TIP 0001: Contiguity Argument for Memory Consistency

TIP	0001
authors:	Alan Szepieniec and Ferdinand Sauer
title:	Contiguity Argument for Memory Consistency
status:	draft
created:	2022-08-15
issue tracker:	https://github.com/TritonVM/triton-vm/pull/35

Abstract. In the current specification, the memory-like tables `RamTable`, `JumpStackTable`, and `OpStackTable` do not satisfy memory-consistency. Specifically, they are vulnerable to Yuncong's attack, which exploits the unverified and thus possibly-incorrect sorting in these tables. This TIP addresses one part of the issue by introducing a new table argument, the *contiguity argument*. It establishes the contiguity of the regions of fixed memory pointer. This note is a companion to TIP-0002, which introduces an new table to establish the correct sorting of rows by clock cycle within each contiguous region. Together, TIP-0001 and TIP-0002 fix the memory consistency issue.

Introduction

The memory tables `RamTable`, `JumpStackTable`, and `OpStackTable` should be sorted by memory pointer first, and by clock cycle second. When this correct sorting is not enforced, it gives rise to attack undermining memory-consistency.

Part V of the BrainSTARK tutorial shows that memory-consistency follows if, in the memory table, every sublist of rows with the same memory pointer forms a contiguous region. The sorting rule is just one way to guarantee this contiguity. The sorting by clock cycle within each contiguous region is still necessary.

This TIP proposes a subprotocol for establishing the contiguity of all regions with a given memory pointer. This *contiguity argument* is a collection of four extension columns and several extension AIR constraints. The first extension column is a running product similar to that of a conditioned permutation argument. The second extension column is the formal derivative of the first, up to randomization. The next two are Bézout coefficients. An AIR constraint takes the weighted sum and equates it to one, testing the Bézout relation. It can only be satisfied if the greatest common divisor of the running product and its formal derivative is one – implying that no change in the memory pointer resets it to a value used earlier.

Contiguity Argument

The contiguity argument is only needed for the Ram Table because the memory pointer there (`ramp`) can take any value. In contrast, the two other memory-like

tables, OpStackTable and JumpStackTable, are stacks. After sorting by the memory pointer, checking the contiguity of the access pattern is easy: across two consecutive rows, the memory pointer either remains unchanged or increments by one.

Since the contiguity argument may have applications elsewhere, it is presented here in a generic language.

Let `ramp` be the column whose contiguity we wish to establish. We add three extension columns: the difference inverse `di`, the running product `rp`, and the formal derivative `fd`. Additionally, the prover commits to two polynomials $a(X)$ and $b(X)$, but these polynomials do not correspond to columns in the table.

The values contained in these columns is undetermined until the verifier's challenge α is known; before that happens it is worthwhile to present the polynomial expressions in X, anticipating the substitution $X \mapsto \alpha$.

The difference inverse `di` takes the inverse of the difference between the current and next `ramp` values if that difference is non-zero, and zero else. This constraint corresponds to two consistency constraint polynomials: $(\mathsf{ramp}^\star - \mathsf{ramp}) \cdot ((\mathsf{ramp}^\star - \mathsf{ramp}) \cdot \mathsf{di} - 1)$ and $\mathsf{di} \cdot ((\mathsf{ramp}^\star - \mathsf{ramp}) \cdot \mathsf{di} - 1)$.

If zero-knowledge is an objective, let g and h be uniformly random scalars. These scalars are used to randomize columns `rp` and `fd`, respectively. If zero-knowledge is not necessary, set $g = 1$ and $h = 0$. Note that h does not need to be the formal derivative of g. In that case, the name "formal derivative" for column `fd` is not entirely accurate. However, it may help at first reading to imagine these scalars as polynomials satisfying $h(X) = g'(X)$.

