COUNCIL OF ALPHAS

Complete Technical Specification

An Evolutionary Multi-Agent Framework for Alpha Discovery

HackEurope 2026 | Team: The Greeks (Andreas + Markos)

Version: Final | February 22, 2026

An evolutionary multi-agent framework that prevents mode collapse in LLM strategy generation through enforced specialist diversity, niche-preserving selection, deterministic hybrid construction, and evidence-locked diagnostic refinement.

1. Data

Parameter	Value
Instrument	SOL-USD
Timeframe	1-hour candles
Source	Binance (parquet file)
Date Range	Jan 2022 - Feb 2026 (~36,000 bars)
Index	open_time (UTC datetime, already set as index)

Columns Kept from Raw Binance Data

Core OHLCV: open, high, low, close, volume

Extended: quote_volume, count, taker_buy_volume

2. Regime Detection

Three dimensions used exclusively as diagnostic buckets — NOT as strategy entry conditions.

2.1 Session Regime

Label	UTC Hours
ASIA	00:00 - 07:59
LONDON	08:00 - 12:59
NY	13:00 - 20:59
OTHER	21:00 - 23:59

2.2 Trend Regime

Indicator: SMA(50) slope over 3-bar lookback (pct_change(3)). Threshold: +/-0.0005

Label	Condition
UPTREND	slope > +0.0005
DOWNTREND	slope < -0.0005
CONSOLIDATION	-0.0005 <= slope <= +0.0005

2.3 Volatility Regime

ATR window: 24 bars (= 24 hours on 1h). Smoothing: SMA(20) of ATR.

Label	Condition
HIGH_VOL	ATR(24) > SMA20(ATR(24))

LOW_VOL	ATR(24) <= SMA20(ATR(24))

Total micro-buckets: 4 sessions x 3 trend states x 2 vol states = 24 micro-buckets

3. Triple Barrier Labeling (TBM)

3.1 Fixed Parameters (System-Wide)

Parameter	Value
Win multiplier	2.0 x ATR (Take Profit)
Loss multiplier	1.0 x ATR (Stop Loss)
Time horizon	24 bars (~24 hours on 1h)
ATR window	24 bars (24 hours on 1h)
Tie-break	stop_first (worst-case on whipsaw)

3.2 Label Values

Label	Meaning
+1.0	Long trade: TP hit (price went up)
-1.0	Short trade: TP hit (price went down)
0.0	Timeout: neither barrier hit in time_horizon
NaN	Whipsaw: both long AND short hit in same bar (untradable)

3.3 Logic

- Long scan and short scan run simultaneously for every candle
- Long takes priority: if long hits TP -> +1, else check short -> -1, else 0
- Whipsaw (both directions hit) -> NaN label (excluded from ML training)
- Every single candle in the dataset gets labeled

3.4 Output Columns (appended to State Matrix)

Column	Description
tbm_label	Oracle label: +1, -1, 0, or NaN
tbm_long_pnl	Exact fractional return if long trade taken
tbm_long_exit_idx	Row index where long trade exits
tbm_long_duration	Candles the long trade was open
tbm_short_pnl	Exact fractional return if short trade taken
tbm_short_exit_idx	Row index where short trade exits

tbm_short_duration	Candles the short trade was open

4. State Matrix

A single pre-computed pandas DataFrame saved as parquet. Built ONCE, loaded on every subsequent run. Every downstream component reads from it — nothing is recomputed.

4.1 Complete Column Schema (21 columns)

Column	Source	Type
open, high, low, close, volume	Raw Binance	float
quote_volume, count, taker_buy_volume	Raw Binance	float
ATR_24	StateMatrixBuilder	float (KEPT — used by backtester)
session	StateMatrixBuilder	str: ASIA/LONDON/NY/OTHER
trend_regime	StateMatrixBuilder	str: UPTREND/DOWNTREND/CONSOLIDATIO
vol_regime	StateMatrixBuilder	str: HIGH_VOL/LOW_VOL
tbm_label	core/labeling.py	float: +1/-1/0/NaN
tbm_long_pnl, tbm_long_exit_idx, tbm_long_duration	core/labeling.py	float/int/int
tbm_long_outcome	core/labeling.py	str: TP/SL/TIMEOUT
tbm_short_pnl, tbm_short_exit_idx, tbm_short_duration	core/labeling.py	float/int/int
tbm_short_outcome	core/labeling.py	str: TP/SL/TIMEOUT

4.2 Build Rules

- ATR_24 is KEPT — backtester uses it for leverage calculation

- Intermediate columns dropped before saving: sma_50, sma_50_slope_3, ATR_24_SMA_20

- Load from parquet if exists. Build and save if not. force_rebuild=True to override.

