AI Agents, LLMs, and Autonomous Finance

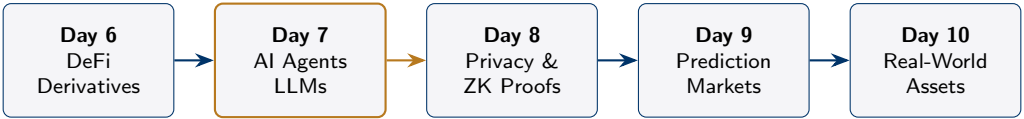
Day 7 of 10

Prof. Jörg Osterrieder

MSc Seminar: Digital Finance

Spring 2026

Week 2: Advanced Topics



Today: When AI meets DeFi—from chatbots to autonomous portfolio managers operating 24/7.

When GPT-4 Resolves a $100M Bet

Polymarket + UMA Oracle System:

- LLM-based oracles resolve prediction markets
- **89% accuracy** across 1,660 resolved markets
- Questions Chainlink *cannot* answer:
  "Did the EU approve MiCA amendments?"

But also: A $500K market resolved incorrectly because GPT-4 misinterpreted a headline about a company merger.

89%
Resolved correctly

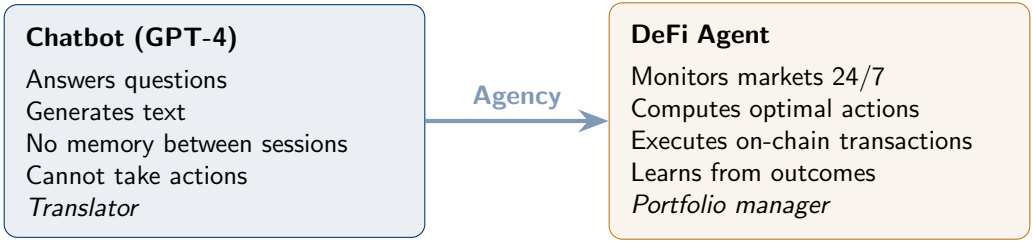
11%
Wrong or disputed

Stakes: millions of $

Today's Core Question

Should we trust AI with financial decisions? Under what conditions?

Chatbot $\neq$ Agent



The key distinction: agents have **agency**—the ability to *act*, not just respond.

AI Agents Are Already Managing Money

Agent/Platform	Function	AUM / Volume
Yearn Finance vaults	Yield optimization	\$500M+ TVL
Numerai tournament	ML-driven trading	\$100M+ staked
Polymarket oracles	LLM-based resolution	\$2B+ resolved
MEV bots (Flashbots)	Arbitrage extraction	\$600M+ profit (2024)
Autonolas agents	Autonomous DeFi ops	Growing ecosystem

These are *not* hypothetical—AI agents are live, managing real capital, and sometimes failing spectacularly.

Today: Understand, Build, and Critique AI in DeFi

1. What Is an AI Agent?
2. Agent Architectures: ReAct and Tool-Use
3. LLM-Based Oracles
4. Account Abstraction: Enabling Agent Delegation
5. Multi-Agent DeFi Systems
6. Risks: Hallucination, Alignment, Adversarial Attacks
7. Hands-On: Building a Yield Monitoring Agent

