A shallow arithmetisation of first-order validity

Murdoch J. Gabbay

BCTCS

14 April 2025

Arithmetisation of first-order logic

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 cryptographic treatments of logic and computation — as required e.g. for *verifiable* and *zero-knowledge* computation, motivated by modern distributed systems including blockchain (‘weak’ *verifier* verifies ‘strong’ *prover*; shielded transactions).

Arithmetisation is now a billion-dollar industry.

In a recent draft paper *“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.

We will need two elementary Lemmas:

Lemma 1 (a standard result)

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

Lemma 1. Suppose $\vec{v} \in \mathbb{Q}^{len}$ is a tuple of length $len \in \mathbb{N}$. Then there exists a (unique) interpolating polynomial

$$ip(\vec{v}) \in \mathbb{Q}[X]^{<len}$$

(meaning that $ip(\vec{v})$ is a polynomial over X with rational coefficients and degree strictly less than len) such that

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

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

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

See diagram on Wikipedia.

Lemma 2 (an easy arithmetic fact)

Lemma 2. Suppose $x, y \in \mathbb{Q}_{\geq 0}$. 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.

This Lemma, easy as it is, suggests that we can treat $\mathbb{Q}_{\geq 0}$ as a domain of truth-values, such that zero represents 'true' and non-zero values represent 'false'.

A FOL syntax

Fix some data:

- An index variable symbol X.
- A set of polynomial function symbols $\mathtt{F} \in \text{PolyFunc}$, each with arity $\mathit{arity}(\mathtt{F}) \in \mathbb{N}_{\geq 1}$.

Write $\mathtt{F} : n$ for "$\mathtt{F} \in \text{PolyFunc}$ and $\mathit{arity}(\mathtt{F}) = n$".

Then define terms and predicates by:

$$\begin{aligned} t &::= X \mid q \in \mathbb{Q} \mid t + t \mid t * t \mid \mathtt{F}_i(t) \mid \text{len}(\mathtt{F}) \mid \text{reify}(\phi) \\ \phi &::= t\mathbf{=}t \mid \phi\wedge\phi \mid \phi\vee\phi \mid \forall_{\mathtt{F}} X.\phi \end{aligned}$$

Above, $i \in 1..\mathit{arity}(\mathtt{F})$.

Valuations (we'll use them in the next slide)

A valuation ς maps each $\mathtt{F} : n$ to an $n \times l$ matrix $\varsigma(\mathtt{F}) \in \mathbb{Q}^{n \times l}$. Write $\mathit{length}(\varsigma(\mathtt{F}))$ for l (number of columns).

Think of $\varsigma(\mathtt{F})$ as a finite sample from a function graph, possibly with auxiliary information.

Example: Assume $\mathtt{Sqr} : 2$ (length 3) and $\mathtt{Fact}$ (length 5) and set:

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

1. Row 2 is the square/factorial of row 1.
2. For factorial, row 3 uses a 'pointer trick': each entry points to a column with the relevant inductive step, if required. E.g. $4! = 4 * 3!$ and $\varsigma(\mathtt{Fact})[3, 5] = 4$.
3. By Lemma 1, each matrix is representable as a tuple of two/three interpolating polynomials of degree at most 2/4.

Let's characterise these matrices in our logic

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

satisfy the following predicates, for every value of X in $1..\mathit{len}(\varsigma(\texttt{Sqr}))$ and $1..\mathit{len}(\varsigma(\texttt{Fact}))$ respectively:

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

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

Denotation $[\![t]\!]_\varsigma, [\![\phi]\!]_\varsigma \in \mathbb{Q}[X]$

$$[\![X]\!]_\varsigma = X \qquad [\![q]\!]_\varsigma = q \quad (q \in \mathbb{Q})$$

$$[\![t + t']\!]_\varsigma = [\![t]\!]_\varsigma + [\![t']\!]_\varsigma \qquad [\![t * t']\!]_\varsigma = [\![t]\!]_\varsigma * [\![t']\!]_\varsigma$$

$$[\![\mathtt{F}_i(t)]\!]_\varsigma = ip(\varsigma(\mathtt{F}_i))([\![t]\!]_\varsigma) \qquad [\![\mathrm{len}(\mathtt{F})]\!]_\varsigma = length(\varsigma(\mathtt{F}_i))$$

