Superposition Futarchy: Conditional Execution via Market-Based State Collapse

Greshams Code[1]

[1]Founder, Govex.ai

June 2025

Abstract

We present a novel form of futarchy where conditional outcomes exist in superposition until resolved by market consensus. Upon initiation of a futarchy proposal all proposed token actions are minted and performed immediately with conditional tokens. These tokens trade freely until being resolved by highest reading the Time-Weighted-Average-Price. The protocol operates on a state budget rather than limiting concurrent events, allowing flexible combinations (e.g., the Cartesian product of 4 binary events or one 16-outcome event within a 16-state budget).

1 Immediate Conditional Token Creation

- Current implementations of futarchy allow users to create proposals for a company treasury to transfer spot tokens to an address when if the proposal passes [1]. Assuming the proposal measuring period is X seconds. This creates X seconds latency between decision proposal and decision actions. This has an opportunity cost of :

$$C = V \cdot X \cdot r \tag{1}$$

where:

$$\begin{aligned} C &= \text{Opportunity Cost} \\ V &= \text{Value to Transfer} \\ X &= \text{Latency (in seconds)} \\ r &= \text{Market Interest Rate} \end{aligned}$$

Immediate transfers offer immediate capital utility.

- In Superposition futarchy if Alice creates a proposal for company B to pay her 1000 USDC to do work. She immediately get sent 1000 Accept-USDC. This will only be redeemable for 1000 spot USDC if the proposal passes. So she both does and doesn't get paid. The decision markets collapse the superposition to one of the options.

- This is a useful abstraction that allows a futarchy treasury to atomically buy back or dilute its own stock when it is trades below or above net asset value, without being front run. MntCapital an onchain fund had significant friction with buy backs [2]. This requires deep conditional liquidity, which a futarchy AMM provides [3].

- Conditional tokens trade freely until resolution. This helps the market to more fairly price and token actions.

2 Superposition State Budget

- Current leading implementations of futarchy allow for N outcomes markets that share the same liquidity[4]. Other proposals do not share the same liquidity. Assuming all liquidity is in a Uniswap-V2 style AMM, thicker liquidity will create greater incentives for traders to price the outcomes accurately.

- Each proposal i defines a **factor space** $\mathcal{F}_i$ with k_i possible outcomes. For binary proposals, $\mathcal{F}_i = \{\text{Accept}_i, \text{Reject}_i\}$ where $k_i = 2$.

  Consider two binary proposals: Alice requesting 1000 USDC and Bob requesting 1000 USDC. The complete state space is:

  $$\mathcal{S} = \mathcal{F}_A \times \mathcal{F}_B \tag{2}$$

  This yields four states:

	Accept_B	Reject_B
Accept_A	Alice: 1000 USDC, Bob: 1000 USDC	Alice: 1000 USDC, Bob: 0 USDC
Reject_A	Alice: 0 USDC, Bob: 1000 USDC	Alice: 0 USDC, Bob: 0 USDC

- For n proposals, the state space size is:

  $$|\mathcal{S}| = \prod_{i=1}^{n} k_i \tag{3}$$

  subject to the state budget constraint:

  $$|\mathcal{S}| \leq B \tag{4}$$

  where B is the maximum state budget. This allows flexible combinations (e.g., 4 binary proposals yield $2^4 = 16$ states, equivalent to one 16-outcome proposal).

- The state space forms a tensor $\mathcal{T} \in \mathbb{R}^{k_1 \times k_2 \times \ldots \times k_n}$ where each proposal corresponds to a mode. Each element $\mathcal{T}_{i_1, i_2, \ldots, i_n}$ represents the state where proposal 1 has outcome i_1, proposal 2 has outcome i_2, etc.

  When proposal j resolves to outcome o_j^*, the state space collapses along factor $\mathcal{F}_j$:

  $$\mathcal{S}_{\text{collapsed}} = \mathcal{F}_1 \times \ldots \times \{o_j^*\} \times \ldots \times \mathcal{F}_n \tag{5}$$

  This is equivalent to taking a tensor slice along mode j at index o_j^*.