The running product `rp` starts with g initially and accumulates a factor $X - \mathsf{ramp}$ in every pair of rows where $\mathsf{ramp} \neq \mathsf{ramp}^\star$. This evolution corresponds to one transition constraint: $(\mathsf{ramp}^\star - \mathsf{ramp}) \cdot (\mathsf{rp}^\star - \mathsf{rp} \cdot (X - \mathsf{ramp})) + (1 - (\mathsf{ramp}^\star - \mathsf{ramp}) \cdot \mathsf{di}) \cdot (\mathsf{rp}^\star - \mathsf{rp})$.

The associated terminal value is rp_T and the terminal constraint stipulates that the factor $X - \mathsf{ramp}$ from the last row is accumulated as well: $\mathsf{rp} \cdot (X - \mathsf{ramp}) - \mathsf{rp}_T$. Alternatively, if the last row is a padding row and should not be accumulated into the running product, then the last value in the column is already the terminal value: $\mathsf{rp} - \mathsf{rp}_T$.

The formal derivative `fd` contains the "formal derivative" of the running product with respect to X (ignoring the masking factors g and h). The formal derivative is initially h. The transition constraint applies the product rule of differentiation conditioned upon the difference in `ramp` being nonzero; in other words, if $\mathsf{ramp} = \mathsf{ramp}^\star$ then the same value persists; but if $\mathsf{ramp} \neq \mathsf{ramp}^\star$ then fd is mapped as

$$\mathsf{fd} \mapsto \mathsf{fd}^\star = (X - \mathsf{ramp}) \cdot \mathsf{fd} + \mathsf{rp}.$$

This update rule is called the *product rule of differentiation* because, when

ignoring $g(X)$ and $h(X)$ or when setting $h(X) = g'(X)$ and assuming $\mathsf{ramp}^\star \neq \mathsf{ramp}$, then

$$\begin{aligned}\frac{\mathsf{drp}^\star}{\mathsf{d}X} &= \frac{\mathsf{d}(X - \mathsf{ramp}) \cdot \mathsf{rp}}{\mathsf{d}X} \\ &= (X - \mathsf{ramp}) \cdot \frac{\mathsf{drp}}{\mathsf{d}X} + \frac{\mathsf{d}(X - \mathsf{ramp})}{\mathsf{d}X} \cdot \mathsf{rp} \\ &= (X - \mathsf{ramp}) \cdot \mathsf{fd} + \mathsf{rp} \ .\end{aligned}$$

The transition constraint for fd is $(\mathsf{ramp}^\star - \mathsf{ramp}) \cdot (\mathsf{fd}^\star - \mathsf{rp} - (X - \mathsf{ramp}) \cdot \mathsf{fd}) + (1 - (\mathsf{ramp}^\star - \mathsf{ramp}) \cdot \mathsf{di}) \cdot (\mathsf{fd}^\star - \mathsf{fd})$.

The terminal value associated with fd is fd_T. The terminal constraint stipulates one last application of the product rule of differentiation: $(X - \mathsf{ramp}) \cdot \mathsf{fd} + \mathsf{rp} - \mathsf{fd}_T$. Alternatively, if the last row is padding and should not contribute a factor to rp, then the terminal constraint for fd is simply $\mathsf{fd} - \mathsf{fd}_T$ because the terminal value is already present in the last row of this column.

Until the verifier supplies the challenge α, the terminal "values" are in fact polynomials: $\mathsf{rp}_T(X)$ and $\mathsf{fd}_T(X)$. The polynomials $a(X)$ and $b(X)$ are *randomized* Bézout coefficients of the relation

$$a(X) \cdot \mathsf{rp}_T(X) + b(X) \cdot \mathsf{fd}_T(X) = \gcd(\mathsf{rp}_T(X), \mathsf{fd}_T(X)) \ .$$

If zero-knowledge is an objective, then $a(X) = a^*(X) + k \cdot \mathsf{fd}_T(X)$ and $b(X) = b^*(X) - k \cdot \mathsf{rp}_T(X)$, where $a^*(X)$ and $b^*(X)$ are the minimal-degree Bézout coefficients as returned by the extended Euclidean algorithm, and where $k \in \mathbb{F}$ is a uniformly random field element called the *Bézout randomizer*. For the sake of the soundness analysis, set the degree bound on $a(X)$ and $b(X)$ to T, the height of the table.

