Arithmetisation of first-order logic (zkFOL)

Murdoch J. Gabbay

SPLV lightning talk

27 July 2024

Arithmetisation of first-order logic

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

Many computational tasks can be arithmetised, e.g. the 3-colour problem.

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, by giving a

> *sound and complete compositional shallow mapping from FOL predicates to polynomials.*

Correctness of the construction relies on an elementary observation:

Key Lemma

Lemma. 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.

Intuition: $\mathbb{Q}_{\geq 0}$ as a domain of truth-values, where zero represents 'true' and non-zero values represent 'false'.

FOL denotation in polynomials

Consider FOL with equality, over a finite model with n elements; wlog set this domain to be $\{1, \ldots, n\}$. Let X be a term variable ranging over the domain.

Then we can map FOL predicates to nonnegative univariate polynomials in $\mathbb{Q}[X]$.

$$
\begin{array}{rlrl}
[\![X]\!] &= X & [\![q]\!] &= q \qquad (q \in \mathbb{Q}) \\
\\
[\![\top]\!] &= 0 & [\![\bot]\!] &= 1 \\
[\![t{=}t']\!] &= ([\![t]\!] - [\![t']\!])^2 & & \\
[\![\phi \wedge \phi']\!] &= [\![\phi]\!] + [\![\phi']\!] & [\![\phi \vee \phi']\!] &= [\![\phi]\!] * [\![\phi']\!] \\
[\![\forall X.\phi]\!] &= \sum_{1 \le x \le n} [\![\phi]\!] & [\![\exists X.\phi]\!] &= \prod_{1 \le x \le n} [\![\phi]\!]
\end{array}
$$

Theorem (soundness and completeness): Suppose ϕ is a closed predicate. Then $\vDash \phi$ if and only if $[\![\phi]\!](x) = 0$ for $x \in \{1, \ldots, n\}$.

Sketch of the rest of the paper

1. Encode arbitrary predicate and function constant symbols.
2. Apply this to mathematically-structured programming ($\approx$ inductively defined relations).
3. Encode negation, so we get full FOL (though this is not actually required for many inductive definitions, since these tend to be positive).
4. Leverage methods from cryptography to obtain *correct-by-construction*, *compositional*, highly compact (perhaps *smaller by orders of magnitude than current state of the art*) succinct proofs in the cryptographic sense, of computations specified using FOL.

Happy to discuss: `https://eprint.iacr.org/2024/954`.