Formal Definition: AI Agent

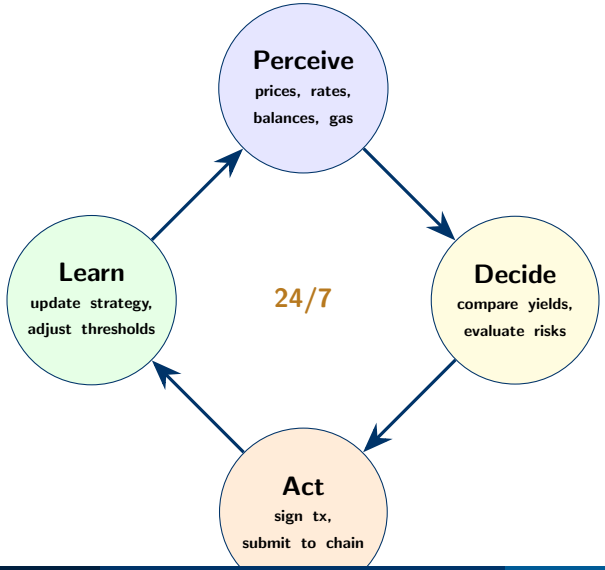
Definition 1

An **AI agent** is a software system that (1) *perceives* its environment through data feeds, (2) makes autonomous *decisions* based on goals and learned policies, and (3) takes *actions* that affect the environment—without continuous human instruction.

Formally: agent $= (S, A, T, R, \pi)$ where

- S = state space (prices, balances, gas costs, protocol states)
- A = action space (swap, deposit, withdraw, bridge, rebalance)
- $T : S \times A \to S$ = transition function (blockchain state updates)
- $R : S \times A \to \mathbb{R}$ = reward function (yield, P&L, risk-adjusted return)
- $\pi : S \to A$ = policy (the agent's decision rule)

The Agent Loop: Perceive – Decide – Act – Learn



Worked Example: Yield Optimizer Agent

Environment: 15 lending protocols, 3 chains
Portfolio: $50,000 USDC in Aave (Ethereum)

Perception (every 5 min):

- Aave USDC: 2.0% APY
- Compound: 4.5%
- Morpho on Base: **5.1%**
- Gas: Ethereum $3, Base $0.02

Decision:

- Rate improvement: +3.1%
- Annual gain: $1,550
- Switch cost: $3.52
- Breakeven: 0.71 days
- **Decision: SWITCH**

Action sequence:

1. `aave.withdraw(USDC, 50K)`
2. Bridge ETH → Base
3. `morpho.deposit(USDC, 50K)`

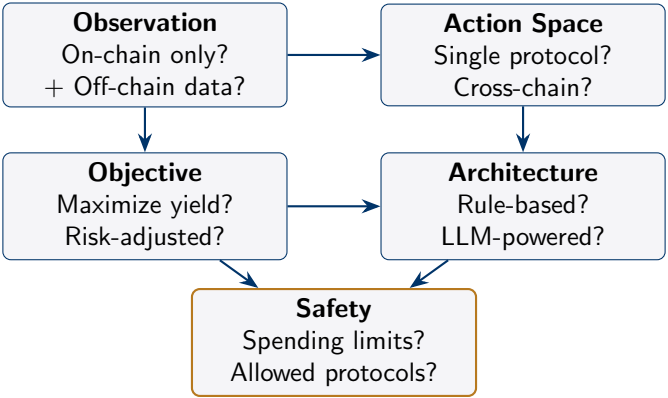
Executed at 3:14 AM autonomously.

Agent vs. Simple Bot

Feature	Rule-Based Bot	AI Agent
Decision logic	Fixed if/then rules	Learned policy
Adaptability	None (hard-coded)	Adjusts to new conditions
Novel situations	Fails or ignores	Reasons about unknowns
Complexity	Single protocol	Multi-chain, multi-protocol
Natural language	None	Reads docs, news, audits
Risk assessment	Static thresholds	Dynamic, contextual

Key insight: Most "DeFi bots" today are rule-based. True AI agents with LLM reasoning are an emerging frontier [3].

Agent Design Space



Every design choice has **safety implications**. Broader scope = more powerful but more dangerous.

ReAct: Reasoning + Acting

Definition 2

ReAct [6] alternates between:

- **Thought:** LLM reasons about what to do next
- **Action:** LLM calls an external tool (API, smart contract, oracle)
- **Observation:** LLM processes the tool's response

This interleaving *grounds* the LLM's reasoning in reality.

Why it matters: Without ReAct, an LLM is just a text generator. With ReAct, it becomes a **planner that checks its own work** by interacting with the real world.