When the verifier supplies α, the prover responds with (among other things) the terminal values $\mathsf{rp}_T = \mathsf{rp}_T(\alpha)$ and $\mathsf{fd}_T = \mathsf{fd}_T(\alpha)$. The prover also supplies the Bézout coefficients $A = a(\alpha)$ and $B = b(\alpha)$. The verifier verifies the correct calculation of the terminals using the AIR, and the correct calculation of the Bézout coefficients using the DEEP technique: the prover adds $q_a(X) = \frac{a(X) - A}{X - \alpha}$ and $q_b(X) = \frac{b(X) - B}{X - \alpha}$ to the nonlinear combination, and the verifier verifies that it was added. The verifier additionally verifies that $A \cdot \mathsf{rp}_T + B \cdot \mathsf{fd}_T = 1$.

Completeness. The prover is incapable of proving that the gcd equals 1 when $\mathsf{rp}_T(X)$ and $\mathsf{fd}_T(X)$ share a factor. Since g and h are scalars they do not change the gcd. Therefore, the gcd can only be non-one for dishonest provers. As a result, the protocol has perfect completeness. □

Soundness. If the table has at least one non-contiguous region, then $\mathsf{rp}_T(X)$ and $\mathsf{fd}_T(X)$ share at least one factor. As a result, no Bézout coefficients $a(X)$ and $b(X)$ can exist such that $a(X) \cdot \mathsf{rp}_T(X) + b(X) \cdot \mathsf{fd}_T(X) = 1$. The verifier

therefore probes unequal polynomials of degree at most $2T$. According to the Schwartz-Zippel lemma, the false positive probability is at most $2T/|\mathbb{F}|$. □

Zero-Knowledge. The prover sends four polynomial evaluations, two of which are terminals. We treat each value in turn. 1. $\mathsf{rp}_T(\alpha)$: For any $\mathsf{rp}_T(\alpha)$, for any list $\{\mathsf{ramp}_i\}_i$ of accumulated ramp values, and for any challenge α, there is an initial g that is consistent with the given view. Indeed, the mapping from $g \mapsto \mathsf{rp}_T(\alpha)$ is affine and the coefficient is nonzero as long as $\alpha \notin \{\mathsf{ramp}_i\}_i$. 2. $\mathsf{fd}_T(\alpha)$: For any pair $(\mathsf{rp}_T(\alpha), \mathsf{fd}_T(\alpha))$, for any list $\{\mathsf{ramp}_i\}_i$ of accumulated ramp values, and for any challenge α, there is an initial h that is consistent with the given view. Indeed, the mapping from $h \mapsto \mathsf{fd}_T(\alpha)$ is affine and the coefficient is nonzero as long as $\alpha \notin \{\mathsf{ramp}_i\}_i$. 3. $a(\alpha)$: For any triple $(\mathsf{rp}_T(\alpha), \mathsf{fd}_T(\alpha), a(\alpha))$, for any list $\{\mathsf{ramp}_i\}_i$ of accumulated ramp values, and for any challenge α, there is a Bézout randomizer k that is consistent with the given view. Indeed, the mapping from $k \mapsto a(\alpha)$ is affine and the coefficient is nonzero as long as $\alpha \notin \{\mathsf{ramp}_i\}_i$. 4. $b(\alpha)$: The remaining Bézout evaluation $b(\alpha)$ is constrained by $a(\alpha) \cdot \mathsf{rp}_T(\alpha) + b(\alpha) \cdot \mathsf{fd}_T(\alpha) = 1$, and therefore does not leak any new information about the witness. □