Splash StablePool protocol: Formal Spec

June 7, 2024

1 Model

Let n denote a number of distinct tradable assets $n, n \in N \setminus \{0, 1\}$.
Let NFT denote a unique non-fungible identifier of the pool.
Let $\boldsymbol{a} = \{a_k\}_{k=1}^{n}$ denote a set of distinct tradable assets (including native currency token).
Let $P_{cp} = P^{a_1 \otimes a_2 \otimes \cdots \otimes a_n}$ denote a unique pool comprised of assets $\boldsymbol{a}$.
Let P_i denote an i'th state of a unique liquidity pool P.
Let l denote an asset representing the liquidity provider's share of the P.
Let $d^{a_1 \oplus a_2 \oplus \cdots \oplus a_n}$ denote an integer multiplier for assets. Multipliers are calculated as the ratio of the maximum value of decimals from set $\boldsymbol{a}$ to the decimals of the stablecoin $a_k \in \boldsymbol{a}$.
Let $M^{a_1 \oplus a_2 \oplus \cdots \oplus a_n}$ denote an integer amount of any asset from $\boldsymbol{a}$.
Let $T^{a_1 \oplus a_2 \oplus \cdots \oplus a_n}$ denote an integer amount of treasury any asset from $\boldsymbol{a}$.
Let Y^l denote supply of the liquidity token l.
Let $C = P^{num} \otimes L^{num} \otimes F^{den}$ denote pool fee configuration, where P^{num} - integer value of the protocol fee numerator, L^{num} - integer value of the liquidity provider fee numerator, F^{den} - integer value of pool fees denominator.
Let Q denote staking credential of a pool.
Let A denote an integer value of the amplification coefficient in the StableSwap invariant multiplied by n^{2n}.
Let Z denote script credential of treasury script address.
Let D denote StablePool-proxy DAO script witness.
Let V denote Splash DAO voting script witness.
Let $LP_{edit} = 0 \oplus 1$ denote ability to change P^{num} in DAO-actions.
Let $A_{edit} = 0 \oplus 1$ denote ability to change A in DAO-actions.

Using the notations above a unique StablePool can be defined as:

$$\boxed{P = \quad M_{cp} \otimes T_{cp} \otimes d_{cp} \otimes Y^l \otimes C \otimes A \otimes Z \otimes D \otimes LP_{edit} \otimes A_{edit}}$$

where $M_{cp} = M^{a_1} \otimes M^{a_2} \otimes \cdots \otimes M^{a_n}$, $T_{cp} = T^{a_1} \otimes T^{a_2} \otimes \cdots \otimes T^{a_n}$, $d_{cp} = d^{a_1} \otimes d^{a_2} \otimes \cdots \otimes d^{a_n}$.

Lets now define sum of possible state transitions allowed to be applied to P. We will highlight 2 main categories:

- AMM-actions $A = D \otimes R \otimes S$, where:
  - $D = M_{cp} \otimes Y^l$ - deposit liquidity action;
  - $R = Y^l \otimes M_{cp}$ - redeem liquidity action;
  - $S = M^{a_i \oplus a_j} \otimes M^{a_{i_1} \oplus a_{j_1}} \otimes \cdots \otimes M^{a_{i_m} \oplus a_{j_m}}, i_k \neq j_k \wedge m = C_n^2$ - swap action a_{i_k} to a_{j_k} and vise versa.

- DAO-actions $U \equiv \{U_{ulf}, U_{upf}, U_{uta}, U_{usc}, U_{uac}, U_{wt}\}$, where:
  - Update liquidity provider fee $U_{ulf} = t \otimes \sigma_D \otimes \sigma_V$, where t is a new parameter integer value and $\sigma_D = \mathbb{1}$, $\sigma_V = \mathbb{1}$ stand for validation results of D and V DAO-scripts respectively;
  - Update protocol fee $U_{upf} = t \otimes \sigma_D \otimes \sigma_V$;
  - Update treasury address $U_{uta} = z \otimes \sigma_D \otimes \sigma_V$ to new address z;
  - Update staking credential of the pool $U_{usc} = q \otimes \sigma_D \otimes \sigma_V$ to new credential q;
  - Update amplification coefficient of the pool $U_{uac} = a \otimes \sigma_D \otimes \sigma_V$ to new value a;
  - Withdrawal treasury and set a new value vector of treasury balances $\vec{t_u}$: $U_{wt} = \vec{t_u} \otimes \sigma_D \otimes \sigma_V$.

