,2)(x)^2 + (P(1,2,4)(x) - 1)^2\big) * \Big(\big(P(2,2,2)(x) - P(2,2,2)(P(1,1,2)(x))\big)^2 \\ &+ \big(P(0,1,2)(x) - (P(0,1,2)(P(1,1,2)(x)) + 1)\big)^2 \\ &+ \big(P(1,2,4)(x) - P(2,2,2)(x) * P(1,2,4)(P(1,1,2)(x))\big)^2\Big)\end{aligned}$$

Recall: *Arithmetising Logic : Polynomial Semantics of FOL* **[Gab25]**

Applying the semantics to $\mathrm{baseCase}(X) \vee \mathrm{indStep}(X)$:

$$\begin{aligned}\langle \phi_{\mathrm{pow}} \rangle_{\varsigma}^{x} = &\big(P(0,1,2)(x)^2 + (P(1,2,4)(x) - 1)^2\big) * \Big(\big(P(2,2,2)(x) - P(2,2,2)(P(1,1,2)(x))\big)^2 \\ &+ \big(P(0,1,2)(x) - (P(0,1,2)(P(1,1,2)(x)) + 1)\big)^2 \\ &+ \big(P(1,2,4)(x) - P(2,2,2)(x) * P(1,2,4)(P(1,1,2)(x))\big)^2\Big)\end{aligned}$$

Then $2^2 = 4$ if and only if the above polynomial has zeroes at $x = 1, 2,$ and 3.

Recall: *Arithmetising Logic : Polynomial Semantics of FOL* **[Gab25]**

Applying the semantics to $\mathrm{baseCase}(X) \vee \mathrm{indStep}(X)$:

$$\begin{aligned}\langle \phi_{\mathrm{pow}} \rangle_{\varsigma}^{x} = &\big(P(0,1,2)(x)^2 + (P(1,2,4)(x) - 1)^2\big) * \Big(\big(P(2,2,2)(x) - P(2,2,2)(P(1,1,2)(x))\big)^2 \\ &+ \big(P(0,1,2)(x) - (P(0,1,2)(P(1,1,2)(x)) + 1)\big)^2 \\ &+ \big(P(1,2,4)(x) - P(2,2,2)(x) * P(1,2,4)(P(1,1,2)(x))\big)^2\Big)\end{aligned}$$

Then $2^2 = 4$ if and only if the above polynomial has zeroes at $x = 1, 2,$ and 3.

E.g. $x = 2$:

$$\langle \phi_{\mathrm{pow}} \rangle_{\varsigma}^{2} = \big(1^2 + (2-1)^2\big) * \Big(\big(2-2\big)^2 + \big(1 - (0+1)\big)^2 + \big(2 - 2 * 1\big)^2\Big) = 0$$

There may be more efficient ways of verifying that $2^2 = 4$.

Our Work: the Cryptographic Side

Instead of Lagrange interpolation, we want to use multilinear extensions over the hypercube.

Define quad-indexed variable symbols $B_{i,\nu}^{j,x}$, where $i, j, x, \nu \in \mathbb{N}$.

Our Work: the Cryptographic Side

Instead of Lagrange interpolation, we want to use multilinear extensions over the hypercube.

Define quad-indexed variable symbols $B_{i,\nu}^{j,x}$, where $i, j, x, \nu \in \mathbb{N}$.

Define $mkQ^x : \mathsf{EP} \to \mathbb{Z}\left[B_{1,1}^{0,1}, \ldots, B_{1,\mu}^{0,1}, \ldots, B_{ar(\mathtt{C}),1}^{ar(\mathtt{C}),len(\varsigma(\mathtt{C}))}, \ldots, B_{ar(\mathtt{C}),\mu}^{ar(\mathtt{C}),len(\varsigma(\mathtt{C}))}\right]$:

$$mkQ^x(q) = q \quad mkQ^x(P + P') = mkQ^x(P) + mkQ^x(P') \quad mkQ^x(P * P') = mkQ^x(P) * mkQ^x(P')$$
$$mkQ^x(X) = x \quad mkQ^x(\mathtt{C}_i(X)) = b2int(B_{i,1}^{0,x}, \ldots, B_{i,\mu}^{0,x}) \quad mkQ^x(\mathtt{C}_i\mathtt{C}_j(X)) = b2int(B_{i,1}^{j,x}, \ldots, B_{i,\mu}^{j,x})$$

Our Work: the Cryptographic Side

Instead of Lagrange interpolation, we want to use multilinear extensions over the hypercube.

