A simple multivariate AIR argument inspired by *SuperSpartan*

William Borgeaud

Introduction

Customizable constraint system (CCS) is a new arithmetization introduced in [1] that generalizes commonly used arithmetizations such as R1CS, Plonk, or AIR. A general purpose proving system for CCS called *SuperSpartan* is described in the same paper. In this short note, we distill the resulting proving system for AIR and slightly simplify it.

AIR and Multi-Column CCS (MCCCS)

We will consider the following basic version of AIR. Let $\mathbb{F}$ be a finite field and let $F \in \mathbb{F}[X_0, \dots, X_{2C-1}]$ be a constraint polynomial. An AIR witness for this instance is a table of C columns $z_0, \dots, z_{C-1}$ of size $n = 2^v$, such that for all $i = 0, \dots, n-2$,

$$F(z_0[i], \dots, z_{C-1}[i], z_0[i+1], \dots, z_{C-1}[i+1]) = 0.$$

A CCS structure [1] is given by

- Matrices $M_0, \dots, M_{t-1} \in \mathbb{F}^{m \times n}$.
- Multisets $S_0, \dots, S_{q-1}$ with elements in $\{0, \dots, t-1\}$.
- Constants $c_0, \dots, c_{q-1} \in \mathbb{F}$.

A witness is a vector $z \in \mathbb{F}^n$ such that[1]

$$\sum_{i=0}^{q-1} c_i \cdot \bigcirc_{j \in S_i} M_j \cdot z = \mathbf{0}.$$

We introduce *Multi-Column CCS* (MCCCS), a generalization of CCS where the witness is structured as a table of C columns. The additional information required is

- A function $c : \{0, \dots, t-1\} \to \{0, \dots, C-1\}$ assigning a column to each matrix.

[1]Disregarding public inputs for now.

A witness is then a table of C columns $z_0, \ldots, z_{C-1} \in \mathbb{F}^n$ such that

$$\sum_{i=0}^{q-1} c_i \cdot \bigcirc_{j \in S_i} M_j \cdot z_{c(j)} = \mathbf{0}.$$

It is easy to see that MCCCS is equivalent to CCS by flattening/unflattening the columns.

Proving system for MCCCS

The SuperSpartan proving system can easily be modified to accommodate MCCCS. The only difference is that the prover commits to all MLE $\widetilde{Z}_j(X)$ of the columns and then equation (14) in [1] becomes

$$0 = \sum_{a \in \{0,1\}^{\log m}} \widetilde{\mathrm{eq}}(\tau, a) \cdot \sum_{i=0}^{q-1} c_i \cdot \prod_{j \in S_i} \left(\sum_{y \in \{0,1\}^{\log n}} \widetilde{M}_j(a, y) \widetilde{Z}_{c(j)}(y) \right),$$

which can be proved with multiple calls to the sumcheck protocol.

AIR as a MCCCS

The AIR arithmetization is shown to be a special case of CCS in [1]. We show here that AIR can also be seen as a MCCCS with a very simple transformation.
Consider the AIR constraint polynomial $F \in \mathbb{F}[X_0, \ldots, X_{2C-1}]$, let $S_0, \ldots, S_{q-1}$ be multisets with values in $\{0, \ldots, 2C-1\}$ representing the monomials of F, and let $c_0, \ldots, c_{q-1}$ be the coefficients of these monomials, i.e.,

$$F(X_0, \ldots, X_{2C-1}) = \sum_{i=0}^{q-1} c_i \prod_{j \in S_i} X_j.$$

Let I_{n-1} be the identity matrix of size $n-1$, let A_0 be the $n \times n$ matrix

$$A_0 = \begin{bmatrix} I_{n-1} & 0 \\ 0 & 0 \end{bmatrix}$$

and let A_1 be the $n \times n$ matrix

$$A_1 = \begin{bmatrix} 0 & I_{n-1} \\ 0 & 0 \end{bmatrix}$$

Note that for A_0 and A_1 the last row is only used for padding so that the matrix is square.
Finally, let $t = 2C$, $M_j = A_{\lfloor j/C \rfloor}$, and $c(j) = j \bmod C$. Then the MCCCS with these parameters is equivalent to the original AIR.