4.3 StateMatrixBuilder Parameters

```
StateMatrixBuilder(
    trend_slope_threshold=0.0005,  # updated from 0.001
    atr_window=24,                  # 24 bars = 24 hours on 1h
    vol_sma_window=20,
    tbm_win=2.0,                    # fixed system-wide
    tbm_loss=1.0,                   # fixed system-wide
    tbm_time_horizon=24,            # 24 bars = 24 hours on 1h
    tbm_tie_break="stop_first",
)
```

5. Specialist Agents

4 LLM agents (Claude Sonnet, temp=0), each locked to a distinct strategy family.

5.1 Families & Indicator Subsets

Family	Allowed Indicators
Trend	ema, hma, macd, adx, slope
Momentum	rsi, cci, roc, mfi, zscore
Volatility	natr, bollinger_bands, keltner_channels, choppiness_index
Volume	vwap, obv, cmf

5.2 Random Indicator Sampling

Before each strategy generation, IndicatorSampler randomly samples 2-4 indicators from the family pool. Sampled subset injected into prompt only — LLM cannot use what it cannot see. Prevents intra-specialist mode collapse.

5.3 Strategy Base Class

```
class Strategy(Indicators):
    name: str = "unnamed"
    family: str = "unknown"   # trend/momentum/volatility/volume/hybrid
    description: str = ""     # one sentence: what does this strategy do

    def generate_signals(self, data: pd.DataFrame) -> pd.Series:
        # Returns: 1 (long), -1 (short), 0 (flat)
        # Index must match data.index exactly. No lookahead.
        raise NotImplementedError
```

IMPORTANT: tbm_win and tbm_loss are NOT on Strategy objects — fixed system-wide in config.py

5.4 Generation Rules

- PoC: 1-3 strategies per specialist (MAX_STRATEGIES_PER_SPECIALIST in config.py)

- Score immediately after each generation (not batched)

- Max 3 code generation attempts per strategy with error feedback injection

- Parallel generation: asyncio.gather(return_exceptions=True)

- 60-second timeout per strategy execution

5.5 Code Validation Steps

Step	Check	Action on Failure
1	Syntax: compile(code)	Retry with SyntaxError message
2	Execution: strategy.generate_signals(sample_data)	Retry with exception message
3	Return type: valid pd.Series of 1/-1/0	Retry with type feedback
4	Trade count: must produce > 0 trades	Retry with guidance

6. Backtesting

Custom VectorizedBacktester — no VectorBT dependency. Numba-accelerated.

Parameter	Value
Fee	0.00040 (0.04% MEXC taker fee)
Risk per trade	0.5% of account equity
Initial capital	$100,000
Max leverage	20x
Capital lock	No overlapping trades allowed
TBM	Reads pre-computed columns from State Matrix

Trade Log Output Schema

Column	Description
entry_ts, exit_ts	Entry and exit timestamps
entry_index, exit_index	Row indices
duration	Candles open
side	1 (long) or -1 (short)
net_trade_return	Leveraged portfolio return after fee
account_balance	Compounding equity after trade
session, trend_regime, vol_regime	Regime tags from entry candle

7. Diagnostics Engine

Hierarchical bucket system. 5 metrics across all regime combinations.