Define quad-indexed variable symbols $B_{i,\nu}^{j,x}$, where $i, j, x, \nu \in \mathbb{N}$.

Define $mkQ^x : \mathsf{EP} \to \mathbb{Z}\left[B_{1,1}^{0,1}, \ldots, B_{1,\mu}^{0,1}, \ldots, B_{ar(\mathtt{C}),1}^{ar(\mathtt{C}),len(\varsigma(\mathtt{C}))}, \ldots, B_{ar(\mathtt{C}),\mu}^{ar(\mathtt{C}),len(\varsigma(\mathtt{C}))}\right]$:

$$mkQ^x(q) = q \quad mkQ^x(P + P') = mkQ^x(P) + mkQ^x(P') \quad mkQ^x(P * P') = mkQ^x(P) * mkQ^x(P')$$
$$mkQ^x(X) = x \quad mkQ^x(\mathtt{C}_i(X)) = b2int(B_{i,1}^{0,x}, \ldots, B_{i,\mu}^{0,x}) \quad mkQ^x(\mathtt{C}_i\mathtt{C}_j(X)) = b2int(B_{i,1}^{j,x}, \ldots, B_{i,\mu}^{j,x})$$

Example: applying mkQ^x to $\phi_{\text{pow}} = \text{baseCase}(X) \vee \text{indStep}(X)$:

$$\begin{aligned} mkQ^x\langle\phi_{\text{pow}}\rangle = &\big(b2int(B_{2,1}^{0,x}, \ldots, B_{2,\mu}^{0,x})^2 + (b2int(B_{3,1}^{0,x}, \ldots, B_{3,\mu}^{0,x}) - 1)^2\big) \\ &* \Big(\big(b2int(B_{1,1}^{0,x}, \ldots, B_{1,\mu}^{0,x}) - b2int(B_{1,1}^{4,x}, \ldots, B_{1,\mu}^{4,x})\big)^2 \\ &+ \big(b2int(B_{2,1}^{0,x}, \ldots, B_{2,\mu}^{0,x}) - \Big(b2int(B_{2,1}^{4,x}, \ldots, B_{2,\mu}^{4,x}) + 1\Big)\big)^2 \\ &+ \big(b2int(B_{3,1}^{0,x}, \ldots, B_{3,\mu}^{0,x}) - b2int(B_{2,1}^{0,x}, \ldots, B_{2,\mu}^{0,x}) * b2int(B_{3,1}^{4,x}, \ldots, B_{3,\mu}^{4,x})\big)^2\Big) \end{aligned}$$

Our Work: the Cryptographic Side

Let $\mathcal{F} = \left\{ \tilde{f}^i_{\varsigma(\mathtt{C})} : i \in [ar(\mathtt{C})] \right\}$ denote the MLEs of the rows of $\varsigma(\mathtt{C})$.

Define $\beta_{\mathcal{F}}(B^{0,x}_{i,\nu})$ and $\beta_{\mathcal{F}}(B^{j,x}_{i,\nu})$ by

$$\beta_{\mathcal{F}}(B^{0,x}_{i,\nu}) = \pi_\nu \tilde{f}^i_{\varsigma(\mathtt{C})}(\mathbb{0}_{\mathtt{b}} x) \quad \text{and} \quad \beta_{\mathcal{F}}(B^{j,x}_{i,\nu}) = \begin{cases} \pi_\nu \tilde{f}^i_{\varsigma(\mathtt{C})}(\mathbb{0}_{\mathtt{b}} \tilde{f}^j_{\varsigma(\mathtt{C})}(\mathbb{0}_{\mathtt{b}} x)) & \text{if } f_j(x) \in [len(\varsigma(\mathtt{C}))] \\ 0 & \text{if } f_j(x) \notin [len(\varsigma(\mathtt{C}))] \end{cases}$$