Example

If $C = 2$ and $F = X_0X_1^2 - X_2 - 2X_1X_3$, the MCCCS constraint is

$$A_0z_0 \circ A_0z_1 \circ A_0z_1 - A_1z_0 - 2 \cdot A_0z_1 \circ A_1z_1 = \mathbf{0}.$$

Proving system for AIR

The MLE of the matrices A_0 and A_1 are simple. We have

$$A_0[i,j] = \mathrm{eq}(i,j) - \mathrm{eq}(i,n-1) \cdot \mathrm{eq}(j,n-1)$$

and

$$A_1[i,j] = \mathrm{next}(i,j),$$

where $\mathrm{next}(i,j) = \mathrm{eq}(i+1,j)$ is defined in Section 5.1 of [1]. Therefore the MLE of A_0 is

$$\widetilde{A_0}(x,y) = \widetilde{\mathrm{eq}}(x,y) - \widetilde{\mathrm{eq}}(x,n-1) \cdot \widetilde{\mathrm{eq}}(y,n-1)$$

which can be evaluated in time $O(v = \log n)$, and the MLE of A_1 is

$$\widetilde{A_1}(x,y) = \widetilde{\mathrm{next}}(x,y)$$

which can also be evaluated in time $O(v)$, as shown in Theorem 2 of [1]. See also below for a description of $\widetilde{\mathrm{next}}$.
Therefore, oracle access to $\widetilde{A_0}$ and $\widetilde{A_1}$ is not required by the verifier.

Cyclic constraints

Some versions of AIR use *cyclic constraints*, i.e., the constraint polynomial also vanishes at $i = n-1$:

$$F(z_0[n-1], \ldots, z_{C-1}[n-1], z_0[0], \ldots, z_{C-1}[0]) = 0.$$

The MCCCS for AIR can be adapted to support this by using the matrices

$$A_0 = I_n = \begin{bmatrix} I_{n-1} & 0 \\ 0 & 1 \end{bmatrix}$$

and

$$A_1 = \begin{bmatrix} 0 & I_{n-1} \\ 1 & 0 \end{bmatrix}.$$

The MLE of these matrices can also be computed in time $O(v)$ using the following identites

$$\widetilde{A_0}(x,y) = \widetilde{\mathrm{eq}}(x,y)$$

and

$$\widetilde{A_1}(x,y) = \widetilde{\mathrm{next}}(x,y) + \widetilde{\mathrm{eq}}(x,n-1) \cdot \widetilde{\mathrm{eq}}(y,0).$$

Handling public inputs

AIR is often used to arithmetize an execution trace. For example, zkVMs often have the structure of an AIR where the first row contains the input of the program and the last row contains the output. Each other row represents a cycle in the zkVM.
For this reason, we might want (parts of) the first and last rows to be public inputs in the proving system. This can be achieved in the proving system for MCCCS as follows.
Say we want the first element of the column z_i to be public. Instead of committing to the MLE of the full column, the prover commits to the MLE of $z'_i = (0\ z_i[1..])$, the same column with the first element set to 0. Then, the verifier can compute the value $\widetilde{Z}_i(r_y)$ using the relation

$$\widetilde{Z}_i(r_y) = \widetilde{Z'}_i(r_y) + z_i[0] \cdot \chi_0(r_y),$$

where χ_0 is the 0-th Lagrange basis multilinear polynomial, which can be evaluated in $O(v)$.
A similar modification can be used to make the last row public.

When the constraint is an arithmetic circuit

In applications, the constraint polynomial $F(X_0, \ldots, X_{2C-1})$ is rarely given in sum-of-monomials form. Most of the time, it is given as an arithmetic circuit. The circuit representation can be much more efficient than the sum-of-monomials representation.
For example,

$$F(X_0, \ldots, X_{2C-1}) = (X_0 + \cdots + X_{2C-1})^2$$

takes $2C - 1$ additions and 1 multiplication to evaluate as an arithmetic circuit, but is a sum of $2C^2 + C$ monomials. So the (MC)CCS representation can be very inefficient, especially when the number of columns in the AIR is large.
However, it is easy to generalize (MC)CCS to work with arithmetic circuits. Let $\mathcal{C}$ be an arithmetic circuit[2] with t inputs $x_0, \ldots, x_{t-1}$ and 1 output $\mathcal{C}(x_0, \ldots, x_{t-1})$. Then, we can modify the CCS constraint equation to be