As a result, a complete set of state transitions that can be applied to the pool P is $\Omega = A \otimes U \otimes K$ and the state transition function can be defined as follows:

$$\boxed{apply: \quad P_i \to \Omega \to P_{i'} \oplus \mathbb{1}}$$

2 Security Assumptions

General preliminary assumptions are:

- **Pool creation**. The correctness of the creation of the pool is validated only off-chain, since adding on-chain validations will permanently affect the pool contract by increasing execution units (exUnits) and memory usage (exMemory), which is undesirable. All users can can independently verify the pool's creation accuracy due to the open-source nature of contracts and off-chain code.

- **Minting policy**. Minting policies validators for NFT and l assets are already implemented and are out of the scope of the this protocol.

- **Orders**. There is no restriction on the types of orders the pool can execute, this is protocol feature.

2.1 Notations

It's useful to introduce some commonly used notations. First of all, we will denote all sets of length n in form of vectors, i.e. $P_i.\vec{d}$ is a vector of assets and etc. Also, we will use Δ notations to represent difference between integer values of the same type, i.e. $\Delta Y^l = P_{i'}.Y^l - P_i.Y^l$ and etc.

- $\vec{M_{i'}^{tr}} = P_{i'}.\vec{M} - P_{i'}.\vec{T}$ - tradable balances (without protocol fees since we accumulate is inside the pool);

- $\vec{M_{i'}^{trnl}} = \vec{M_{i'}^{tr}} - \vec{L_{i'}^{fee}}$ - tradable balances without latest protocol fees and liquidity provider fees $\vec{L_{i'}^{fee}}$, which aren't stored anywhere and are calculated on the fly;

- $check\_invariant(A, n, W, \vec{d}, \vec{M_i^{tr}}, \vec{M_{i'}^{trnl}}) \to 0|1$ - function to check the validity of the StableSwap invariant for the pool state transition.

- $check\_invariant\_exact(A, n, W, \vec{d}, \vec{M_i^{tr}}, \vec{M_{i'}^{trnl}}) \to 0|1$ - function to check the validity of the StableSwap invariant for the pool state transition with exact calculations.

There is also a pre-defined value E representing total emission of Y^l tokens that is the same for pools with any number of tradable assets n.

Let us also introduce so-called "config set of parameters":

$$P_i.conf = \{P_i.n,\ P_i.NFT,\ P_i.d_{cp},\ P_i.\boldsymbol{a},\ P_i.l,\ P_i.LP_{edit},\ P_i.A_{edit},\ P_i.D\}.$$

And "set of mutable parameters":

$$P_i.mut = \{P_i.L^{num},\ P_i.P^{num},\ P_i.A,\ P_i.Z,\ P_i.Q\}.$$

We will also often refer to 2 types of validations:

- **Relaxed** - validator ensures that the reserves are safu;
- **Exact** - validator ensures that the calculations with reserves are accurate;

2.2 Invariant validation

To validate off-chain calculated values of the invariant we are using special function than checks the pool state transition validity in the relaxed case $check\_invariant(A, n, W, \vec{d}, \vec{M}_i^{tr}, \vec{M}_{i'}^{trnl})$. Since the values of the invariant I cannot be calculated in integers, after rounding the invariant, exact equality in the StableSwap equality

$$A/n \sum \vec{M} + I = IA/n + I^{n-1}/n^n / \prod \vec{M}$$

is not achieved. Let's introduce a solution witness W. In fact, it is a number provided by client belonging to the interval $[I, I']$ where I is the old invariant value, I' is the new one. The idea is that given the witness, it is easy to check that the interval $[I, I']$ is not empty.

Let α, β be the polynomial coefficients for I. Using notations from the pool config $\alpha = (A - n^n) \prod \vec{M}_i^{tr} \vec{d}$ and $\beta = A \prod \vec{M} \vec{d} \sum \vec{M}_i^{tr} \vec{d}$. If α, β values are corresponding to the old state and α', β' are corresponding to the new state $\vec{M}_i^{trnl}$, then validations are:

- $W^{n+1} + \alpha W - \beta \geq 0$,
- $W^{n+1} + \alpha' W - \beta' \leq 0$.

There is also a modified version $check\_invariant\_exact(A, n, W, \vec{d}, \vec{M}_i^{tr}, \vec{M}_{i'}^{trnl})$ for the exact case. This function ensures that value W is close to the I' as much as it possible. Let us fix some error e as maximum allowed error for swap output in minimal units of the corresponding asset. Now we will reduce the quote asset j tradable balance in the $\vec{M}_{i'}^{trnl}$ by this error and calculate the shifted values $\alpha^{shifted}, \beta^{shifted}$. Final system of validations is:

- $W^{n+1} + \alpha W - \beta \geq 0$,
- $W^{n+1} + \alpha' W - \beta' \leq 0$,
- $W^{n+1} + \alpha^{shifted} W - \beta^{shifted} \geq 0$.

