log(swap)

A constant-LVR liquidity primitive with fungible positions and model-free fee accounting

In log we trust

Unimodular Research, draft v0.8, 2026-11-08

Abstract. log(swap) is a new liquidity primitive introducing the **log invariant**. Holdings that satisfy the invariant replicate $V(p) \propto \log p$; a **log position** is a claim on such holdings.

Four properties follow. **Market impact is predictable**: log-price slippage is $\Delta x = \Delta Q/L$, so the displayed liquidity L fixes the impact exactly. **Liquidity exposure is explicit**: the base leg stays worth exactly its exposure at every price ($pR = L$), so what is at risk never drifts; LVR is constant in price, so one fee compensates it uniformly, income and cost accrue on the same path statistic, and an LP's realized edge — an APY that means something — is one subtraction. **Capital is efficient**: a position holds no liquidity below its floor, so liquidity concentrates where it is placed. **Positions are affine**: active positions sum to one aggregate position, so a book of floors prices a swap as a single curve.

0. Introduction 2
Part I: The primitive 2
1. The log invariant 2
2. Swaps 3
3. Liquidity 3
4. Capping it 4
5. Many bases, one quote 5
6. Positions add 5
7. Fees as a path statistic 5
Part II: The protocol 6
8. Two complementary implementations 6
9. The contracts and their surfaces 8
10. Representing the state 9
11. Fee machinery as built 10
12. Settlement: one lock per manager 10
13. Numerics, guardrails, verification 11
Part III: Comparison and extensions 11
14. Comparison 12
15. Composition with credit: Leverage *(work in progress)* 12
16. A variance market 12

0. Introduction

Part I is the mathematics, contract-free: the floored position and its invariant, then what a swap costs on it and what its liquidity is, then the two extensions it admits (an upper cap, and n bases against one quote), and the fee and aggregation properties all three share. It ends on the fact that forces everything after it: **the cap and the basket spend the same free constant**, so no single pool can offer both.

Part II is the protocol, and opens by resolving that. The primitive leads to **two complementary implementations**. The **fungible** one, the **F-pool**, keeps one floor for every share, which makes claims interchangeable, deletes fee accounting outright, and is what enables the multi-asset features: the basket and the launchpad, the two products that want no range at all. The **concentrated** one, the **C-pool**, gives each LP its own floor and cap, which buys concentration and two-sided resting orders, and costs the fee accumulators and multi-asset. The rest of Part II is how both are built. Part III compares and extends.

Part I: The primitive

One curve, simple and elegant mathematics. No contracts appear in this part.

1. The log invariant

Fix a **quote** asset (the numéraire) and a **base** asset, write p for the price of base in quote, and work in $x = \log p$.

The primitive is the invariant, a relation between reserves alone. A pool holding base R and quote Q runs the **log invariant** if

$$\boxed{R\, e^{Q/L} = \frac{L}{p_{\min}}}$$

for an **exposure** $L > 0$ in quote units and a **floor** $p_{\min}$, $\xi = \log p_{\min}$. That relation is the curve a swap moves along, and everything below is either a consequence of it or a variant of it. A **log position** $\Lambda(L, \xi)$ is a claim on holdings that satisfy it.

The holdings follow from the curve, not the other way round. Differentiating the invariant gives $dQ/dR = -L/R$, so the marginal price is

$$p = -\frac{dQ}{dR} = \frac{L}{R},$$

and substituting back fixes the state at any point on it:

$$R = \frac{L}{p} \quad \text{(base)}, \qquad Q = L(x - \xi) \quad \text{(quote)}, \qquad V = Q + pR = L(1 + x - \xi).$$

At $p = p_{\min}$ the position is all base and $Q = 0$: the floor is where the quote buffer runs out. Below it the position is **dormant**: it holds $L/p_{\min}$ base, frozen, earning nothing until price returns. Dormancy is the model's one nonlinearity. Upward, $R = L/p$ is an asymptote, so the position is never fully converted to quote and never needs re-entry.

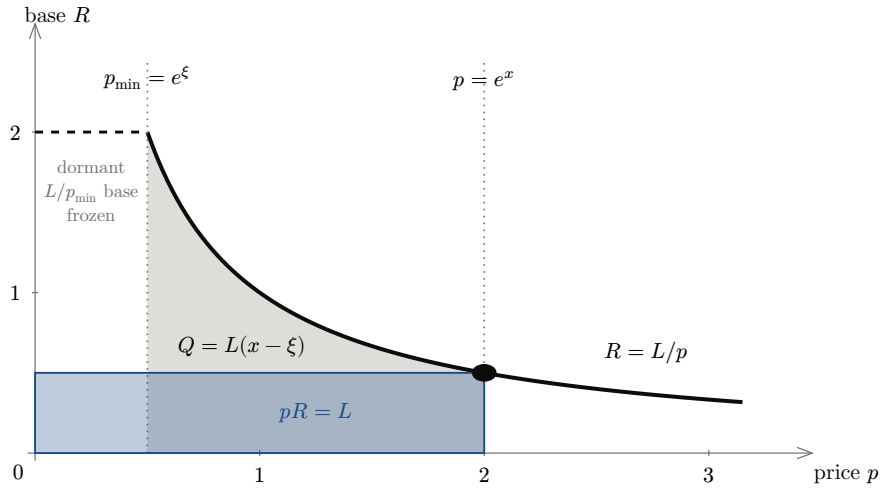


Figure 1: The base leg $R = L/p$, drawn with $L = 1$, $p_{\min} = 1/2$, $p = 2$. Q is the area under the curve from the floor to the price, $\int_{p_{\min}}^{p} L\, \mathrm{d}p'/p' = L(x - \xi)$; the rectangle under the curve at the price is $pR = L$, the same area at every price (**I1**). At the floor the quote has run out; below it the holdings freeze at $L/p_{\min}$.

