Pre-Coding Essentials (Component: crates/vm\_algo/src/allocation/wta.rs, Version/FormulaID: VM-ENGINE v0) — 42/89

1. Goal & Success

Goal: Winner-take-all allocation per Unit: pick the option with the highest Unit score and allocate 100% power to it. Enforce the rule **WTA ⇒ Unit.magnitude = 1**.

Success: Deterministic winner for any input; m≠1 rejected; ties resolved per **VM-VAR-032 tie\_policy** (status\_quo / deterministic / random with **VM-VAR-033 tie\_seed**).

1. Scope

In scope: Max-by-score selection, WTA coherence checks, tie breaking, return of Allocation { unit\_id, seats\_or\_power } with 100% for the winner.

Out of scope: Tabulation, gates/labels, aggregation, schema/I/O.

1. Inputs → Outputs

Inputs:

* scores: &UnitScores (from TABULATE; integers only)
* magnitude: u32 (**must be 1**)
* options: &[OptionItem] (canonical order, includes order\_index and is\_status\_quo)
* tie\_policy: TiePolicy and optional rng: &mut TieRng if random

Output:  
Allocation { unit\_id, seats\_or\_power: { winner → 100 }, last\_seat\_tie: bool } (WTA uses power=100% as in tests).

1. Entities/Tables (minimal)

None.

1. Variables (used here)

* **VM-VAR-032 tie\_policy** ∈ {status\_quo, deterministic, random} (default: status\_quo)
* **VM-VAR-033 tie\_seed** ∈ integers (≥ 0) (default: 0) — used only when tie\_policy = random

1. Functions (signatures only)

use std::collections::BTreeMap;

use vm\_core::{

ids::{UnitId, OptionId},

entities::OptionItem,

rng::TieRng,

variables::TiePolicy,

};

use crate::tabulation::UnitScores;

pub fn allocate\_wta(

scores: &UnitScores,

magnitude: u32,

options: &[OptionItem],

tie\_policy: TiePolicy,

rng: Option<&mut TieRng>,

) -> Result<Allocation, AllocError>;

// helpers

fn top\_by\_score(scores: &UnitScores) -> (u64, Vec<OptionId>); // max score and all tied at max

fn break\_tie\_wta(

tied: &[OptionId],

options: &[OptionItem],

tie\_policy: TiePolicy,

rng: Option<&mut TieRng>,

) -> OptionId;

1. Algorithm Outline (implementation plan)

**Preconditions**

* Require magnitude == 1; else AllocError::InvalidMagnitude. (Also validated earlier in VALIDATE.)

**Find maximum**

* Scan scores.scores (integers) to get max and the list of tied options at max. Integer math only.

**Tie handling**

* If tied.len() == 1 → winner = tied[0].
* Else apply **tie\_policy (VM-VAR-032)**:
  + status\_quo → pick the option with is\_status\_quo = true; if none or multiple, fall back to deterministic.
  + deterministic → pick the smallest **(order\_index, OptionId)** among tied (uses Option.order\_index).
  + random → draw uniformly using **ChaCha20** seeded RNG constructed from **tie\_seed (VM-VAR-033)**; log via pipeline TieLog rules.

**Assemble allocation**

* seats\_or\_power = { winner: 100 }, last\_seat\_tie = (tied.len() > 1).

1. State Flow

Pipeline order: TABULATE → **ALLOCATE (this WTA)** → AGGREGATE.  
Ties (if any) are recorded per pipeline tie-logging rules; RNG only used when tie\_policy = random.

1. Determinism & Numeric Rules

Stable option ordering by (order\_index, id); exact integers; no floats.  
Random tie breaks use only **tie\_seed (VM-VAR-033)**; same inputs + same seed ⇒ identical outcome/logs across OS/arch.

1. Edge Cases & Failure Policy

* magnitude != 1 ⇒ AllocError::InvalidMagnitude.
* All scores zero ⇒ still select per tie policy (status\_quo → SQ; else deterministic/random).
* Unknown options cannot appear (UnitScores comes from validated loader); if encountered, panic in debug / AllocError::UnknownOption in release.
* Multiple is\_status\_quo = true is invalid upstream; if encountered, fall back to deterministic.

1. Test Checklist (must pass)

* **VM-TST-002 (WTA wipe-out):** plurality A/B/C/D = 10/20/30/40, m=1 ⇒ D gets 100%.
* **Magnitude guard:** set m=2 under WTA ⇒ fail with InvalidMagnitude.
* **Status-quo tie:** top tie between Change and Status Quo with equal scores ⇒ winner = Status Quo under status\_quo.
* **Deterministic tie:** same tie with deterministic ⇒ pick lowest (order\_index, id).
* **Random tie (seeded):** same tie with random and fixed **tie\_seed** ⇒ reproducible winner and TieLog.