Lecture #8 Task

Distributed Lab

September 12, 2024

0.1 R1CS In Rust

0.1.1 Introduction

This time, the task is a bit unusual: you need to implement a simple Rank-1 Constraint System (R1CS) in Rust. For that reason, consider a pretty simple problem: the prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$ that he knows the modular cube root of y modulo p for the given $y \in \mathbb{F}_p$. Here, p is the BLS12-381 prime, which will become handy in the next tasks.

For that reason, we construct the circuit of the following form:

$$\mathtt{C}(x, y) = x^3 - y,$$

Here, we need only two constraints to check the correctness of the prover's statement:

1. $r_1 = x \times x$.
2. $r_2 = x \times r_1 - y$.

Therefore, the solution vector becomes $\mathbf{w} = (1, x, y, r_1, r_2)$. The goal of this task is to:

- Implement the basic Linear Algebra operations for R1CS in Rust.
- Implement the R1CS satisfiability check.
- Construct the matrices L, R, O to check the satisfiability of the given solution vector $\mathbf{w}$ (checking the cubic root of given y).

0.1.2 Task 1: Preparation

All the source code we are going to refer to is specified by the link below:

`https://github.com/ZKDL-Camp/lecture-8-r1cs-qap`

Download Rust[1] (in case you do not have one), clone/fork the repository and verify that

[1]If you are the total beginner, you might find these official resources useful: `https://www.rust-lang.org/learn`

everything compiles (just that, the code does not work yet). In case you are confused, the project is structured as follows:

- `src/main.rs` contains the entrypoint where you can test your implementation.
- `src/finite_field.rs` contains the $\mathbb{F}_p$ specification — you will not need it.
- `src/linear_algebra.rs` contains the basic Linear Algebra operations (with vectors and matrices) you need to implement.
- `src/r1cs.rs` contains the R1CS implementation where you also would need to implement a piece of functionality.

0.1.3 Task 2: Linear Algebra Operations

Now, recall that our ultimate goal is to construct the matrices L, R, O to check the following satisfiability condition:

$$L\mathbf{w} \odot R\mathbf{w} = O\mathbf{w},$$

And additionally, for education purposes, we will want to check the satisfiability of any specified constraint, that is:

$$\langle \boldsymbol{\ell}_j, \mathbf{w} \rangle \times \langle \boldsymbol{r}_j, \mathbf{w} \rangle = \langle \boldsymbol{o}_j, \mathbf{w} \rangle.$$

For that reason, we need to have the Hadamard product (element-wise multiplication) and inner (dot) product of two vectors and the matrix-vector product. For that reason, implement the following functions in the `linear_algebra.rs` module:

1. `Vector::dot(&self, other: &Self) -> Fp` — the inner product of two vectors.
2. `Vector::hadamard_product(&self, other: &Self) -> Self` — the Hadamard (elementwise) product $\mathbf{v} \odot \mathbf{u}$ of two vectors.
3. `Matrix::hadamard_product(&self, other: &Self) -> Self` — the Hadamard (elementwise) product $A \odot B$ of two matrices.
4. `Matrix::vector_product(&self, other: &Vector) -> Vector` — the matrix-vector product $A\mathbf{v}$.

To test the correctness of your implementation, run

```
cargo test linear_algebra
```

0.1.4 Task 3: R1CS Satisfiability Check

Now, we need to implement the R1CS satisfiability check. For that reason, implement the following functions in the `r1cs.rs` module:

1. `R1CS::is_satisfied(&self, witness: &Vector<WITNESS_SIZE>) -> bool` — the function that checks the satisfiability of the given solution vector $\mathbf{w}$.
2. `R1CS::is_constraint_satisfied(&self, witness: &Vector<WITNESS_SIZE>, j: usize) -> bool` — the function that checks whether the j-th constraint is satisfied.

To test the correctness of your implementation, run

```
cargo test r1cs
```

0.1.5 Task 4: R1CS for Cubic Root

Now, as the final step, construct the matrices L, R, O for the given R1CS problem and check the satisfiability of the solution vector $\mathbf{w} = (1, x, y, r_1, r_2)$ where x is the cubic root of y

modulo p. For that reason, insert the missing pieces of code in the `main.rs` file. This file will automatically:

1. Generate a random valid witness.
2. Construct the R1CS with the given matrices L, R, O.
3. Check the satisfiability of the given solution vector.

Hint. In the lecture, we considered a bit more complicated circuit

$$\mathtt{C}(x_1, x_2, x_3) = x_1 \times x_2 \times x_3 + (1 - x_1) \times (x_2 + x_3), \quad x_1 \in \{0, 1\}, \quad x_2, x_3 \in \mathbb{F}_p$$

You might take a look at how this circuit is implemented in the `r1cs.rs` file in the `tests` module and adapt it to the cubic root problem.