The reserves are the state; the drawdown $\ell = x - \xi = Q/L \geq 0$ is read off them, and L is the size. **Three identities** carry everything that follows:

	identity	reading
I1	$pR \equiv L$	the marginal-price relation above, read as a statement about value: the base leg is worth L at every price, so L is both size parameter and the quote value of the base holdings
I2	$dQ = L\, dx$	the quote leg is linear in log-price
I3	$p^2\lvert V''(p)\rvert = L$	dollar gamma is constant, since $V''(p) = -L/p^2$

Where the shape comes from. I3 read backwards is a differential equation, $p^2 V'' = -L$, whose general solution is

$$V(p) = L \log p + b\, p + c.$$

The invariant above is the $b = 0$ member, with c fixed by $Q(p_{\min}) = 0$: the **floored** form, and the simplest. The remaining freedom is one constant, b, and §4 is what happens when it is spent.

2. Swaps

A swap moving $x \to x + u$ exchanges

$$\Delta Q = L\, u, \qquad \Delta R = L\big(e^{-x} - e^{-(x+u)}\big),$$

closed-form in one exponential. The average execution price is

$$\frac{\Delta Q}{\Delta R} = e^{x}\, g(u), \quad g(u) = \frac{u}{1 - e^{-u}}, \quad \log p_{\text{exec}} = x + \frac{u}{2} - \frac{u^2}{24} + O(u^4).$$

Two consequences. **Price impact in log space does not depend on price**: by **I2** a trade of quote size ΔQ moves x by $\Delta Q/L$, wherever price sits. L is the liquidity (§3). And **the fill lands near the log-midpoint of its own move**: to first order an equal-weight average of the pre- and post-trade states, with $u^2/24$ the leading correction. §11 uses this to fix the memory of a volatility estimator computed from Σ.

3. Liquidity

Gamma is liquidity. What a trade prices against is the curvature of the position: the liquidity a swap meets at p is the dollar gamma $\lambda(p) = p^2\lvert V''(p)\rvert$, in quote units; Uniswap v3′s liquidity parameter is exactly this quantity. More gamma per dollar of position is more fill per dollar of position.

For the log position, gamma is constant. By **I3**, $\lambda = L$: the same liquidity at every price, and that is what lets L be called *the* liquidity with no price attached.

And it is higher per unit of position value. With $V = L(1+\ell)$,

$$\frac{\lambda}{V} = \frac{1}{1+\ell}, \quad \text{against} \quad \frac{\lambda}{V} \equiv \frac{1}{4} \quad \text{for } xy = k$$

(there $V = 2\sqrt{kp}$ and $p^2|V''| = V/4$ at every price). Liquidity per unit of capital is up to $4\times$ constant product (exactly $4/(1+\ell)$, maximal at the floor), because the position holds nothing below its floor: all of its capital works where price is, instead of being spread over $(0,\infty)$. This is the abstract's *concentration for free*.

The bill for liquidity is LVR, and it is the same λ. Under geometric Brownian motion with volatility σ, loss-versus-rebalancing runs at $\frac{1}{2}\sigma^2 p^2|V''| = \frac{1}{2}\sigma^2\lambda$ for *every* invariant: no curve buys liquidity at a discount, so the $4\times$ above costs nothing per unit of liquidity. For the log position

$$\text{LVR} = \frac{1}{2}\,\sigma^2 L,$$

independent of price, and unchanged by the cap of §4. The revenue needed to break even against informed flow is the same wherever price sits. In Uniswap v2 the same rate scales as $\sqrt{p}$, so the adverse-selection bill depends on where price happens to be. Constant LVR is what makes a single global fee calibration meaningful (§7).

Scale-covariance. V'' is homogeneous of degree -2, so $p \mapsto \lambda p$ acts as $(Q, R) \mapsto (Q + L\log\lambda, R/\lambda)$ and preserves the form. Floors enter only through $x - \xi$.

4. Capping it

The floored position never fully converts to quote. Setting $b = -L/p_{\max}$ adds the missing boundary. Holdings become

$$R = L\left(\frac{1}{p} - \frac{1}{p_{\max}}\right), \quad Q = L(x-\xi), \quad V = L(1+x-\xi) - \frac{Lp}{p_{\max}},$$

and the invariant gains exactly one term:

$$\boxed{\left(R + \frac{L}{p_{\max}}\right) e^{Q/L} = \frac{L}{p_{\min}}}$$

with §1 recovered as $p_{\max} \to \infty$. Write $\mu = \log p_{\max}$ and call this $\Lambda(L, \xi, \mu)$. At $p_{\max}$ the position is all quote and goes dormant above, mirroring the floor.

