SNARKs for pure logic

Murdoch J. Gabbay

Circle Research Seminar

12 November 2025

Hello

Thanks to the organisers for the invitation to speak. Thank you for coming.

Some background on me:

1. Name: Murdoch (Jamie) Gabbay.
2. 2001: PhD in mathematical computer science, Cambridge, UK.
3. 2006–present: Professor (lecturer) at Heriot-Watt University in Edinburgh, UK (nearly 20 years).
4. 2019: Alonzo Church Award for Outstanding Contributions to Logic and Computation.
5. Approx 90 journal & conference publications, some prizewinning.
6. 2024: Published book introducing *Semitopologies*.
7. 2018–present: researcher in blockchain space, working with Ethereum, Starkware, Tezos, IOG, and others.
8. Nov 2025: invited to give a Circle Research seminar!

Recent research

I've been working in distributed algorithms (think: Paxos, byzantine generals). I *won't* discuss this today but if you want to discuss properties of distributed algorithms, let me know!

If there's a theme to my research, it's taking ideas and techniques from one field and applying them to another. Sometimes you can do a lot just by transferring undergraduate-level insights between fields.

In this talk I will sketch one such transfer for your consideration, between logic and cryptography.

Arithmetisation

To arithmetise a computational task is to convert it into a task of finding roots of some polynomial.

Arithmetisation is the essential first step for cryptographically verifiable logic and computation: using cryptographic techniques to prove that assertions hold, without redoing the underlying computation or proof.

Think: weak *verifier* verifies correctness of computation performed by strong *prover*.

Arithmetisation is now a billion-dollar industry. This has sprung up only very recently; roughly in the last ten years.

Arithmetisation

In a recent paper (pending publication) *"Arithmetisation of computation via polynomial semantics for first-order logic"* (https://eprint.iacr.org/2024/954) I arithmetise first-order logic via a compositional shallow translation to assertions about roots of polynomials.

In this talk I'll give a flavour of how this works.

This will be an accessible presentation. If you want technical detail, please ask me after. We will use just two elementary lemmas.

Lemma 1 (undergraduate level)

Write $1..n$ for the set $\{x \in \mathbb{N} \mid 1 \leq x \leq n\}$.

Lemma 1. Suppose $\mathbf{v} \in \mathbb{Q}^n$ is an n-tuple of rationals. Then there exists a unique lowest-degree interpolating polynomial

$$ip(\mathbf{v}) \in \mathbb{Q}[X]^n$$

such that

$$\forall x \in 1..n.\ ip(\mathbf{v})(x) = \mathbf{v}[x]$$

where $\mathbf{v}[x]$ is the xth element of $\mathbf{v}$.

Example. $ip(1.5, 2, 2.5, 3) = 1 + 0.5 * X$.

See diagram on Wikipedia.

Lemma 2 (school level)

Lemma 2. Suppose $x, y \in \mathbb{Q}_{\geq 0}$ (nonnegative rationals). Then:

1. $x + y = 0$ iff $x = 0 \wedge y = 0$.
2. $x * y = 0$ iff $x = 0 \vee y = 0$.
3. $(x - y)^2 = 0$ iff $x = y$.

Proof. Facts of arithmetic.

Observation 3.

1. $P \wedge Q = \top$ iff $P = \top \wedge Q = \top$.
2. $P \vee Q = \top$ iff $P = \top \vee Q = \top$.
3. $(x = y) = \top$ iff $x = y$.

Lemma 2 + Observation suggest that we can treat $\mathbb{Q}_{\geq 0}$ as a domain of truth-values, such that

- zero represents 'true' and
- non-zero represent 'false'.

Enriched polynomials

Definition 4. Fix some data:

- An index variable symbol X.
- Polynomial symbols $\mathbb{F} \in \mathrm{PolySymb}$, each with arity $\mathit{arity}(\mathbb{F}) \in \mathbb{N}_{\geq 1}$.

Write $\mathbb{F} : n$ for "$\mathbb{F} \in \mathrm{PolySymb}$ and $\mathit{arity}(\mathbb{F}) = n$".

Define enriched (rational) polynomials (EPs) by:

$$P ::= X \mid q \in \mathbb{Q} \mid P + P \mid P * P \mid \mathbb{F}_i(P) \quad (1 \leq i \leq \mathit{arity}(\mathbb{F}))$$

In English: X and q are EPs; if P and P' are EPs then so are $P + P'$ and $P * P'$ and $\mathbb{F}_i(P)$.