7.1 Hierarchy (60 rows total)

Level	Description	Rows
GLOBAL	All trades combined	1
1D	By session, trend, or vol (separate)	4 + 3 + 2 = 9
2D	Session x Trend, Session x Vol, Trend x Vol	12 + 8 + 6 = 26
3D	Session x Trend x Vol (all combinations)	24
TOTAL		60

7.2 Output Schema

Column	Description
granularity	GLOBAL / 1D / 2D / 3D
session, trend_regime, vol_regime	Regime values or ALL
trade_count	N trades in bucket
win_rate	% winning trades
sharpe	mean_return / std_return (ddof=0, NaN if std=0 or n<2)
max_consecutive_losses	Longest losing streak (chronological)
sufficient_evidence	True if trade_count >= 30

Critic receives ONLY rows where sufficient_evidence = True

8. Fitness Function

```
Score = Global_Sharpe x ln(N) x Coverage
```

8.1 Component Definitions

Global_Sharpe: Sharpe from the GLOBAL row of diagnostics table.

ln(N): Natural log of total trade count. Rewards sample size with diminishing returns.

Coverage (trade-weighted):

```
active = 3D buckets where sufficient_evidence == True
profitable_trades = sum(trade_count for buckets where sharpe > 0)
total_trades = sum(trade_count for all active buckets)
coverage = profitable_trades / total_trades
```

8.2 Hard Eliminations (return -999 before scoring)

```
if global_row.sufficient_evidence == False: return -999
if isnan(global_row.sharpe): return -999
if len(active_3d_buckets) == 0: return -999
if total_trades_in_tradable_3d_buckets < 300: return -999
```

Negative Sharpe is allowed through — strategies with Sharpe between -5.0 and 0 can compete and be improved by the Scientist loop. The is_unviable() gate (Sharpe < -5.0) catches truly hopeless strategies before wasting API calls. The 300-trade minimum ensures sufficient evidence across tradable buckets.

8.3 Why This Formula Works

Component	What it catches
Global_Sharpe	Is this strategy profitable risk-adjusted? (negative allowed through)
ln(N)	Do we have enough evidence? Diminishing returns above ~1000 trades
Coverage	Is it profitable across the regimes where it actually TRADES?
300-trade minimum	Eliminates strategies that barely fire — not enough data to trust

Same formula used for both champions (Stage 3) and hybrids (Stage 5).

9. Niche Selection

Rule	Detail
Selection	Top 1 strategy per family by fitness score
Threshold	Score > 0 (must pass hard eliminations)
No viable champion	Family eliminated, pipeline continues with remaining
Minimum	Pipeline continues with as few as 1 champion

Order: Specialists generate -> Fitness scored -> Niche selection -> HybridBuilder -> Scientist -> Final fitness -> Ranking

10. HybridBuilder (Pure Python — No LLM)

Deterministic Python class. Takes up to 4 champions. Produces exactly 3 hybrids.

Hybrid 1 — Regime Router

For each of the 24 regime combinations, assign the champion with the highest Sharpe in that specific 3D bucket (sufficient_evidence=True). Fallback to best GLOBAL Sharpe champion if no sufficient evidence in that bucket.

Hybrid 2 — Consensus Gate

All champions vote. Fire long if >= 3/4 agree on long. Fire short if >= 3/4 agree on short.

```
votes = trend_signal + momentum_signal + vol_signal + volume_signal
long_signal = (votes >= 3)
short_signal = (votes <= -3)
```

Hybrid 3 — Weighted Combination

Champions weighted by fitness score. Weighted sum's sign determines direction.

```
weights = [fitness_trend, fitness_momentum, fitness_vol, fitness_volume]
weighted_sum = sum(w * s for w, s in zip(weights, signals))
signal = np.sign(weighted_sum)
```

10.4 Code Structure

All hybrids are inline Strategy subclasses with combination parameters as CLASS ATTRIBUTES (modifiable by the Scientist's Refiner). Champion strategy references are injected via _champion_strategies dict:

```
class RegimeRouterHybrid(Strategy):
    name = "regime_router"
    family = "hybrid"
    description = "Routes to best champion per regime bucket"

    ROUTING = {('ASIA', 'UPTREND', 'HIGH_VOL'): 'volume', ...}
    FALLBACK = 'volatility'
    def generate_signals(self, data):
        champ_sigs = {}
        for fam, strat in self._champion_strategies.items():
            champ_sigs[fam] = strat.generate_signals(data)
        # ... routing logic using ROUTING table ...
```

If a champion has zero sufficient_evidence 3D buckets -> use its global signal directly.