Our Work: the Cryptographic Side

Let $\mathcal{F} = \left\{ \tilde{f}^i_{\varsigma(\mathtt{C})} : i \in [ar(\mathtt{C})] \right\}$ denote the MLEs of the rows of $\varsigma(\mathtt{C})$.

Define $\beta_{\mathcal{F}}(B^{0,x}_{i,\nu})$ and $\beta_{\mathcal{F}}(B^{j,x}_{i,\nu})$ by

$$\beta_{\mathcal{F}}(B^{0,x}_{i,\nu}) = \pi_\nu \tilde{f}^i_{\varsigma(\mathtt{C})}(\mathbb{0}_{\mathtt{b}} x) \quad \text{and} \quad \beta_{\mathcal{F}}(B^{j,x}_{i,\nu}) = \begin{cases} \pi_\nu \tilde{f}^i_{\varsigma(\mathtt{C})}(\mathbb{0}_{\mathtt{b}} \tilde{f}^j_{\varsigma(\mathtt{C})}(\mathbb{0}_{\mathtt{b}} x)) & \text{if } f_j(x) \in [len(\varsigma(\mathtt{C}))] \\ 0 & \text{if } f_j(x) \notin [len(\varsigma(\mathtt{C}))] \end{cases}$$

Key result:

$$\beta_{\mathcal{F}}(mkQ^x\langle t \rangle) = \langle t \rangle^x_\varsigma \quad \text{and} \quad \beta_{\mathcal{F}}(mkQ^x\langle \phi \rangle) = \langle \phi \rangle^x_\varsigma$$

Our Work: the Cryptographic Side

Let $\mathcal{F} = \left\{ \tilde{f}^i_{\varsigma(\mathtt{C})} : i \in [ar(\mathtt{C})] \right\}$ denote the MLEs of the rows of $\varsigma(\mathtt{C})$.

Define $\beta_{\mathcal{F}}(B^{0,x}_{i,\nu})$ and $\beta_{\mathcal{F}}(B^{j,x}_{i,\nu})$ by

$$\beta_{\mathcal{F}}(B^{0,x}_{i,\nu}) = \pi_\nu \tilde{f}^i_{\varsigma(\mathtt{C})}(\mathbb{0}_{\mathtt{b}} x) \quad \text{and} \quad \beta_{\mathcal{F}}(B^{j,x}_{i,\nu}) = \begin{cases} \pi_\nu \tilde{f}^i_{\varsigma(\mathtt{C})}(\mathbb{0}_{\mathtt{b}} \tilde{f}^j_{\varsigma(\mathtt{C})}(\mathbb{0}_{\mathtt{b}} x)) & \text{if } f_j(x) \in [len(\varsigma(\mathtt{C}))] \\ 0 & \text{if } f_j(x) \notin [len(\varsigma(\mathtt{C}))] \end{cases}$$

Key result: $$\beta_{\mathcal{F}}(mkQ^x\langle t \rangle) = \langle t \rangle^x_\varsigma \quad \text{and} \quad \beta_{\mathcal{F}}(mkQ^x\langle \phi \rangle) = \langle \phi \rangle^x_\varsigma$$

What does this mean?

Recall: $\langle \mathtt{C}_i(X) \rangle^x_\varsigma = \varsigma(\mathtt{C})_{@i,x}$, and $\langle \mathtt{C}_i\mathtt{C}_j(X) \rangle^x_\varsigma = \varsigma(\mathtt{C})_{@i,\varsigma(\mathtt{C})_{@j,x}}$

The mkQ^x are multivariate polynomials in the binary decompositions of the entries of $\varsigma(\mathtt{C})$, preserving the logical content of predicates.

