Token Valuation and Redemption System: Algorithmic Specification

Specification Document

February 22, 2025

1 System Specification: Formal Definition

1.1 System Parameters

$$N = \text{Total number of tokens in the system} = 1,000,000 \tag{1}$$
$$N_{\text{active}} = \text{Number of active tokens} = 100,000 \text{ initially} \tag{2}$$
$$P_{\text{initial}} = \text{Initial token price} = \$1.00 \tag{3}$$
$$P_{\text{sale}} = \text{Token sale price} = \$1.35 \tag{4}$$
$$P_{\text{core}} = \text{Core token price (without premium)} = \$1.00 \tag{5}$$
$$P_{\text{premium}} = \text{Premium component of sale price} = \$0.35 \tag{6}$$
$$F_{\text{initial}} = \text{Initial operational fund value} = P_{\text{core}} \times 900,000 = \$1.00 \times 900,000 = \$900,000 \tag{7}$$
$$F_{\text{reserve}} = \text{Premium reserve (not utilized for operations)} = P_{\text{premium}} \times 900,000 = \$0.35 \times 900,00 \tag{8}$$

1.2 Token State Variables

For each token $i \in \{1, 2, \ldots, N\}$:

$$s_i = \begin{cases} 1 & \text{if token is active} \\ 0 & \text{if token is inactive (redeemed)} \end{cases} \tag{9}$$
$$t_i = \text{Token activation timestamp} \tag{10}$$
$$v_i = \text{Current token valuation} \tag{11}$$

1.3 Fund State Variables

$$F(t) = \text{Total fund value at time } t \tag{12}$$

$$R(t_1, t_2) = \text{Return rate of the fund between times } t_1 \text{ and } t_2 \tag{13}$$

1.4 Token Valuation Formula

For an active token i at time t, its value is:

$$v_i(t) = P_{\text{initial}} \times \prod_{j=t_i}^{t} (1 + R(j-1, j)) \tag{14}$$

1.5 Token Redemption Process

When token i is redeemed at time t_{redeem}:

$$\text{Redemption Value} = v_i(t_{\text{redeem}}) \tag{15}$$

$$F(t^+_{\text{redeem}}) = F(t^-_{\text{redeem}}) - v_i(t_{\text{redeem}}) \tag{16}$$

$$s_i = 0 \tag{17}$$

1.6 Token Reissuance Process

When token i is reissued at time t_{reissue}:

$$s_i = 1 \tag{18}$$

$$t_i = t_{\text{reissue}} \tag{19}$$

$$v_i(t_{\text{reissue}}) = P_{\text{initial}} \tag{20}$$

1.7 Percentage-Based Earnings Formula

For a token i active during time period $[t_1, t_2]$:

$$\text{Earnings}_i(t_1, t_2) = v_i(t_1) \times R(t_1, t_2) \tag{21}$$

2 Algorithm Implementation

Algorithm 1 Token Valuation and Redemption System

procedure InitializeSystem($N, N_{\text{active}}, P_{\text{initial}}, P_{\text{sale}}$)
    ▷ Initialize system with input parameters and validate
    if $N < N_{\text{active}}$ **or** $P_{\text{sale}} \leq P_{\text{initial}}$ **then**
        return ERROR_INVALID_PARAMETERS
    end if
    $tokens \leftarrow$ new Array$[N]$ ▷ Allocate token storage
    $operational\_fund \leftarrow P_{\text{core}} \times (N - N_{\text{active}})$ ▷ Core capital for fund operations
    $premium\_reserve \leftarrow P_{\text{premium}} \times (N - N_{\text{active}})$ ▷ Premium reserve (untouched)
    $fund\_value \leftarrow operational\_fund$ ▷ Only core capital is actively managed
    for $i \leftarrow 1$ **to** N **do**
        if $i \leq N_{\text{active}}$ **then**
            $tokens[i].status \leftarrow$ ACTIVE
            $tokens[i].activation\_time \leftarrow$ current_time()
            $tokens[i].value \leftarrow P_{\text{initial}}$
        else
            $tokens[i].status \leftarrow$ INACTIVE
            $tokens[i].activation\_time \leftarrow 0$
            $tokens[i].value \leftarrow 0$
        end if
    end for
    $system\_state \leftarrow \{\text{tokens}, fund\_value, P_{\text{initial}}, P_{\text{sale}}, N, N_{\text{active}}\}$
    return $system\_state$
end procedure
procedure UpdateFundValue($system\_state, return\_rate, period\_start, period\_end$)
    ▷ Update operational fund value based on return rate for specified period
    if $return\_rate < -1.0$ **then**
        return ERROR_INVALID_RETURN_RATE ▷ Prevent invalid negative returns exceeding -100%
    end if
    $operational\_fund \leftarrow system\_state.operational\_fund$
    $new\_operational\_fund \leftarrow operational\_fund \times (1 + return\_rate)$
    $system\_state.fund\_return\_history[period\_start, period\_end] \leftarrow return\_rate$
    $system\_state.operational\_fund \leftarrow new\_operational\_fund$
    ▷ Premium reserve remains untouched - enforcing immutability constraint
    if $system\_state.premium\_reserve \neq P_{\text{premium}} \times (N - N_{\text{active}})$ **then**
        LOG_WARNING("Premium reserve integrity violation detected")
        $system\_state.premium\_reserve \leftarrow P_{\text{premium}} \times (N - N_{\text{active}})$ ▷ Restore reserve integrity
    end if
    return $system\_state$