Why a linear term, and nothing else. Two things must survive. Since $R = V'$ and $Q = V - pV'$, a term $b\,p$ contributes b to R and **nothing to Q**, so Q is untouched, **I2** holds unchanged, and the whole fee apparatus of §7 carries over verbatim. And $b\,p$ has zero second derivative, so V'' is unchanged and **I3**, constant LVR, survives too. Within the family of §1 the linear term is the only freedom there is, which is the sense in which the cap is determined rather than chosen. Other caps exist outside that family ($-\log(1 - p/p_{\max})$ also drives R to zero), at the cost of constant LVR.

A capped position is a difference of two floored ones, at every price:

$$\Lambda(L, \xi, \mu) = \Lambda(L, \xi) - \Lambda(L, \mu).$$

Note the sign: the difference contains a **short** leg, and since $\log p$ and p are linearly independent, no *positive* combination of floored positions produces a cap. §8.2 records what that forces on a pool that lets positions be held separately.

5. Many bases, one quote

The second extension widens the base side. Take $V = \sum_j L_j \log p_j + c$, so each base leg keeps $p_j R_j \equiv L_j$: **I1** holding per asset. There is still one free constant, hence **one floor for the whole pool**. With $x_j = \log p_j$, $L = \sum_j L_j$ and weights $w_j = L_j/L$, define the **composite index**

$$\bar{x} = \sum_j w_j x_j = \log \prod_j p_j^{w_j},$$

the log of the L-weighted geometric mean of prices. Then $Q = L(\bar{x} - \bar{\xi})$ and $V = L(1 + \bar{x} - \bar{\xi})$ with the weighted average log-floor $\bar{\xi} = \bar{x} - Q/L$ (the implementation calls it `logFloor`), and the invariant becomes

$$\boxed{\left(\prod_j R_j^{w_j}\right) e^{Q/L} = \frac{L}{e^{\bar{\xi}}}}$$

The multi-asset invariant is the log invariant on the basket index. Here $x_j = \log p_j$, and bars denote weighted averages: $\bar{x}$ for log-price and $\bar{\xi}$ for the log-floor. For one base, these reduce to $x = \log p$ and ξ. Everything else in Part I carries over using these averages. Four properties are specific to $n > 1$:

- **Balancer between the bases.** At constant Q the invariant reduces to $\prod_j R_j^{L_j} = \text{const}$. Base-to-base is native and exact at $\Delta Q = 0$.
- **LVR is correlation-free.** V is separable, so its Hessian is diagonal: $\text{LVR} = \frac{1}{2}\sum_j \sigma_j^2 L_j$, no covariance terms and no cross-asset price impact. The price paid is that cross-pair liquidity is the series combination $1/L_{\text{eff}} = 1/L_j + 1/L_k$.
- **Base-to-base needs no quote and cannot reach the floor.** A $\Delta Q = 0$ move holds $\bar{x}$ invariant, so it executes in a pool that has never held quote.
- **The basket auto-rebalances.** $p_j R_j \equiv L_j$ makes the holdings a constant-value-weight portfolio.

The cost is that the floor is a hyperplane, not a box: one shared $\bar{\xi}$ means one member's crash spends the common buffer. That is a risk statement rather than a mathematical one, and it is why baskets suit correlated assets.

The two extensions do not compose freely. Both §4 and §5 spend the same constant. A cap is a second boundary on one price; a basket is one boundary on an index of n prices. Part II is where that tension is resolved, and it is the whole reason the protocol builds two pools rather than one.

6. Positions add

Over positions that are simultaneously active, parameters map to holdings affinely:

$$Q_i = L_i(x - \xi_i), \quad R_i = L_i/p \quad \Longrightarrow \quad L = \sum_i L_i, \quad R = \frac{L}{p}, \quad Q = L(x - \bar{\xi}), \quad \bar{\xi} = \frac{\sum_i L_i \xi_i}{\sum_i L_i}.$$

A sum of active log positions is a log position, with floor the L-weighted mean of the floors. Two Uniswap v3 ranges do not compose into a Uniswap v3 range. The obstruction to global affinity is the dormancy kink of §1: below its floor a position leaves the sum.

Two corollaries matter later. A pool holding many floors can price a swap against a single aggregate $(L, \bar{\xi})$ between crossings (§8.2). And a blend of two adjacent floors *is* one position at the weighted mean, so a discrete set of floors spans a continuum exactly.

7. Fees as a path statistic

Charge a percentage ϕ of the quote leg. By **I2** the quote leg is linear in x, so this is a rate on displacement:

$$\text{fee} = \phi\, L\, |dx| \quad \Longrightarrow \quad dF = \phi\, |dx|, \quad F = \phi \cdot \text{TV}(\log p),$$

with F measured per unit of exposure. A base-side fee does not take this form (it carries a $1/p$ weight), which is why fees here are quote-only.