What's the Point of all this Symbol Salad?

We can turn (ϕ, ς) into an AIR encoding logical properties.

$\beta_{\mathcal{F}}(mkQ^x\langle\phi\rangle) = 0$ is just $mkQ^x\langle\phi\rangle$ evaluating to zero on

$$(\mathfrak{V}_{\mathtt{b}}(\varsigma(\mathtt{C})_{@1,1}), \ldots, \mathfrak{V}_{\mathtt{b}}(\varsigma(\mathtt{C})_{@ar(\mathtt{C}),len(\varsigma(\mathtt{C}))}))$$

What's the Point of all this Symbol Salad?

We can turn (ϕ, ς) into an AIR encoding logical properties.

$\beta_{\mathcal{F}}(mkQ^x\langle\phi\rangle) = 0$ is just $mkQ^x\langle\phi\rangle$ evaluating to zero on

$$(\mathfrak{N}_{\mathtt{b}}(\varsigma(\mathtt{C})_{@1,1}), \ldots, \mathfrak{N}_{\mathtt{b}}(\varsigma(\mathtt{C})_{@ar(\mathtt{C}),len(\varsigma(\mathtt{C}))}))$$

Then

$$(\mathbb{x}, \mathbb{w}) \in \left\{ \; (\phi, \varsigma) \;\middle|\; \begin{array}{l} \varsigma(\mathtt{C}) \in \mathcal{M}_{ar(\mathtt{C}) \times len(\varsigma(\mathtt{C}))}(\mathbb{N}) \\ \mathtt{C} \vDash_{\varsigma} \phi \end{array} \right\} \iff$$

$$\left((mkQ^x\langle \mathbb{x} \rangle)_x, (\tilde{f}^i_{\mathbb{w}})_i\right) \in \left\{ \; \left((mkQ^x\langle\phi\rangle)_{x \in [len(\varsigma(\mathtt{C}))]}, (\tilde{f}^i_{\varsigma(\mathtt{C})})_{i \in [ar(\mathtt{C})]}\right) \;\middle|\; \begin{array}{l} \varsigma(\mathtt{C}) \in \mathcal{M}_{ar(\mathtt{C}) \times len(\varsigma(\mathtt{C}))}(\mathbb{N}), \\ \beta_{\mathcal{F}}(mkQ^x\langle\phi\rangle) = 0, \\ \qquad \forall x \in [len(\varsigma(\mathtt{C}))] \end{array} \right\}$$

What's the point of all this Symbol Salad?

We turn this into an AIR:

Given (ϕ, ς), compute $\left((mkQ^x\langle\phi\rangle)_{x\in[len(\varsigma(\mathtt{C}))]}, (\tilde{f}^i_{\varsigma(\mathtt{C})})_{i\in[ar(\mathtt{C})]}\right)$

Set

$$(\mathbb{x}, \mathbb{w}) = \left(([\tilde{f}^1_{\varsigma(\mathtt{C})}], \ldots, [\tilde{f}^{ar(\mathtt{C})}_{\varsigma(\mathtt{C})}]), (\tilde{f}^1_{\varsigma(\mathtt{C})}, \ldots, \tilde{f}^{ar(\mathtt{C})}_{\varsigma(\mathtt{C})})\right)$$

and

$$\mathcal{Q} = \left((mkQ^x\langle\phi\rangle)_{x\in[len(\varsigma(\mathtt{C}))]}, (B^{j,x}_{i,\nu}(B^{j,x}_{i,\nu} - 1))_{i,j,x,\nu}\right)$$

What's the point of all this Symbol Salad?