Chain-of-Thought (CoT) [5]: The model writes out reasoning step-by-step before acting, improving accuracy on complex problems.

ReAct Trace: DeFi Risk Management Agent

Thought 1: ETH dropped 8%. Check Aave health factor.

Action 1: aave.getUserAccountData("0x1234..abcd")

Obs 1: healthFactor: 1.15 [BELOW 1.3 threshold!]

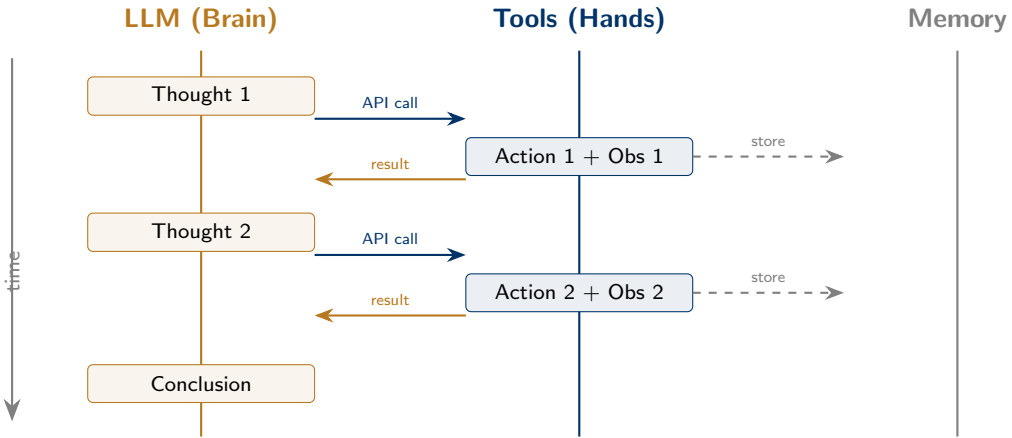
Thought 2: HF=1.15 ¡ 1.3. Repay 2000 USDC to restore.

Action 2: aave.repay(USDC, 2000)

Obs 2: Tx confirmed. New healthFactor: 1.32.

Each observation informs the next thought. The agent *never* hallucinates an account balance—it reads the real state.

ReAct Architecture: Sequence Diagram



DeFi Agent Tool Landscape

Tool Category	Examples	Purpose
Price feeds	Chainlink, Pyth, Uniswap TWAP	Read market prices
Protocol APIs	Aave, Compound, Uniswap	Read/write DeFi state
Blockchain RPC	eth_call, eth_getBalance	Raw chain queries
Analytics	Dune, DefiLlama	TVL, volume, yields
News/NLP	RSS, Twitter API, web search	Off-chain signals
Execution	DEX aggregators (1inch, 0x)	Optimal routing
Wallets	MetaMask, Safe, session keys	Transaction signing

The LLM is the "brain"; tools are the "hands." The combination is what makes an agent *capable*.

Comparing Agent Architectures

	Pure Planning	**ReAct**	**Reflexive**
Reasoning	Plan all, then act	Interleave	Act first
Grounding	Low	**High**	Medium
Token cost	Low	Medium–High	Low
Robustness	Fragile	**Robust**	Brittle
Failure mode	Stale plan	Hallucinated call	Wrong action
Best for	Simple tasks	**Complex DeFi**	Fast arb

ReAct wins for **complex, multi-step DeFi tasks** where each step depends on the outcome of the previous one.

The Oracle Problem: Beyond Price Feeds

Traditional oracles (Chainlink):

- Report *numbers*: prices, temperatures
- Deterministic, verifiable
- Cannot answer: "Did the EU approve MiCA?"