- Proposals can contribute factors of varying sizes:
  - Binary vote: $|\mathcal{F}_i| = 2$
  - Multi-option ranking: $|\mathcal{F}_i| = 5$ (for 5 options)
  - Complex conditional proposal: $|\mathcal{F}_i| = k_i$ conditional paths

- New proposals enter freed slots when proposal j resolves, the state budget capacity freed is:

  $$\Delta B = |\mathcal{S}| \cdot \left(1 - \frac{1}{k_j}\right) \tag{6}$$

  New proposal m with k_m outcomes can enter if:

  $$|\mathcal{S}_{\text{collapsed}}| \cdot k_m \leq B \tag{7}$$

- State expansion and continuous decision pipeline: when proposal m enters at time t, the state space expands:

  $$\mathcal{S}_{t+1} = \mathcal{S}_t \times \mathcal{F}_m \quad \text{where} \quad |\mathcal{S}_{t+1}| = |\mathcal{S}_t| \cdot k_m \tag{8}$$

  This creates a continuous pipeline where proposals can be added as capacity becomes available, maintaining:

  $$\prod_{i \in \text{active}} k_i \leq B \tag{9}$$

- State collapse mechanics: when proposal j resolves to o_j^*, the collapse operation:

$$\Pi_j(o_j^*) : \mathcal{S} \to \mathcal{S}_{\text{collapsed}} \tag{10}$$

reduces the state count by factor k_j:

$$|\mathcal{S}_{\text{collapsed}}| = \frac{|\mathcal{S}|}{k_j} \tag{11}$$

- When spot tokens enter the system, they split 1:1 into each active state. For example, 1000 spot USDC with 16 active states becomes 1000 conditional USDC in each state (16,000 total conditional tokens). This ensures full liquidity depth in every state - there is no fragmentation or liquidity sharding.

- Total AMM liquidity L is preserved during state transitions. When all proposals resolve, conditional liquidity converts to spot liquidity at a 1:1 ratio for winning outcomes.

- Lazy token state updates: When users interact with their conditional tokens, the system must resolve their current valid states. Two approaches:

  - **State History Traversal**: Maintain a log of state collapses. When user tokens are accessed, replay collapses from their last update timestamp. Complexity: O(k) where k is number of resolutions since last update.
  - **Finalization Checkpoints**: After sufficient proposals resolve, mark remaining states as "finalized" with direct spot redemption rates. Tokens in finalized states skip traversal entirely (O(1) redemption).

  Example: User holds tokens from state (Accept_A, Accept_B, Reject_C). If proposal A resolved to Reject, their tokens are worthless. If A and B both resolved to Accept, and only C remains active, their tokens map to states (Accept_C) and (Reject_C) with defined redemption values.

- TWAP tracking via maximum state price: For each proposal i, the system tracks the highest-priced state for each outcome at every time window. Every window (e.g., 1 minute):

$$\text{TWAP}_{o,t} = \max_{\{s \in \mathcal{S} : s_i = o\}} \text{TWAP}(s) \tag{12}$$

  At resolution time T, the winning outcome is:

$$o_i^* = \arg\max_{o \in \mathcal{F}_i} \frac{1}{T} \sum_{t=0}^{T} \text{TWAP}_{o,t} \tag{13}$$

  Example: Proposal A tracked over 3 windows:

  - Window 1: Accept_A max state TWAP: \$0.70, Reject_A max state TWAP: \$0.35
  - Window 2: Accept_A max state TWAP: \$0.75, Reject_A max state TWAP: \$0.30
  - Window 3: Accept_A max state TWAP: \$0.68, Reject_A max state TWAP: \$0.32

  Accept_A wins with average of max TWAPs: \$0.71 vs Reject_A's \$0.323.

3 References

1. https://www.govex.ai/
2. https://metadao.fi/mtncapital/trade-v4/CV5gPgHMyJQV3a9m5FZnAxvRXsAn65dMScsANTCydHrX
3. https://x.com/metaproph3t/status/1930686351680409637
4. https://www.govex.ai/trade/0x840e677cda2f2078ca7042d2cda9b586e13089e5e4956a63f5ee386f2fa98cb6