Example. $X + \mathbb{F}_2(X) * \mathbb{F}_1(X)$ is an EP, where $\mathbb{F} : 2$.

Enriched polynomials are ordinary polynomials, with binary syntax-formers $+$ and $*$ *and* some unary syntax-formers $\mathbb{F}_i$.

Enriched polynomials

We can turn an enriched polynomial into an ordinary polynomial:

- interpret each $\mathbb{F}$ with $\mathit{arity}(\mathbb{F}) = n$ as
- an n-tuple $\varsigma(\mathbb{F})=(F_1, \ldots, F_n)$ of (unenriched) polynomials, and
- we set $\mathbb{F}_i(P)_\varsigma = F_i(P)$ (composition of F_i with P) for $i \in 1..n$.

For example:

- Consider $X + \mathbb{F}_1(1 * X) * \mathbb{F}_2(2 * X)$.
- Define $\varsigma(\mathbb{F}) = (F_1, F_2) = (1 + X, 1 - X)$.
- Obtain $$\begin{aligned}(X + \mathbb{F}_1(1 * X) * \mathbb{F}_2(2 * X))_\varsigma &= \\ X + F_1(1 * X) * F_2(2 * X) &= \\ X + (1 + X) * (1 - 2 * X)\end{aligned}$$

Why we will want to do this will become clear shortly.

A simple propositional logic

For this talk, we will consider a simple but expressive logical language. More powerful logics are possible but this illustrates the idea:

Definition 5. Define terms and predicates by:

$$t ::= X \mid q \in \mathbb{Q} \mid t + t \mid t * t \mid \mathbb{F}_i(t) \quad (i \in 1..\mathit{arity}(\mathbb{F}))$$
$$\phi ::= t{=}t \mid \phi \wedge \phi \mid \phi \vee \phi$$

In English: X is a term; q is a term; if t and t' are terms then so are $t + t'$ and $t * t'$ and $\mathbb{F}_i(t)$; if t and t' are terms then $t{=}t'$ is a predicate; if ϕ and ϕ' are predicates then so are $\phi \wedge \phi'$ and $\phi \vee \phi'$.

Perhaps surprisingly, this logical syntax suffices for Turing-complete computation and its correctness properties.

We'll compile to enriched polynomials in the next slide, then illustrate how to encode computation.

Compiling syntax to enriched polynomials

Definition 6 (denotation in enriched polynomials).

$$[X] = X \qquad [q] = q \quad (q \in \mathbb{Q})$$
$$[t + t'] = [t] + [t'] \qquad [t * t'] = [t] * [t']$$
$$[\mathbb{F}_i(t)] = \mathbb{F}_i([t])$$
$$[t{=}t'] = ([t] - [t'])^2$$
$$[\phi \wedge \phi'] = [\phi] + [\phi'] \qquad [\phi \vee \phi'] = [\phi] * [\phi']$$

This compilation will give enriched polynomials logical expressivity.

Note that we can chain **[-]** with the interpretation $()_\varsigma$ of the polynomial symbols as n-tuples of (ordinary) polynomials. Given a predicate ϕ or term t, and an *interpretation* ς, we obtain an ordinary polynomial $[\phi]_\varsigma$ or $[t]_\varsigma$.

Interpretation

Definition. An interpretation ς is a function mapping

- an n-ary polynomial symbol $\mathtt{F} : n$ to
- an $n \times l$ matrix $\varsigma(\mathtt{F}) \in \mathbb{Q}^{n \times l}$.

Write $\mathit{len}(\varsigma(\mathtt{F}))$ for l (number of columns). The value of n is fixed at $\mathit{arity}(\mathtt{F})$; the value of l varies with the choice of ς.

Equivalently by Lemma 1, ς maps $\mathtt{F}$ to an n-tuple of interpolating polynomials. So this is just a tabular presentation of ς from before.

Example. Assume $\mathtt{Sqr} : 2$ and $\mathtt{Fact} : 3$ and set:

$$\varsigma(\mathtt{Sqr}) = \begin{pmatrix} 1 & 2 & \text{-}3 \\ 1 & 4 & 9 \end{pmatrix} \qquad \varsigma(\mathtt{Fact}) = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 1 & 2 & 6 & 24 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix}$$

Equivalently (LH matrix) $\varsigma(\mathtt{Sqr}) = (\mathit{ip}(1, 2, \text{-}3), \mathit{ip}(1, 4, 9))$.