The edge. Track $\Sigma = \int (dx)^2$ alongside F. By §3 the informed-flow loss per unit L is $\frac{1}{2}\Sigma$ on the realized path, so

$$\boxed{\text{edge per unit} \quad L = F - \frac{1}{2}\Sigma}$$

Two accumulated numbers and a subtraction: no volatility model, no benchmark portfolio, no oracle. They are commensurable because both are path statistics of the same $\log p$. Simulation at $\sigma = 60\%$, one year of 12-second steps, 40 paths: $\mathbb{E}[F] = 0.180013$ against $\mathbb{E}\left[\frac{1}{2}\Sigma\right] = 0.180020$, theory 0.18.

Calibration. Set $\phi = \kappa\sigma$ and equate expected income with the bill. For GBM sampled at Δt, $\mathbb{E}|\Delta x| = \sigma\sqrt{2\Delta t/\pi}$ against a bill of $\frac{1}{2}\sigma^2\Delta t$, giving

$$\kappa^\star = \sqrt{\pi/8} \approx 0.6267,$$

at which the edge is zero in expectation under those assumptions. The sign of the realized edge then reports whether the fee over- or under-charged on the path that happened. This is how the fixed fee is **sized**, one constant per market class (§11), rather than a quantity re-estimated on-chain.

Fees translate the floor. A quote injection f at fixed (L, x) cannot leave the position unchanged; since $Q = L(x - \xi)$ it is a floor translation,

$$\Delta\xi = -f/L.$$

Compounding is a relabelling rather than a trade: there is no ratio to rebalance into. §8.1 makes this identity the F-pool's entire fee mechanism.

Part II: The protocol

The primitive has two natural, purpose-built implementations. The contracts appear here.

8. Two complementary implementations

Part I left one thing unresolved. The cap of §4 and the basket of §5 both spend the same free constant: a cap is a second boundary on a single price, a basket is a single boundary on an index of n prices. **No one implementation can offer both**, so the primitive leads to **two complementary implementations**, the **fungible** and the **concentrated**, and the line between them is *how many claims a pool divides itself into.*

Keep one floor for every share, and claims are interchangeable: the fungible dimension. Then §7′s floor-translation identity does the whole fee accounting: a fee moves the one floor and every share is credited at once, with nothing stored. Interchangeability is also exactly what §5 needs: in-kind fees on n different bases can be distributed through a single shared $\bar{\xi}$ only when no claim has to be told apart from another. So fungibility is what enables the multi-asset feature **for free**, and deletes fee accounting entirely, at the cost of any per-LP boundary and hence any concentration.

That is the **F-pool** (§8.1), and it carries the two products where nobody wants a range in the first place:

- **multi-asset** ($n > 1$): inventory as one productive, auto-rebalanced share, with native base-to-base routing (§5);
- **launchpad** ($n = 1$): where the product *is* the shared floor: a structural bid under a token, deepening on its own from fee income.

Give each LP its own floor and cap, and claims must be told apart: the concentrated dimension. By §6 the aggregate of the active positions is still a log position, so the pool can price a swap against one aggregate between floor crossings while each LP keeps its own boundaries. That is concentration, and by §4 it also turns a position into a resting order in either direction. The bill is that fees can no longer move

a shared floor (the accumulators come back) and that per-asset attribution across n bases would need one accumulator per base with n-valued snapshots at every boundary. So this pool is **one base against one quote**.

That is the **C-pool** (§8.2).

	F-pool (*fungible*)	**C-pool** (*concentrated*)
the floor	one, shared by every share	one per LP
the claim	one fungible claim for the pool	one per (floor, cap) class
assets	n bases, one quote	one base, one quote
concentration	none (liquidity is shared)	per-LP floor and cap
fee accounting	none (the floor *is* the accumulator)	explicit, per-class
resting orders	—	capped positions, both directions
for	capital that wants risk-management guarantees	LPs who want their own range

The letters name how many claims, not fungibility as such: C-pool positions are fungible too. Concentration and width fall out of that one choice in opposite directions.

8.1 The F-pool: one floor, one claim

Give every share the same floor. Claims are then **homogeneous** (every unit interchangeable), and §7′s identity does the accounting.

A fee accrues to all shares at once as $\Delta\bar{\xi} = -\text{feeValue}/L$, so $\bar{\xi} = \bar{\xi}_0 - F$: **the floor is the fee accumulator.** It is not stored but derived, $\bar{\xi} = \bar{x} - Q/L$, from reserves on every read. Accrual is denomination-blind: a quote fee lands in Q at fixed $\bar{x}$; an in-kind base fee marks the derived index down by its value per unit L.

This removes the fee-growth accumulator, per-tick snapshots, per-holder fee slots, the collect-swap-remint cycle, and every derived reserve: the reserves *are* the state (§10). Compounding is not a cheap operation but the absence of one.

The floor has one lever, and a harvest is a dividend. Raising $\bar{\xi}$ releases $L\Delta\bar{\xi}$ in pure quote at fixed L and R — price-neutral — and every share receives its slice: the amount accrues through a per-share accumulator, each holder claims its part or takes it with its next burn, and the operator's own is paid in the same call. A harvest takes the income the fees have earned and, except on a pad, may also take capital down to the policy's quote buffer. The floor moves down only as fees land — no entry puts quote back under a live generation; a deeper bid is a new generation (below). At $\bar{\xi} = \bar{x}$ the pool holds no quote — a resting ask, the state a launch is born in. The mark $\bar{x}$ is derived from reserves and moved only by trades, so no balance-sheet operation lowers the ask: within a generation there is no path by which the operator lowers the floor and then buys the book.