If this is true, then we know that for final balances the invariant does not just lie inside the interval, but lies as close as possible to the right end. Thus, possible output from the pool is as close as possible to the real mathematics of a stable swap and other state transitions (even increasing pool reserves for free) are simply impossible.

2.3 Deposit and Redeem

In deposit/redeem actions in the relaxed case we assume that *apply* : $P_i \rightarrow D \oplus R \rightarrow P_{i'} \oplus \mathbb{1}$ satisfies the following conditions:

$$\boxed{\begin{array}{rl} P_{i'}.conf = & P_i.conf \\ P_{i'}.mut = & P_i.mut \\ P_{i'}.\vec{T} \geq & P_i.\vec{T} \\ \Delta Y^l \vec{M}_i^{tr} \leq & P_i.Y^l \Delta \vec{M}^{tr} \end{array}}$$

In deposit/redeem actions in the exact case we assume that *apply* : $P_i \rightarrow D \oplus R \rightarrow P_{i'} \oplus \mathbb{1}$ satisfies the following conditions:

$$\boxed{\begin{array}{rl} P_{i'}.conf = & P_i.conf \\ P_{i'}.mut = & P_i.mut \\ P_{i'}.\vec{T} = & P_i.\vec{T} \\ \Delta Y^l \vec{M}_i^{tr} = & P_i.Y^l \Delta \vec{M}^{tr} \end{array}}$$

2.4 Swap

To validate the correctness of the applied deposit/redeem state transition we must firstly perform the following steps:

1. Extract $W \leftarrow S$;
2. Define quote and base asset indexes b and q respectively;
3. Calculate $f\_num\_rev = (P_i.F^{den} - P_i.L^{num} - P_i.P^{num})$;
4. Calculate quote balance without latest liquidity provider fees as $M_{trnl}{}_{i'}^{q} = M_{tr}{}_{i'}^{q} - \Delta M_{tr}^{q} P_i.L^{num}/(P_i.L^{num} - P_i.F^{den}) - 1$ and use it to construct the $\vec{M}_{i'}^{trnl}$. Note that -1 in order to ensure that rounding errors are in favor of the pool.

We assume that for the relaxed case *apply* : $P_i \rightarrow S \rightarrow P_{i'} \oplus \mathbb{1}$ satisfies the following conditions:

$$\boxed{\begin{array}{rl} P_{i'}.conf = & P_i.conf \\ P_{i'}.mut = & P_i.mut \\ \Delta T^q \cdot \text{f\_num\_rev} \geq & -\Delta M^q \cdot P_i.P^{num} \\ \sum_{k=1,k\neq q}^{k=n} \Delta T^k = & 0 \\ \Delta M^b = & \Delta M_{tr}^b \\ check\_invariant(P_i.A, P_i.n, P_i.\vec{d}, W, \vec{M}_i^{tr}, \vec{M}_{i'}^{trnl}) = & 1 \end{array}}$$

We assume that for the exact case *apply* : $P_i \rightarrow S \rightarrow P_{i'} \oplus \mathbb{1}$ satisfies the following conditions:

$$\boxed{\begin{array}{rl} P_{i'}.conf = & P_i.conf \\ P_{i'}.mut = & P_i.mut \\ \Delta T^q \cdot \text{f\_num\_rev} \geq & -\Delta M^q \cdot P_i.P^{num} \\ \sum_{k=1,k\neq q}^{k=n} \Delta T^k = & 0 \\ \Delta M^b = & \Delta M_{tr}^b \\ check\_invariant\_exact(P_i.A, P_i.n, P_i.\vec{d}, W, \vec{M}_i^{tr}, \vec{M}_{i'}^{trnl}) = & 1 \end{array}}$$

2.5 DAO Actions

All DAO-actions is validated by the proxy-DAO script. We assume that since main pool contract is involved, proxy-DAO script checks only validity of mutable pool datum fields $\{P^{num}, L^{num}, Z, Q, A, \vec{T}\}$ updates.