3 Examples

3.1 Initial Setup Example

- Total tokens: $N = 1,000,000$
- Active tokens: $N_{\text{active}} = 100,000$
- Initial token price: $P_{\text{initial}} = \$1.00$
- Token sale price: $P_{\text{sale}} = \$1.35$
- Core token price (without premium): $P_{\text{core}} = \$1.00$
- Premium component: $P_{\text{premium}} = \$0.35$
- Initial operational fund: $F_{\text{initial}} = \$1.00 \times 900,000 = \$900,000$
- Premium reserve (untouched): $F_{\text{reserve}} = \$0.35 \times 900,000 = \$315,000$

3.2 Token Valuation Example

- Assume a token was activated at $t = 0$
- Fund returns: $R(0,1) = 10\%$, $R(1,2) = 5\%$, $R(2,3) = 8\%$
- At $t = 3$, the token value is:

$$\begin{aligned} v(3) &= \$1.00 \times (1 + 0.10) \times (1 + 0.05) \times (1 + 0.08) && (22) \\ &= \$1.00 \times 1.10 \times 1.05 \times 1.08 && (23) \\ &= \$1.00 \times 1.2474 && (24) \\ &= \$1.2474 && (25) \end{aligned}$$

3.3 Token Redemption and Reissuance Example

- Token value at redemption: $v_i(t_{\text{redeem}}) = \$1.2474$
- Fund value before redemption: $F(t^-_{\text{redeem}}) = \$350,000$
- Fund value after redemption: $F(t^+_{\text{redeem}}) = \$350,000 - \$1.2474 = \$348,998.7526$
- After reissuance at time t_{reissue}, $v_i(t_{\text{reissue}}) = \$1.00$
- If the fund return for the next month is $R(t_{\text{reissue}}, t_{\text{reissue}} + 1) = 10\%$, then:

$$\text{Earnings}_i(t_{\text{reissue}}, t_{\text{reissue}} + 1) = \$1.00 \times 0.10 = \$0.10 \tag{26}$$

4 Implementation Considerations

4.1 System Atomicity

The system must ensure that all operations, particularly those modifying the fund value (redemptions) and token states, are performed atomically to maintain data consistency. This is critical in a multi-user environment where concurrent operations may occur.

4.2 Time Granularity

The system's performance tracking requires a well-defined time granularity. Typically, fund returns are calculated daily, monthly, or quarterly. The algorithm accommodates any granularity, but implementation should enforce consistent time periods.

4.3 Numerical Precision

Financial calculations require high precision. All monetary values should be stored using fixed-point arithmetic or decimal types rather than floating-point to avoid rounding errors that could accumulate over time.

4.4 Error Handling

The algorithm includes robust error handling for invalid inputs, token operations, and insufficient fund values. Implementation should extend this with comprehensive validation and logging.

4.5 Optimization Opportunities

The calculation of token values can be optimized by maintaining cumulative return indices. Instead of recalculating from the activation time for each valuation, the system can store precomputed factor values at regular intervals.