$$\mathcal{C}(M_0 \cdot z, \ldots, M_{t-1} \cdot z) = \mathbf{0},$$

where the circuit is evaluated on the t vectors entrywise.
As in the SuperSpartan proving system for CCS, it is easy to see that (except with negligible probability) this constraint is equivalent to

$$0 = \sum_{a \in \{0,1\}^{\log m}} \widetilde{eq}(\tau, a) \mathcal{C}\left(u_0, \ldots, u_{t-1}\right), \text{ with } u_i = \sum_{y \in \{0,1\}^{\log n}} \widetilde{M_i}(a, y) \cdot \widetilde{Z}(y),$$

where $\tau \in \mathbb{F}^{\log m}$ is a random vector. This can be proved using an instantiation of the sumcheck protocol for the inner sum, as well as t instantiations of the sumcheck protocol for the values u_i.

[2] Here we consider usual arithmetic circuits over the field $\mathbb{F}$ with binary addition and multiplication gates, as well as unary gates for multiplication by a constant: $\{g_c\}_{c \in \mathbb{F}}$ with $g_c(x) = cx$.

$\widetilde{\textbf{next}}$ and higher order shifts

The function $\widetilde{next}(x, y)$ is described in section 5.1 of [STW23]. It is based on the following algorithm to check that $y = x + 1$ using little-endian bit decomposition.
Let $x_k = 0$ be the first 0 bit of x, so that

$$x = 1 \dots 10x_{k+1} \dots x_{v-1}.$$

Then $y = x + 1$ if and only if $y_i = 0$ for $i < k$, $y_k = 1$, and $x_i = y_i$ for all $i > k$, i.e.,

$$y = 0 \dots 01x_{k+1} \dots x_{v-1}.$$

We can check this with the multilinear polynomial

$$g_k(x, y) = \left(\prod_{i=0}^{k-1} x_i(1 - y_i)\right)(1 - x_k)y_k\ \widetilde{\mathrm{eq}}(x_{k+1}, y_{k+1}) \cdots \widetilde{\mathrm{eq}}(x_{v-1}, y_{v-1}).$$

The big product checks that the first bits of x are 1 and those of y are 0, the term $(1 - x_k)y_k$ checks that $x_k = 0$ and $y_k = 1$, and the $\widetilde{\mathrm{eq}}$ terms check that the remaining bits are equal.
Then we simply have

$$\widetilde{\mathrm{next}}(x, y) = \sum_{k=0}^{v-1} g_k(x, y).$$

Note that if $x = 2^v - 1$ has all 1 bits, then $\widetilde{\mathrm{next}}(x, y) = 0$ for all y since the $1 - x_k$ term is 0 for all k. Finally, note that $g_0(x, y)$ can be computed in time $O(v)$ and that $g_{k+1}(x, y)$ can be computed in time $O(1)$ given $g_k(x, y)$ using

$$g_{k+1}(x, y) = g_k(x, y)\frac{x_k(1 - y_k)(1 - x_{k+1})y_{k+1}}{x_{k-1}(1 - y_{k-1})(1 - x_k)y_k\ \widetilde{\mathrm{eq}}(x_{k+1}, y_{k+1})}.$$

Therefore, $\widetilde{\mathrm{next}}(x, y)$ can be computed in time $O(v)$.

It can sometimes be useful to consider *higher order shifts*, that is constraints of the form

$$F(z_0[i], \dots, z_{C-1}[i], z_0[i + s], \dots, z_{C-1}[i + s]) = 0,$$

where $s > 1$. The above argument can be adapted for this case by using the modified function $\mathrm{next}_s(i, j) = \mathrm{eq}(i + s, j)$. We will only consider the case where $s = 2^e$, but the general case can also be done (I think).
To check $y = x + 2^e$, we observe that it is equivalent to checking that the first e bits of x and y are equal and that the remaining bits satisfy the next relation. That is,

$$\widetilde{\mathrm{next}}_s(x, y) = \widetilde{\mathrm{eq}}(x_0, y_0) \cdots \widetilde{\mathrm{eq}}(x_{e-1}, y_{e-1})\ \widetilde{\mathrm{next}}(x_e, \dots, x_{v-1}, y_e, \dots, y_{v-1}),$$

which can also be computed in time $O(v)$.

References

[1] Srinath Setty, Justin Thaler, and Riad Wahby. Customizable constraint systems for succinct arguments. Cryptology ePrint Archive, Paper 2023/552, 2023. https://eprint.iacr.org/2023/552.