Trading Intelligence

ϕ-Enhanced Financial Reasoning

ARKHEION AGI 2.0 — Paper 36

Jhonatan Vieira Feitosa
Independent Researcher
Manaus, Amazonas, Brazil

February 2026

Abstract

This paper presents **Trading Intelligence**, a financial analysis and portfolio optimization module for ARKHEION AGI 2.0. The system combines **technical analysis**, **ϕ-enhanced optimization**, and **risk management** to support investment decisions. This is a **research prototype**—not financial advice. The architecture demonstrates how AGI capabilities can be applied to financial domains while maintaining transparency and risk awareness.

Keywords: algorithmic trading, portfolio optimization, risk management, Fibonacci, financial AI

Disclaimer: This paper describes research software. No financial advice is provided. Past performance does not guarantee future results. Cryptocurrency and securities trading involves substantial risk of loss.

Epistemological Note

*This paper is **primarily heuristic**. Financial markets are complex adaptive systems where backtesting rarely predicts future performance:*

Heuristic	Status
“ϕ optimization”	Research exploration
“Fibonacci levels”	Technical analysis tool
“Portfolio optimization”	Markowitz framework

Warning: No backtested results are presented as predictive of future returns.

1 Introduction

Financial markets present unique challenges for AI:

- **Non-stationarity**: Patterns change over time
- **Reflexivity**: Predictions affect outcomes
- **Noise**: Signal-to-noise is low
- **Risk**: Substantial losses possible

ARKHEION’s Trading Intelligence explores how AGI capabilities can assist (not replace) human financial decision-making.

2 Technical Analysis

2.1 Fibonacci Levels

Sacred geometry ($\phi = 1.618$) appears in Fibonacci retracements:

Level	Ratio
0%	0.000
23.6%	$1 - 1/\phi^3$
38.2%	$1 - 1/\phi^2$
50%	0.500
61.8%	$1/\phi$
78.6%	$\sqrt{1/\phi}$
100%	1.000

2.2 Indicator Suite

```python
class TechnicalIndicators:
    def sma(self, prices, period=20):
        """Simple Moving Average."""
        return prices.rolling(period).mean()

    def ema(self, prices, period=20):
        """Exponential Moving Average."""
        return prices.ewm(span=period).mean()

    def rsi(self, prices, period=14):
        """Relative Strength Index."""
        delta = prices.diff()
        gain = delta.clip(lower=0).rolling(period).mean()
        loss = (-delta.clip(upper=0)).rolling(period).mean()
        return 100 - 100/(1 + gain/loss)

    def fibonacci_retracement(self, high, low):
        """Fibonacci levels."""
        diff = high - low
        return {
            ’0.0’: low,
            ’23.6’: low + 0.236 * diff,
            ’38.2’: low + 0.382 * diff,
            ’50.0’: low + 0.500 * diff,
            ’61.8’: low + 0.618 * diff,
            ’78.6’: low + 0.786 * diff,
            ’100.0’: high,
        }
```

3 Portfolio Optimization

3.1 Markowitz Framework

Mean-variance optimization:

$$\min_{w} \frac{1}{2} w^T \Sigma w - \lambda \mu^T w \tag{1}$$

subject to $\sum w_i = 1$ and $w_i \geq 0$.

3.2 ϕ-Enhanced Allocation

Golden ratio weighting for sector allocation:

```
def phi_allocation(self, assets: List[str]):
    """Allocate using golden ratio cascade."""
    n = len(assets)
    weights = []
    remaining = 1.0

    for i in range(n-1):
        w = remaining / PHI  # 61.8% of remaining
        weights.append(w)
        remaining -= w

    weights.append(remaining)
    return dict(zip(assets, weights))
```

Example for 4 assets: 38.2%, 23.6%, 14.6%, 23.6%.

4 Risk Management

4.1 Risk Metrics

Metric	Formula	Target
Sharpe Ratio	$(R_p - R_f)/\sigma_p$	> 1.0
Max Drawdown	Max peak-to-trough	$< 20\%$
VaR (95%)	5th percentile	$< 5\%$
Beta	$Cov(R_p, R_m)/Var(R_m)$	Market-relative

4.2 Position Sizing

Kelly criterion with fractional sizing:

$$f^* = \frac{p \cdot b - q}{b} \tag{2}$$

where p is win probability, $q = 1 - p$, b is win/loss ratio. **Safety**: Use $f^*/4$ for conservative sizing.

5 Safety Constraints

5.1 Hard Limits

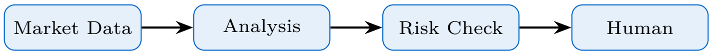
```
class RiskGuardrails:
    MAX_POSITION_SIZE = 0.10  # 10% max per asset
    MAX_SECTOR_EXPOSURE = 0.30  # 30% per sector
    MAX_DAILY_LOSS = 0.05  # 5% stop-loss
    MIN_CASH_RESERVE = 0.10  # 10% always cash

    def check_trade(self, trade, portfolio):
        if trade.size > self.MAX_POSITION_SIZE:
            raise RiskViolation("Position too large")

        if portfolio.daily_loss > self.MAX_DAILY_LOSS:
            raise RiskViolation("Daily loss exceeded")
```

5.2 Cooling-Off Periods

After significant losses:

- 3% daily loss → 1 hour pause
- 5% daily loss → Trading halted for day
- 10% weekly loss → Week-long review

6 System Architecture



Note: Human approval required for all trades.

7 Implementation

Component	Status
Module directory	`src/core/trading/`
Current state	Stub/framework
Data sources	(To be integrated)
Execution	(Manual only)

Current Status: The trading module is a minimal stub. Full implementation requires:

- Market data API integration
- Backtesting framework
- Paper trading validation
- Regulatory compliance review

8 Ethical Considerations

- **No manipulation**: System must not engage in market manipulation
- **Fair access**: Not designed for front-running or HFT
- **Transparency**: All logic explainable
- **Human oversight**: No autonomous trading

9 Conclusion

Trading Intelligence demonstrates how ARKHEION AGI capabilities can be applied to financial analysis. The system emphasizes **risk management**, **human oversight**, and **transparency** over aggressive automation.

Future work:

- Sentiment analysis integration
- Multi-asset correlation modeling
- Explainable trade rationale

References

1. Markowitz, H. “Portfolio Selection.” Journal of Finance, 1952.
2. Kelly, J.L. “A New Interpretation of Information Rate.” Bell System Technical Journal, 1956.