My previous post focused on Jolt, a system that shows how each CPU instruction of a zkVM can be implemented using a table lookup. How to actually implement these table lookups is discussed in a companion paper with the title: Unlocking the lookup singularity with Lasso. Here the word singularity refers to a post by Barry Whitehat, who first set out this vision of using only lookup tables to implement CPU instructions.

In his video presentation on Lasso, Justin Thaler starts by debunking a set of common beliefs, also listed in this post. Some of these observations can be seen as a criticism of the STARK approach which is based on FRI, FFTs and Merkle-hashing.

Thaler argues that Jolt and Lasso scale better using multiscalar multiplications-based commitment schemes. As the abstract reads:

"[T]he committed field elements are small, meaning that, no matter how big the field F is, they are all in the set {0, . . . , m}. When using a multi-exponentiation-based commitment scheme, this results in the prover's costs dominated by only O(m + n) group operations (e.g., elliptic curve point additions), plus the cost to prove an evaluation of a multilinear polynomial whose evaluations over the Boolean hypercube are the table entries. This represents a significant improvement in prover costs over prior lookup arguments (e.g., plookup, Halo2's lookups, lookup arguments based on logarithmic derivatives)."

With respect to CAIRO, a zkVM with very few, zk-friendly instructions developed by Starkware, Thaler states that with Jolt, the cost of having many instructions is not a problem, as the evaluation tables of each instruction are decomposable. In addition, learning to program in CAIRO has been reported to be difficult, while staying close to simpler existing instruction sets such as MIPS is an advantage.

It is somewhat surprising that the remainder of the Lasso video has a significant overlap with Chapters 3 and 4 of Thaler's book. When I started reading his book about a year ago, I did not see the connection with SNARKs, so I skipped these chapters. But now I am seeing a completely different picture.

Thaler is abandoning the conventional SNARK (and STARK) approach, which uses univariate polynomials of (very) high degree and includes algorithms such as FFT and protocols such as FRI. Instead, this approach is based on multivariate polynomials with somewhere between 16 to 100 variables, together with the SumCheck protocol [LFKN90], to prove properties about multilinear extensions (the analogue of low-degree extensions).

Additional Observations

- Looking at the prominent position of the SumCheck protocol in his book, it's obvious that Thaler has been contemplating this approach for a long time.
- The HyperPlonk paper, co-authored by Dan Boneh, also prominently features multivariate polynomials and the SumCheck protocol.
- In theoretical computer science there have been other cases in the past where a technique emerged using univariate polynomials, which were later gradually superseded by multivariate polynomials. History repeats itself.
- The Lasso paper isn't easy to understand. It improves on the lookup table technique called Spark, calling the new protocol Surge. It also relies on several other important techniques, such as checking the correctness of memories[BEGKN]; an efficient protocol for evaluating a log-space uniform arithmetic circuit[GKR]; and efficient verification of matrix multiplication by Thaler himself. I might discuss some of these results in the future, but for now we can only scratch the surface.

A Mind-Boggling Concept

One keeps wondering: how is it possible to have the Prover and Verifier agree on using a table of 2128 elements? The abstract states: "[I]f the table t is structured (in a precise sense that we define), then no party needs to commit to t, enabling the use of much larger tables than prior works (e.g., of size 2128 or larger)." And "[f]or m lookups into a table of size n, Lasso's prover commits to just m + n field elements." How is this possible?

A glimpse of an answer may lie in the following: in his paper on matrix multiplication (see also section 4.4.1 of his book), Thaler presents a protocol in which the Prover proves to the Verifier that he computed intermediate values, but without actually sending them. The Prover is computing a matrix M to the power n. But in many settings it is not the actual value of n that matters; the final answer is just a single number derived from it. In other words, the answer needs to be extracted (in a provably correct manner) from this intermediate value. This is a mind-boggling concept: the Verifier is convinced that the Prover computed this intermediate value correctly without ever seeing (or materializing) them.

Paradoxically, even though I think the level of abstraction of Lasso is higher than of SNARKs, I believe that in the end the underlying theory is simpler. This is because current SNARKs rely heavily on protocols whose security parameters are based on very intricate properties and conjectures about Reed-Solomon codes.

Lasso seems simpler in comparison. In addition, the use of lookup tables to implement CPU instructions will lead to higher-quality, less error-prone zkVMs. For these reasons, I have high expectations about the long-term success of the Lasso approach.