One policy, four parameter sets. The policy is named in the pool's key and answers for the pool for life; its kinds are configurations of one contract. A **public** policy is open: anyone initializes, the salt is zero, no gate is ever set, and the protocol is the authority. A **desk** policy is a sponsor's, and gates swaps and mints per pool (open, listed, or the operator only). A **POL** policy is a sponsor's own liquidity, with an income tax and a quote buffer. A **pad** sends its seed to `0xdead`, harvests income only and never closes, so only income can leave; its authority is a launcher contract (design). The authority is one immutable address per policy and appoints each pool's operator, by default itself; the operator funds the pool, harvests it and closes it, paying the seed and holding its shares.

A pool is relaunched, never re-floored. The operator *closes* a generation at any bid: the income is harvested first, then the liquidity becomes a frozen in-kind bundle, a *frozen generation* that every old share redeems pro rata whenever its holder likes. The operator funds the *next generation* with `deployLiquidity`, from its own tokens, at whatever prices, weights and cap it names over the key's assets; `multicall(close, redeem, deployLiquidity)` is the whole step in one transaction. A pad never closes. The pool id, its policy and its gate modes carry over; the share id carries the generation, so a frozen claim and a live share are

distinct assets. An LP rolls the claim into the new generation through the router or keeps it; mispricing the seed hurts the seeder first, which is what stands in for an oracle.

The two products differ only in who holds the floor.

	the basket ($n > 1$)	**the launchpad** ($n = 1$)
policy kind	public or desk	pad; a POL is its one-base, sponsor-held sibling
authority	one per policy: the protocol, or a sponsor	a launcher contract (design), paying the creator
who moves the floor	the pool's operator: `harvest`, `close`	the operator; income only, never closes
Q at deploy	seeded two-sided by the operator	often $Q = 0$: an all-base resting ask
what it sells	one productive, auto-rebalanced share	a floor the creator cannot walk back
the risk	a shared floor is a hyperplane	the operator harvests, so the manager bounds it: the seed is `0xdead`'s, income only

The **basket** is inventory as one share: auto-rebalanced to constant value weights, any leg pullable by base-to-base without quote and without touching the floor. It is workable lending collateral, since $V = L\big(1 + \bar{x} - \bar{\xi}\big)$ is affine in $\bar{x}$ (§15). The **launchpad** sells the floor itself: a bid that deepens from fee income, of which only the income can be harvested.

8.2 The C-pool: a floor per LP

Give each LP its own floor. Claims become **heterogeneous**.

What is lost. The floor can no longer absorb a fee, so F becomes an explicit accumulator with per-class attribution (§11). And per-asset attribution across n bases would need n accumulators with n-valued snapshots at every boundary (buildable, but against the posture), so a C-pool is one base against one quote. A concentrated basket is assembled from one C-pool per constituent.

What is gained. By §6 the aggregate of active positions is itself a log position, so a swap prices against a single $\big(L_a, \bar{\xi}\big)$ between crossings; crossing a floor adds or removes that floor's L_i. Liquidity concentrates where LPs put it, and L_a becomes a step function of price.

Positions stay fungible. §6′s aggregation carries no per-holder path dependence, so per-holder state collapses to the class: a position is a share balance in a (floor, cap) class. Claims are heterogeneous across classes and homogeneous within one: enough for §8.1′s accrual argument to run one class at a time. This is why positions are fungible tokens rather than NFTs.

A coarse ladder suffices. By §6′s second corollary a blend of two adjacent floors is one position at the weighted mean, so a discrete ladder spans a continuum of floors. Ladder density is an interface parameter, not a precision parameter.

The cap makes resting orders a presentation. By §4 a capped position reaches all quote at $p_{\max}$ and goes dormant above. A resting **buy** is a capped position minted above spot: quote-only on entry, converting to base as price falls to the floor. A resting **sell** is a dormant mint below completion. Both earn fees while the price is inside their band, partly filled included, and are edited in place. There is no passive-order mechanism in the protocol, and a partially filled order is a position mid-band.

What the cap forbids. By §4 a capped position is a difference containing a short leg, so it is held as one indivisible claim. Splitting it would let a holder sell the long leg and keep the short. Because the primitive carries one L, shed at the cap as it was taken on at the floor, the representation has no way to express a net short.