Matrices as logic & computation

$$\varsigma(\texttt{Sqr}) = \begin{pmatrix} 1 & 2 & \text{-}3 \\ 1 & 4 & 9 \end{pmatrix} \qquad \varsigma(\texttt{Fact}) = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 1 & 2 & 6 & 24 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix}$$

Note something interesting about our interpretation:

1. Row 2 of $\varsigma(\texttt{Sqr})$ is the square of row 1.
2. Row 2 of $\varsigma(\texttt{Fact})$ is the factorial of row 1.
3. $\varsigma(\texttt{Fact})$ has a third row. This seems to hold a 'pointer' to a column with a relevant inductive step.
   $4! = 4 * 3!$ and $\varsigma(\texttt{Fact})[3, 5] = 4$.
   $3! = 3 * 2!$ and $\varsigma(\texttt{Fact})[3, 4] = 3$.
   $2! = 2 * 1!$ and $\varsigma(\texttt{Fact})[3, 3] = 2$.
   $1! = 1 * 0!$ and $\varsigma(\texttt{Fact})[3, 2] = 1$.
   Hmmm. What's happening?

A logical observation

$$\varsigma(\texttt{Sqr}) = \begin{pmatrix} 1 & 2 & \text{-}3 \\ 1 & 4 & 9 \end{pmatrix} \quad \text{and} \quad \varsigma(\texttt{Fact}) = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 1 & 2 & 6 & 24 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix}$$

validates the following predicates, for X taking values $x \in 1..\mathit{len}(\varsigma(\texttt{Sqr}))$ and $x \in 1..\mathit{len}(\varsigma(\texttt{Fact}))$ respectively:

$$\texttt{Sqr}_2(X) = \texttt{Sqr}_1(X) * \texttt{Sqr}_1(X)$$

$$\begin{aligned} (\texttt{Fact}_1(X) &= 0 \wedge \texttt{Fact}_2(X) = 1) \vee \\ (\texttt{Fact}_2(X) &= \texttt{Fact}_1(X) * \texttt{Fact}_2(\texttt{Fact}_3(X))) \end{aligned}$$

Unpacking a bit:

$$\varsigma(\texttt{Sqr})[2, x] = \varsigma(\texttt{Sqr})[1, x] * \varsigma(\texttt{Sqr})[1, x]$$

$$\begin{aligned} (\varsigma(\texttt{Fact})[1, x] &= 0 \wedge \varsigma(\texttt{Fact})[2, x] = 1) \vee \\ (\varsigma(\texttt{Fact})[2, x] &= \varsigma(\texttt{Fact})[1, x] * \varsigma(\texttt{Fact})[2, \varsigma(\texttt{Fact})[3, x]]) \end{aligned}$$

Soundness and completeness

Definition 7 (polynomial roots).
Write $\models_\varsigma \phi$ when $[\![\phi]\!]_\varsigma = 0$ (ϕ closed; no X).
Write $\mathtt{F} \models_\varsigma \phi$ when $[\![\phi]\!]_\varsigma(x) = 0$ for $x \in 1..\mathit{len}(\varsigma(\mathtt{F}))$.

Proposition 8.

1. $\models_\varsigma t{=}t'$ iff $[\![t]\!]_\varsigma = [\![t']\!]_\varsigma$ (t closed).
2. $\models_\varsigma \phi{\wedge}\phi'$ iff $\models_\varsigma \phi \wedge \models_\varsigma \phi'$ (ϕ, ϕ' closed).
3. $\models_\varsigma \phi{\vee}\phi'$ iff $\models_\varsigma \phi \vee \models_\varsigma \phi'$ (ϕ, ϕ' closed).
4. $\mathtt{F} \models_\varsigma \phi$ iff $\models_\varsigma \phi[X{:=}x]$ for every $x \in 1..\mathit{len}(\varsigma(\mathtt{F}))$.

Proof. Just from Lemma 2.