11. Scientist / Critic Loop

11.1 Loop Structure (per hybrid, independent)

Step	Action
1	Backtest -> trade log
2	DiagnosticsEngine -> 60-row bucket table
3	Fitness check -> if -999: UNVIABLE, stop
4	Critic (Opus, temp=0) -> structured diagnosis
5	If UNVIABLE -> discard. If CONTINUE -> Refiner
6	Refiner (Sonnet, temp=0) -> updated code (one change only)
7	Validation gate -> accept / revert / early-exit
8	Repeat max 5 iterations

11.2 Iteration Rules

Rule	Detail
Max iterations	5
Early exit	2 consecutive iterations with improvement < 0.05 Sharpe -> stop, keep best
Revert	If new score < previous score -> revert to previous version
Guarantee	Monotonic improvement: v_n >= v_{n-1} always
UNVIABLE fallback	If all 3 hybrids UNVIABLE -> fall back to best champion

11.3 UNVIABLE Conditions

Condition	Threshold
GLOBAL Sharpe < -5.0	Intentionally lenient — lets Scientist attempt recovery
Zero profitable 3D buckets	No 3D bucket with sufficient_evidence=True AND Sharpe > 0
Consecutive losses	GLOBAL max_consecutive_losses > 20

The -5.0 threshold is intentionally lenient. Strategies with Sharpe between -5.0 and 0 are poor but not hopeless — the Scientist loop can recover them. Only truly broken strategies (below -5.0) are eliminated immediately.

11.4 Critic Prompt (Claude Opus, temp=0)

```
You are the Evidence-Locked Critic for the Council of Alphas framework.

ROLE: Diagnose underperformance using ONLY computed numbers. Never guess.
Every claim must cite exact bucket and exact number from the table.

SCAN ORDER: GLOBAL -> 1D -> 2D -> 3D

CONSTRAINTS:
- No structural rewrites
- No changing indicators or family
- One surgical fix only (parameter, threshold, or single condition)
- Every claim must cite: [bucket] | sharpe=[x] | n=[x]

UNVIABLE if ANY of:
- GLOBAL sharpe < -5.0 (with sufficient_evidence=True)
- Zero 3D buckets with sufficient_evidence=True and sharpe > 0
- GLOBAL max_consecutive_losses > 20

OUTPUT FORMAT (exactly this, nothing else):
PRIMARY_FAILURE: [exact bucket] | sharpe=[value] | n=[value]
ROOT_CAUSE: [one sentence citing the code]
SURGICAL_FIX: [exact code change]
EXPECTED_IMPACT: [which metric improves and why]
VERDICT: CONTINUE | UNVIABLE
```

11.5 Refiner Prompt (Claude Sonnet, temp=0)

```
You are the Surgical Refiner. Apply exactly ONE fix from the Critic.
Do NOT restructure, change indicators, add or remove logic.
Return ONLY the complete updated Python class.
No explanation. No markdown. No imports.
```

12. Orchestration

12.1 Pipeline Flow

```
PIPELINE START
|
+-- 1. LOAD DATA (Binance SOL-USD 1h parquet)
|
+-- 2. BUILD / LOAD STATE MATRIX
|   +-- If parquet exists -> load
|   +-- If not -> StateMatrixBuilder.build() -> save parquet
|
+-- 3. SPECIATION (parallel, asyncio.gather)
|   +-- 4 specialists x 1-3 strategies each (scored immediately)
|
+-- 4. NICHE SELECTION (top 1 per family, score > 0)
|
+-- 5. HYBRID BUILDING (pure Python, 3 hybrids)
|
+-- 6. SCIENTIST LOOP (parallel per hybrid, sequential per iteration)
|   +-- Max 5 iterations each, early exit, revert, monotonic
|
+-- 7. FINAL RANKING (fitness score, fallback to best champion)
|
+-- OUTPUT -> Streamlit UI
```