9. The contracts and their surfaces

Two managers, a router and a lens for each, no factory. The F-pool is `ManagerF.sol`, a singleton: every fungible pool in one contract, ERC-6909 shares, its own flash-accounting lock, one contract under the bytecode limit with no facet. Mathematically it is the C-pool's degenerate case (one class, one floor),

so one spec covers both. An F pool is keyed by the hash of its `PoolKeyF` (quote, bases, swap-fee index, policy, salt), so its asset list is fixed for life. `initialize(key)` lists it unfunded; its operator funds it with `deployLiquidity`, which sets the weights for that generation. The C-pool is `ManagerC.sol`, a singleton holding every market, keyed by the hash of its `PoolKeyC`: the economic tuple (pair, ladder spacing, policy, creator, salt, swap fee, mint fee), with initial price excluded since it is state rather than economics. Creating a market is a call, not a deployment: `initialize(key, initialLogPrice)`, from the key's creator, who alone may mint until the market's backstop is seeded. The key rides in calldata at `initialize` only; its fields are stored there, and every later call names the market by `poolId`, as on the F-pool. Each manager has its own router (`RouterF.sol`, `RouterC.sol`), each its own Permit2 spender — a second spender is a second signature, never a second on-chain approval — and its own read-only, freely redeployable lens (`LensF.sol`, `LensC.sol`); nothing in a manager points at either. Both managers inherit the lock, the ledger and settlement from one base contract, `ManagerBase`.

The manager's 24 KB runtime is the binding constraint, which is why the external surface is lean: conveniences that compose from primitives live in the routers, derivations recomputable from public state in the lenses. The packaging is Uniswap v4's, taken for two dividends (one bytecode budget, and cross-market flash netting) while rejecting hooks, two-sided ranges and two-token fee growth.

	F-pool (`ManagerF`)	**C-pool** (`ManagerC`)
deployment	`initialize(key)`, then the operator's `deployLiquidity`	`initialize(key, initialLogPrice)`; `poolId` after
the claim	one ERC-6909 share id per pool	ERC-6909 id per (floor, cap)
the floor	derived $\bar{\xi} = \bar{x} - Q/L$, never stored	one per position, on a bitmap ladder
enter / exit	`mint` / `burn`, pro rata, in kind	`mint` / `burn` per class, `update` in place
swap	quote↔base, base↔base at $\Delta Q = 0$	one path, looping the ladder
the floor	`harvest`, a dividend to every share; `close` · `redeem` · `deployLiquidity`	none (no shared floor)
fee state	none; fee and edge are derivations	F, per-tick flips, per-holder snapshots
fee policy	fixed ϕ from an 8-entry list; Σ kept	fixed ϕ in the key; Σ kept
the state	(L, Q, R_j) — the reserves of the live generation; the mark $x_j = \ln(w_j L/R_j)$ derived; a frozen generation keeps (S, Q, R_j)	x, L_{active}, the signed ladder; reserve shadows for isolation
settlement	its own lock, spanning every F pool	its own lock, spanning every C market
single-sided	`zapIn` / `zapOut` / `rollIn`	`zapIn`, `mintBlended`, `swapExactInPath`

What a user decides. The trader's screen is the same in both: amount in, minimum out. The LP's is not. An F-pool asks one question: how much to deposit. A C-pool asks two: *size*, which is L directly since **I1** makes the base leg's value price-invariant, so one number sets both LVR and fee accrual; and *risk*, the floor, offered as a log-drawdown ℓ from which implied leverage $(1+\ell)^{-1}$, the quote sidecar $Q = L\ell$, and the break-even σ derive. The two are orthogonal.

10. Representing the state

The C-pool stores the log price; the F-pool stores what it holds. The C-pool stores x, never p, as Uniswap v3 stores $\sqrt{P}$: with a ladder the reserve is virtual per active book and cannot be a state. By **I2** the quote leg and the fee accounting are linear in x, so exponentials stay confined to the base leg. The F-pool has one class of liquidity, so its holdings are a complete description: it stores (L, Q, R_j) and derives $x_j = \ln(w_j L/R_j)$ on read — $p_j R_j \equiv w_j L$ holds by definition, the seed names reserves and the price follows, and the rounding rule is the amount rule alone: what leaves rounds down, what enters rounds up, the state is never rounded. A leg is never emptied and never holds more than $L_j \cdot 10^{36}$, the logarithm's domain; with $L > 0$ the price is bounded on both sides by construction.

The ladder: a bitmap and a pointer. Live floors are one bit each over compressed indices $k = \xi/\delta_\xi$. Insert and remove are one `SSTORE`; search walks a 256-bit word at a time, measured flat in ladder depth

against $O(n)$ for a sorted array (2.5–4.8× at depth 100). The crossed set is *maintained* by a `currentK` pointer rather than derived from x, because a tick exactly at x is ambiguous: crossing up should count it, crossing down should not. The bitmap indexes the coarse LP ladder only: liquidity changes at LP boundaries, and x moves continuously between them.

One signed ladder carries caps. By §4 a capped class writes $+L$ at its floor and $-L$ at its cap on the same ladder. Tick deltas are signed, and $L_a \geq 0$ is cumulative rather than per-tick, since every position credits at ξ before debiting at $\mu > \xi$. Aggregate base is $R_{\text{total}} = L_a/p + \mathcal{D}$, where $\mathcal{D}$ (frozen holdings of not-yet-crossed legs) goes negative while short legs are dormant. Since $\{x > \mu\} \subset \{x > \xi\}$ pointwise in time, a capped position's credit minus debit is monotone and it cannot owe fees. A permanently locked uncapped class keeps L_a from vanishing when price crosses every cap.