Now we can reexpress our logical observation as an assertion about roots of polynomials!

$$\mathtt{Sqr} \models_\varsigma \left(\mathtt{Sqr}_2(X){=}\mathtt{Sqr}_1(X) * \mathtt{Sqr}_1(X)\right)$$

$$\mathtt{Fact} \models_\varsigma (\mathtt{Fact}_1(X){=}0 \wedge \mathtt{Fact}_2(X){=}1) \vee (\mathtt{Fact}_2(X){=}\mathtt{Fact}_1(X) * \mathtt{Fact}_2(\mathtt{Fact}_3(X)))$$

A flavour of how this can be cryptographically applied

Suppose a *prover* wants to convince a *verifier* that $\mathbb{F} \models_\varsigma \phi$ for some concrete *witnessing* $\varsigma(\mathbb{F})$ with 10^9 columns (approx 1 terabyte).

On the face of it we would have to evaluate ϕ on 10^9 claimed roots.

Alternatively, a prover could compute polynomials $\mathit{zeroes} = \prod_{x \in 1..10^9}(X\text{-}x)$ and $h = [\![\phi]\!]_\varsigma / \mathit{zeroes}$.

The verifier picks a random $q \in \mathbb{Q}$ and forces the prover to produce a cryptographically assured proof that $h(q) * \mathit{zeroes}(q) = [\![\phi]\!]_\varsigma(q)$.

That's *one* polynomial evaluation, instead of 10^9.

By the Schwartz-Zippel lemma, for low-degree polynomials P and Q, equality $P - Q = 0$ is (to a high degree of certainty) the same as equality of their evaluation at a random point $P(x) - Q(x) = 0$, since for low-degree polynomials, $P(x) - Q(x)$ is zero nearly nowhere in the ambient field.

What's going on? How expressive is the logic?

Matrices can express logical derivation trees. Abusing notation a bit, we can write:

$$\text{Matrix} \;=\; \text{Logical derivation tree}$$

Each step in the derivation tree gets a matrix column; the 'pointer trick' allows us to pack tree structure just as a sequence of columns. Does that remind you of techniques from SNARKs?

Derivation trees can encode arbitrary inductive definitions. This includes computationally significant functions like Ackermann's Function, Gödel encodings, and SK-combinator reduction. (cf. paper)

With more fiddling, we can express general recursion, negation, and arbitrarily nested function-definitions. Using a predicate reify, which reifies predicates as terms, we can express arbitrary Skolem functions. The full logic is *very* expressive; surprisingly so.

What's going on? How expressive is the logic?

On the cryptographic side, we have a relation $R = \{(\mathbb{F} \vDash \phi, \varsigma) \mid ....\}$ where $(\mathbb{F} \vDash \phi)$ is an *instance* and ς is a *witness* interpretation such that $\mathbb{F} \vDash_{\varsigma} \phi$.

Logic gets compiled down to polynomials; logical judgements become judgements about zeroes of polynomials; and (so the idea) this all gets plugged in to standard IOP machinery.

Because ϕ can express arbitrary logic and matrices can pack arbitrary proofs/derivations, we can succinctly verify . . . anything!

No explicit zkVM is required, because we are making *direct use of the native logical structure of polynomials*, as per Lemma 2.

Future work: application to concrete schemes

We're working to operationalise this and benchmark it. Assistance would be very welcome.

Practical cryptographic schemes use $\mathbb{F}_p[X]$ for a prime p, instead of $\mathbb{Q}[X]$; or, we can use Zinc (and IOP prover over $\mathbb{Z}$). I'm working on that now, in collaboration.

Cryptographic virtual machines are essentially deep embeddings of VMs in polynomials. Programming on 'virtual chips', laid out in finite fields instead of on silicon. This introduces overhead: you have to load the VM and live within it.

To my knowledge, nobody's thought of doing a shallow embedding of FOL-style validity. This gives a new and direct way to plug the formidable power of logic in computer science — inductive definitions, declarative programming, correctness-by-construction, etc — directly into polynomials as a compilation target, and thence to the equally formidable power of modern cryptographic techniques, with minimal overhead.

More future work

Obvious generalisation is to multivariate polynomials: admit X, X', and Y.

Other direction: what logics naturally correspond to validity assertions on roots of polynomials? I used $[\![\phi]\!]_\varsigma = 0$ in this talk. An obvious thing is to set $\phi \vDash_\varsigma \psi$ when $[\![\phi]\!]_\varsigma \mid [\![\psi]\!]_\varsigma$. This would lead to a resource-conscious logic.

What is the natural logic of square matrices as a (noncommutative) ring? There's an interesting design space here.

https://eprint.iacr.org/2024/954

Happy to collaborate to explore and/or operationalise t

you for listening.