We turn this into an AIR:

Given (ϕ, ς), compute $\left((mkQ^x\langle\phi\rangle)_{x\in[len(\varsigma(\mathtt{C}))]}, (\tilde{f}^i_{\varsigma(\mathtt{C})})_{i\in[ar(\mathtt{C})]}\right)$

Set

$$(\mathbb{x}, \mathbb{w}) = \left(([\tilde{f}^1_{\varsigma(\mathtt{C})}], \ldots, [\tilde{f}^{ar(\mathtt{C})}_{\varsigma(\mathtt{C})}]), (\tilde{f}^1_{\varsigma(\mathtt{C})}, \ldots, \tilde{f}^{ar(\mathtt{C})}_{\varsigma(\mathtt{C})})\right)$$

and

$$\mathcal{Q} = \left((mkQ^x\langle\phi\rangle)_{x\in[len(\varsigma(\mathtt{C}))]}, (B^{j,x}_{i,\nu}(B^{j,x}_{i,\nu} - 1))_{i,j,x,\nu}\right)$$

By the theory of the previous slides: $\mathtt{C} \vDash_\varsigma \phi \iff$

$$(\mathbb{x}, \mathbb{w}) \in \mathrm{REL}_{\mathrm{pp},\mathcal{Q}} = \left\{ (\mathbb{x}, \mathbb{w}) \;\middle|\; \begin{array}{l} \mathrm{pp} = (k, \mu, \partial), \\ \mathbb{x} = ([\tilde{f}^1_{\varsigma(\mathtt{C})}], \ldots, [\tilde{f}^{ar(\mathtt{C})}_{\varsigma(\mathtt{C})}]) \\ \mathbb{w} = (\tilde{f}^1_{\varsigma(\mathtt{C})}, \ldots, \tilde{f}^{ar(\mathtt{C})}_{\varsigma(\mathtt{C})}), \\ Q\left(\left(\tilde{f}^1_{\varsigma(\mathtt{C})}(\mathbf{x}), \ldots, \tilde{f}^{ar(\mathtt{C})}_{\varsigma(\mathtt{C})}(\mathbf{x})\right)_{\mathbf{x}\in\{0,1\}^\mu}\right) = 0, \\ \quad \text{for all } Q \in \mathcal{Q} \end{array} \right\}$$

Conclusions

- First-order logic can specify (Turing-complete) computation and correctness properties thereof (SK combinators).

Conclusions

- First-order logic can specify (Turing-complete) computation and correctness properties thereof (SK combinators).
- But we can do much more! We arithmetise logic, not a model of computation.

Conclusions

- First-order logic can specify (Turing-complete) computation and correctness properties thereof (SK combinators).
- But we can do much more! We arithmetise logic, not a model of computation.
- E.g. “Function f computes the square root” or “Abstract machine A, when given input x, returns output $f(x)$” or “Transaction t is valid according to the rules of the system” or “Proposed state-change S is compliant with legal requirement Z”, etc.

Conclusions

- First-order logic can specify (Turing-complete) computation and correctness properties thereof (SK combinators).
- But we can do much more! We arithmetise logic, not a model of computation.
- E.g. “Function f computes the square root” or “Abstract machine A, when given input x, returns output $f(x)$” or “Transaction t is valid according to the rules of the system” or “Proposed state-change S is compliant with legal requirement Z”, etc.
- Our semantics shows logic is multilinear polynomial manipulation (in some sense) - we very directly turn FOL into polynomial relation checks.

Conclusions

- First-order logic can specify (Turing-complete) computation and correctness properties thereof (SK combinators).
- But we can do much more! We arithmetise logic, not a model of computation.
- E.g. “Function f computes the square root” or “Abstract machine A, when given input x, returns output $f(x)$” or “Transaction t is valid according to the rules of the system” or “Proposed state-change S is compliant with legal requirement Z”, etc.
- Our semantics shows logic is multilinear polynomial manipulation (in some sense) - we very directly turn FOL into polynomial relation checks.
- Technical detail: we get an AIR in the binary decompositions of the witness integers, not in the witness integers - because $\mathbb{0}_b$ is not a polynomial.

IMPERIAL

- Arithmetise directly over finite fields.
- (Non-toy) Implementation.

IMPERIAL

- Arithmetise directly over finite fields.
- (Non-toy) Implementation.

Thank you.
Questions?

SNARKs
for First Order Logic
09/05/2026