$$[\![\mathrm{reify}(\phi)]\!]_\varsigma = [\![\phi]\!]_\varsigma$$

$$[\![t \mathbf{=} t']\!]_\varsigma = ([\![t]\!]_\varsigma - [\![t']\!]_\varsigma)^2$$

$$[\![\phi \wedge \phi']\!]_\varsigma = [\![\phi]\!]_\varsigma + [\![\phi']\!]_\varsigma \qquad [\![\phi \vee \phi']\!]_\varsigma = [\![\phi]\!]_\varsigma * [\![\phi']\!]_\varsigma$$

$$[\![\forall_{\mathtt{F}} X.\phi]\!]_\varsigma = \sum_{x \in 1..length(\varsigma(\mathtt{F}))} [\![\phi]\!]_\varsigma$$

$$\varsigma \vDash \phi \quad \text{when} \quad [\![\phi]\!]_\varsigma = 0 \quad (\phi \text{ closed})$$

$$t ::= X \mid q \in \mathbb{Q} \mid t + t \mid t * t \mid \mathtt{F}_i(t) \mid \mathrm{len}(\mathtt{F}) \mid \mathrm{reify}(\phi)$$

$$\phi ::= t \mathbf{=} t \mid \phi \wedge \phi \mid \phi \vee \phi \mid \forall_{\mathtt{F}} X.\phi$$

Soundness and completeness

Proposition 3:

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

Proof. Just from Lemma 2.

Now we can express that $\varsigma(\mathtt{Sqr})$ represents a partial graph for squaring, and $\varsigma(\mathtt{Fact})$ for factorial, by asserting:

$$\models_\varsigma \forall_{\mathtt{Sqr}} X.\big(\mathtt{Sqr}_2(X){=}\mathtt{Sqr}_1(X) * \mathtt{Sqr}_1(X)\big)$$

$$\begin{aligned}\models_\varsigma \forall_{\mathtt{Fact}} X.\big((&\mathtt{Fact}_1(X){=}0 \wedge \mathtt{Fact}_2(X){=}1) \vee \\ &(\mathtt{Fact}_2(X){=}\mathtt{Fact}_1(X) * \mathtt{Fact}_2(\mathtt{Fact}_3(X)))\big)\end{aligned}$$

A flavour of how this can be cryptographically applied

(There's much more to it than what's on this slide.)

Suppose a prover wants to convince a verifier that $\vDash_\varsigma \forall_{\mathbb{F}} X.\phi$ for some $\varsigma(\mathbb{F})$ it has computed, 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?

What's going on is that the matrices pack derivation trees. Each node in the tree gets a column, and the 'pointer trick' allows us to reassemble the tree structure. Thus we can encode arbitrary inductive definitions.

We can specify computationally significant functions like Ackermann's Function, Gödel encodings, and SK-combinator reduction.

With more fiddling, we can express general recursion, negation, and arbitrarily nested function-definitions. Using reify, which reifies predicates as terms, we can express arbitrary Skolem functions.

This logic is *very* expressive; surprisingly so.

Future work: application to concrete schemes

This work is impractical as presented, but the ideas could be operationalised. They'd look different: practical cryptographic schemes use $\mathbb{F}_p[X]$ for a prime p, instead of $\mathbb{Q}[X]$.[1] I'm working on that now.

Many companies offer cryptographic virtual machines. All the ones I know of are essentially deep embeddings of VMs in polynomials: programming on 'virtual chips' built in finite fields instead of silicon.

To my knowledge, nobody's thought of doing a shallow embedding of FOL-style validity. This would permit all the goodness of logic in computer science — inductive definitions, high-level programming, correctness-by-construction, etc — directly in a compilation target, which is logical validity encoded in polynomials.

[1] I'm not entirely convinced this is necessary, but it's where the industry is.

Future work

In this talk, I fixed a univariate logic based on the expressivity required to specify functions of interest, and arithmetised it. Obvious generalisation is the multivariate case: admit X, X', and Y.

Also, 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.

Other flavours and variations are possible.

There's an interesting design space of logics here!

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

Much to do. Thank you for listening.