2.5.1 Update liquidity provider fee

We assume that *apply* : $P_i \to U_{ulf} \to Pi' \oplus \mathbb{1}$ satisfies the following conditions:

$$\boxed{\begin{array}{rl} P_i.LP_{edit} = & 1 \\ P_{i'}.I = & P_i.I \\ P_{i'}.A = & P_i.A \\ P_{i'}.Q = & P_i.Q \\ P_{i'}.\vec{T} = & P_i.\vec{T} \\ P_{i'}.\vec{M} = & P_i.\vec{M} \\ P_{i'}.Y^l = & P_i.Y^l \\ P_{i'}.I = & P_i.I \\ P_{i'}.L^{num} = & U_{ulf}.t \\ P_{i'}.P^{num} = & P_i.P^{num} \\ P_{i'}.Z = & P_i.Z \end{array}}$$

2.5.2 Update protocol fee

We assume that *apply* : $P_i \to U_{upf} \to Pi' \oplus \mathbb{1}$ satisfies the following conditions:

$$\boxed{\begin{array}{rl} P_{i'}.I = & P_i.I \\ P_{i'}.A = & P_i.A \\ P_{i'}.Q = & P_i.Q \\ P_{i'}.\vec{T} = & P_i.\vec{T} \\ P_{i'}.\vec{M} = & P_i.\vec{M} \\ P_{i'}.Y^l = & P_i.Y^l \\ P_{i'}.I = & P_i.I \\ P_{i'}.L^{num} = & P_i.L^{num} \\ P_{i'}.P^{num} = & U_{upf}.t \\ P_{i'}.Z = & P_i.Z \end{array}}$$

2.5.3 Update treasury address

We assume that *apply* : $P_i \to U_{uta} \to Pi' \oplus \mathbb{1}$ satisfies the following conditions:

$$\boxed{\begin{array}{rl} P_{i'}.I = & P_i.I \\ P_{i'}.A = & P_i.A \\ P_{i'}.Q = & P_i.Q \\ P_{i'}.\vec{T} = & P_i.\vec{T} \\ P_{i'}.\vec{M} = & P_i.\vec{M} \\ P_{i'}.Y^l = & P_i.Y^l \\ P_{i'}.I = & P_i.I \\ P_{i'}.L^{num} = & P_i.L^{num} \\ P_{i'}.P^{num} = & P_i.P^{num} \\ P_{i'}.Z = & U_{uta}.z \end{array}}$$

2.5.4 Update stake credential

We assume that $apply : P_i \to U_{usc} \to Pi' \oplus \mathbb{1}$ satisfies the following conditions:

$$\boxed{\begin{array}{rl} P_{i'}.I = & P_i.I \\ P_{i'}.A = & P_i.A \\ P_{i'}.Q = & U_{usc}.q \\ P_{i'}.\vec{T} = & P_i.\vec{T} \\ P_{i'}.\vec{M} = & P_i.\vec{M} \\ P_{i'}.Y^l = & P_i.Y^l \\ P_{i'}.I = & P_i.I \\ P_{i'}.L^{num} = & P_i.L^{num} \\ P_{i'}.P^{num} = & P_i.P^{num} \\ P_{i'}.Z = & P_i.Z \end{array}}$$

2.5.5 Update amplification coefficient

We assume that $apply : P_i \to U_{uac} \to Pi' \oplus \mathbb{1}$ satisfies the following conditions:

$$\boxed{\begin{array}{rl} P_i.A_{edit} = & 1 \\ P_{i'}.I = & P_i.I \\ P_{i'}.A = & U_{uac}.a \\ P_{i'}.Q = & P_i.Q \\ P_{i'}.\vec{T} = & P_i.\vec{T} \\ P_{i'}.\vec{M} = & P_i.\vec{M} \\ P_{i'}.Y^l = & P_i.Y^l \\ P_{i'}.I = & P_i.I \\ P_{i'}.L^{num} = & P_i.L^{num} \\ P_{i'}.P^{num} = & P_i.P^{num} \\ P_{i'}.Z = & P_i.Z \end{array}}$$

2.5.6 Withdrawal treasury

We assume that $apply : P_i \to U_{wt} \to Pi' \oplus \mathbb{1}$ satisfies the following conditions:

$$\boxed{\begin{array}{rl} P_{i'}.I = & P_i.I \\ P_{i'}.A = & P_i.A \\ P_{i'}.Q = & P_i.Q \\ P_{i'}.\vec{T} = & U_{wt}.\vec{t_u} \\ P_{i'}.\vec{M} = & P_i.\vec{M} - P_i.\vec{T} + U_{wt}.\vec{t_u} \\ P_{i'}.Y^l = & P_i.Y^l \\ P_{i'}.I = & P_i.I \\ P_{i'}.L^{num} = & P_i.L^{num} \\ P_{i'}.P^{num} = & P_i.P^{num} \\ P_{i'}.Z = & P_i.Z \end{array}}$$