11. Fee machinery as built

The C-pool's accumulator. A claim is $L_i \cdot \Delta F$ over the active interval, computed $O(1)$ by a Uniswap v3-style flipped per-tick accumulator. §7′s compounding identity survives: a quote injection is a floor translation $\Delta\xi = -f/L$, the same additive shift for every active position, preserving ladder spacing.

The mint fee. Newly *earning* exposure pays a fixed fraction of L. On the C-pool the fraction is a key field, $2\phi \leq \text{mintFee} \leq 2\%$, fixed for life and charged in quote; dormant and capped-out mints are exempt, and edits pay only on the net earning increase, so repositioning is free while the dormant-then-activate JIT bypass is closed. On the F-pool it is the entry after the swap fee on the fee list (below), taken in shares on outside mints. On both, the whole mint fee is the protocol's, paid to the treasury.

The protocol's take. Three channels, each bounded in the manager and fixed before anyone provides liquidity: the mint fee; on the C-pool a swap-fee tax, a share of every fee paid out, set at birth by the market's policy, at most 25% and zero on a market with no policy; on the F-pool a tax on harvested income, set by the policy, at most 50% (10% on a POL and 20% on a pad in the reference configuration).

The fee is fixed. A market quotes a **constant** ϕ, sized per market class with margin: majors around 5 bp, the long tail several times that. On the C-pool ϕ is a key field from 1 bp to `FEE_MAX` = 1%, or zero where a policy lists it; on the F-pool the key holds an index into `FEE_LIST` = [1, 2, 5, 10, 20, 50, 100, 200] bp. A base-for-base swap pays ϕ once, so the direct hop is the cheapest route. §7 is what sizes the fee — $\phi = \kappa^\star\sigma$ evaluated at the volatility the class is expected to live at — so calibration is a listing decision rather than a quantity maintained on-chain. One number per market, and nothing to explain to a router or a trader.

No kernel in the contract. §7′s matched gain made live, $\phi_{\text{eff}} = \max\left(\phi, \kappa\sqrt{\sigma^2_{\text{ema}}}\right)$, is not built: the key carries no κ and the swap path keeps no estimator. The contract keeps Σ, in every market, one term per swap — the swap's total $(\Delta x)^2$ regardless of ticks crossed, since variance is quadratic and segment-sampling would charge u^2/n for the same move split over n ticks. A kernel computed from Σ folds with decay weight $\alpha = \frac{1}{2}$, matching §2: each swap executes near the log-midpoint of its move, so the estimator observes at the weighting the pool executes at. Its constant is then the Jensen-corrected $\kappa^{\star\star} \approx 0.6733$, not $\kappa^\star$, since the $\alpha = \frac{1}{2}$ EMA is a biased σ estimate.

What fixed gives up. Cycle coverage. Above the volatility its margin covers, a fixed fee goes quietly underwater — Uniswap v3′s failure mode, a tier adequate in calm but not through the cycle. The defence is margin in ϕ plus the freedom to list a new market at a new fee, not automatic repricing. `FEE_MAX` bounds the worst half-spread a C market can quote.

Fixed fee, calculable kernel. Because Σ is public state, a fixed-fee market still publishes everything needed to compute the fee a kernel *would* have quoted, and the model-free edge $F - \frac{1}{2}\Sigma$, without being charged for the machinery. A kernel stays a claim about LVR that a reader can evaluate, not a default every listing inherits.

12. Settlement: one lock per manager

C-pool operations move no tokens. They accrue signed transient deltas (EIP-1153), settled when the lock closes, and the lock cannot close with any delta outstanding, debt or credit. One lock spans every market

in the manager, so a multi-hop route settles each token once and intermediates never move. An EOA path auto-locks per call; a router opens one lock across many operations, so blending two ladder rungs or funding a mint from a swap costs one transfer per token. A positive delta may instead be taken as a **token claim** (an ERC-6909 IOU on tokens left inside), which lets a market maker park inventory serving every market of that manager at once; claims are backed by the same close-time check.

This replaces the re-entrancy guard (a locker re-enters by design) with a different guarantee: re-entry is permitted, an unsettled delta is fatal. The F-pool carries its own lock of the same kind; the two managers share no lock, and each has its own router and lens.

13. Numerics, guardrails, verification

Solvency. On the F-pool the reserve is the state, so there is no derivation to keep consistent: a buy leaves $\lceil Re^{-u} \rceil$, the sale back pays $\lfloor L \ln(1 + b/R') \rfloor \leq L \ln(R/R') \leq Lu$, and a round trip cannot pay at any integer reserve — fuzzed down to a wei. On the C-pool the reserve is virtual per active book and re-derived from (L_{active}, x), so every token leg moves by that same derivation, a difference of two reads rounded in opposite directions (floor where base leaves, ceiling where it enters), never an independently evaluated closed form; an earlier one-signed version shorted the pool monotonically under one-directional flow until the last burn reverted. The tested invariant on both is that every share can always be burned.