12.2 Error Handling Tiers

Stage	Failure Type	Action
Data Load	File not found	Fatal — stop pipeline
State Matrix	Build error	Fatal — stop pipeline
Speciation	All strategies fail	Fatal — stop pipeline
Speciation	One specialist fails	Warn, continue with 3 families
Niche Selection	< 2 champions	Warn, continue
Hybrid Building	One template fails	Skip that hybrid
Scientist	All hybrids UNVIABLE	Fall back to best champion
Ranking	No survivors	Return best champion

12.3 Configuration Constants (core/config.py)

```
MAX_STRATEGIES_PER_SPECIALIST = 3
MAX_GENERATION_ATTEMPTS = 3
STRATEGY_TIMEOUT_SECONDS = 60
MAX_SCIENTIST_ITERATIONS = 5
MIN_IMPROVEMENT_THRESHOLD = 0.05

TBM_WIN = 2.0
TBM_LOSS = 1.0
TBM_TIME_HORIZON = 24          # 24 bars = 24 hours on 1h
TBM_ATR_WINDOW = 24
```

```
BACKTEST_FEE = 0.00040          # 0.04% MEXC taker fee
RISK_PER_TRADE = 0.005          # 0.5% of account equity
INITIAL_CAPITAL = 100000        # $100,000
MAX_LEVERAGE = 20

MIN_TRADES_SUFFICIENT_EVIDENCE = 30
MIN_TOTAL_TRADES_TRADABLE_BUCKETS = 300
TREND_SLOPE_THRESHOLD = 0.0005
UNVIABLE_GLOBAL_SHARPE = -5.0
UNVIABLE_MAX_CONSEC_LOSSES = 20
```

12.4 Parallelism

Component	Execution Model
Specialists (x4)	asyncio.gather — all 4 run concurrently
Scientist (x3 hybrids)	asyncio.gather — all 3 hybrids refined concurrently
Inside each hybrid loop	Sequential — each iteration depends on previous result
Single crash	return_exceptions=True — one failure never kills others

13. Output / UI (Streamlit — Andreas)

Panel	Content
1 - Pipeline Status	Live real-time log of every pipeline event
2 - Champion Leaderboard	Family, Name, Fitness, Sharpe, Win Rate, Trades, Coverage
3 - Diagnostics Heatmap	Plotly: rows=Session x Trend, cols=Vol, color=Sharpe
4 - Scientist Loop Trace	Per hybrid: iteration history, Critic diagnoses, fixes applied
5 - Final Ranked Results	Lineage tree + Cumulative PnL chart (survivors only)

Cumulative PnL: Plotly line chart, one line per surviving strategy, X=trade number, Y=cumulative return %. Fallback to best champion if all hybrids UNVIABLE.

14. Model Assignment Summary

Role	Model	Temperature
Specialist Agents (x4)	Claude Sonnet	0
Architect	PYTHON ONLY — no LLM	N/A
Scientist Critic	Claude Opus	0
Scientist Refiner	Claude Sonnet	0

15. Pre-Built Files (Do Not Rewrite Core Logic)

File	Status	Notes
core/state_builder.py	Needs param update	tbm_win=2.0, tbm_loss=1.0, atr_window=24, time_horizon=24
core/labeling.py	Ready	Dual-direction TBM with whipsaw NaN
core/backtesting.py	Ready	VectorizedBacktester, Numba-accelerated
core/diagnostics.py	Ready	60-row bucket table, sufficient_evidence
core/whitelist_indicators.py	Ready	Full indicator library, Indicators mixin
pipeline/indicator_sampler.py	Needs cleanup	Remove hidden indicators, sampling engine only
core/strategy_base.py	Needs update	Remove tbm params, add description field

16. Files to Build

File	Purpose
core/config.py	All constants in one place — build first
pipeline/prompt_builder.py	Specialist prompts from template + sampled indicators
pipeline/specialist_agent.py	LLM call + code validation + retry logic
pipeline/fitness.py	compute_fitness() + is_unviable()
pipeline/niche_selector.py	Champion selection per family
pipeline/hybrid_builder.py	HybridBuilder class (all 3 templates, pure Python)
agents/critic_agent.py	Opus Critic call + structured response parser
agents/refiner_agent.py	Sonnet Refiner call + code extraction
agents/scientist.py	Full Scientist loop orchestration
orchestrator.py	Main pipeline controller
app.py	Streamlit UI (Andreas)