Guardrails. Fee bounds as in §11. Seeding is the pool's own: on the F-pool only the operator funds a pool (`deployLiquidity`), and on the C-pool only the key's creator mints until the backstop is seeded. Creation and seeding are two steps, and an open seed would let anyone front-run the creator and fix both L and the mark. Parameters are validated at creation, so a mis-parameterised market reverts at `initialize` rather than going live dead. The manipulate-then-liquidate defence does not live in the swap path: the leverage extension gates liquidations on an oracle, and its satellite refuses to liquidate while pool price deviates from the oracle beyond a bound, so the manipulation blinds the mechanism it meant to exploit, and honest swaps are never refused.

Quoting is the router's: it runs the real swap in the manager's lock and reverts with the result, so quotes are exact and need no funded balances.

Verification. About 600 tests, including 25,600-call stateful invariant harnesses (solvency, reserve tracking, pointer consistency, signed-ladder consistency, fee backing, backstop integrity), each probed for vacuity by fault injection. 14 Halmos symbolic proofs cover the tick bitmap (search soundness and completeness over the full index range), the id packing (injectivity, over a 2^{64} key space no fuzzer covers), the id namespace (every position id carries the reserved bit, so no token claim collides with one), and the delta bookkeeping. The exponential swap math is SMT-intractable and is covered by symbolic re-derivation and fuzzing instead.

Part III: Comparison and extensions

Where the primitive stands against the alternatives, and what it admits beyond them. The extensions here are design, not built.

14. Comparison

	Uniswap v2	**Uniswap v3**	**F-pool**	**C-pool**
Value profile	$\sqrt{p}$	piecewise $\sqrt{p}$	$\log p$	piecewise $\log p$
LVR rate	$\propto \sqrt{p}$	range-dependent	constant	constant
Assets	one pair	one pair	n bases, one quote	one pair
Concentration	—	arbitrary bands	—	per-LP floor + cap
Positions compose	trivially	no	vacuous (one class)	yes, affinely
Position token	ERC-20	NFT	ERC-6909 share	ERC-6909 per class
Reposition	—	burn, collect, remint	—	one call, in place
Fee state	none	per-position, two-token	none	per-class accumulator
Fee compounding	automatic	collect → swap → remint	floor translation	floor translation
Fees as a path statistic	no	no	$F = \phi$ TV$(\log p)$	$F = \phi$ TV$(\log p)$
LP P&L readout	external	external	on-chain, $F - \frac{1}{2}\Sigma$	on-chain, $F - \frac{1}{2}\Sigma$
Limit orders	—	range + manual withdraw	—	capped position, both ways
Price state	reserves	$\sqrt{P}$ Q64.96	reserves; $\bar{x}$ derived	$x = \log p$, WAD
Deployment	one per pair	one per pair	singleton; a pool is a call	singleton; a market is a call

Two trades, one per pool. Against Uniswap v3, the C-pool cannot concentrate into an arbitrarily narrow band: liquidity spreads as $1/p$ above each floor. What it buys is composability, hence fungible classes rather than NFTs, uniform adverse-selection pricing, and path-statistic accounting; the floor and cap recover the useful part of concentration (a working range with single-asset entry and exit) without the part that breaks aggregation. The F-pool gives up concentration entirely, and is paid in the two things the C-pool cannot have: no fee accounting, and n assets in one pool.

15. Composition with credit: Leverage *(work in progress)*

Borrow against the position, with the position as collateral; which asset is borrowed picks the product. Quote debt D against $\Lambda(L, \xi)$ has $V = L(1 + x - \xi)$ affine in log-price, so $\text{LLTV} \cdot V \geq D$ inverts in closed form:

$$x_{\text{liq}} = \xi + \left(\frac{D}{\text{LLTV} \cdot L} - 1 \right).$$

Liquidation is a tick, not a search: a point on the ladder the pool already maintains, with no iteration and no auxiliary oracle geometry. Since $V = L\big(1 + \bar{x} - \bar{\xi}\big)$ is affine in the index too (§5), the same holds for an F-pool share, where liquidation is a tick on the index. At $D = L$ everything reads off $\text{CR} = 1/\text{LLTV}$: minimum buffer $\text{CR} - 1$, leverage $1/(\text{CR} - 1)$, auto-deleverage efficacy $\text{CR} - 1$. Defence is a three-lever stack (intra-block move limit primary, per-block close factor and aggregate cap as backstops), which lets the base fee sit at its LVR-optimal level instead of doubling as a manipulation deterrent. Fees accrue to the position rather than banking on transfer, so a borrowed position stays claimable by its holder.

16. A variance market

F and Σ make LP-side volatility a tradeable pair of legs: long pays ΔF and receives $\frac{1}{2}\Delta\Sigma$: a variance swap struck at the pool's own fee, with nothing transacting and no fee leakage.

Full derivations: `specs/math.md`. *The F-pool:* `specs/f-pools/f-pool.md`, *with multi-asset theory in* `specs/f-pools/ma.md`, *and the launch in* `specs/f-pools/launchpad/launchpad.md`. *C-pool mechanisms:* `specs/protocol.md`. *Roles and policies:* `specs/policy.md`. *Actors:* `specs/ux.md`. *Symbols:* `specs/glossary.md`. *Fixed-point details:* `specs/numerics.md`. *Adversarial analysis:* `specs/arbitrage.md`, `specs/leverage/lev.md`.