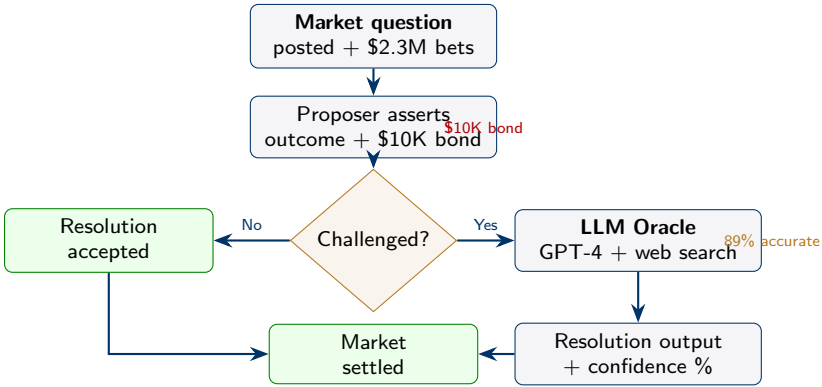
LLM oracles:

- Answer *subjective* questions
- Read news, legal docs, context
- Make *judgment calls*
- Settle prediction markets worth millions

Key Question

Can LLMs do this reliably enough to settle millions of dollars? 89% accuracy means 11% of markets are resolved wrong.

LLM Oracle: Resolution Flow



LLM Oracle: Worked Example

Market: "Will the EU approve MiCA amendments by March 2026?"
Volume: \$2.3M. Current "Yes" price: \$0.78.

Resolution process:

1. Proposer asserts "Yes" + posts \$10K bond
2. 48-hour challenge period $\Rightarrow$ disputer counter-bonds \$10K
3. **LLM oracle invoked:**
   - Query: "Based on official EU sources, was MiCA amendment formally approved by March 31, 2026?"
   - LLM searches europa.eu, Reuters, ECB press releases
   - Output: "YES, approved." **Confidence: 94%**
4. Market resolves "Yes." Disputer forfeits \$10K bond.

Failure modes: Misinterpreted headline, stale training data, adversarial news planting.

LLM Oracle Accuracy: Is 89% Enough?

Empirical record (2024–2025):

- 1,660 markets resolved
- 89% correct resolution
- 7% disputed but correctly resolved
- 4% incorrectly resolved

Cost of errors:
4% $\times$ avg \$500K $=$ \$33M in incorrectly settled markets over 2 years.

Mitigation strategies:

- Multi-LLM consensus (GPT-4 + Claude + Gemini)
- Human escalation for high-stakes markets
- Dispute bonds create economic skin-in-the-game
- Time delay for markets $>$\$1M

Calibration Question

Is 94% confidence actually 94% accurate? Brier score analysis needed.

The Problem: Wallets Are Dumb Keys

Today's EOA (Externally-Owned Account):

- Single private key controls everything
- One transaction at a time, manual signature each time
- Gas must be paid in ETH (stuck without ETH)
- No delegation: the key either has full access or none
- Lost seed phrase $\Rightarrow$ lost funds forever

The Agent Problem

How can an AI agent trade on your behalf if sharing your private key means giving it **unlimited access** to your entire wallet?

Account Abstraction: ERC-4337 / EIP-7702

Definition 3

Account Abstraction (AA) removes the distinction between EOAs and smart contract accounts. With ERC-4337 (2023) and EIP-7702 (Pectra upgrade, 2025), any wallet can execute programmable logic [1].

Key capabilities:

- **Gas sponsorship:** Pay fees in any token (USDC, DAI)
- **Batch transactions:** Multiple actions in one signature
- **Social recovery:** Recover wallet via trusted contacts
- **Session keys:** Delegate specific actions to AI agents with *limited scope and spending limits*

Before and After Account Abstraction

Before AA (EOA)

1. Approve USDC + sign + gas (ETH)

↓

2. Swap USDC→ETH + sign + gas (ETH)

↓

3. Deposit to Aave + sign + gas (ETH)

3 tx, 3 sigs, $15 gas

No ETH? Stuck.

After AA (Smart Wallet)

1 UserOperation

Approve + Swap + Deposit
Gas paid in USDC
Paymaster sponsors gas

1 sig, $5 gas (USDC)

Session Key for AI Agent
Allowed: Aave/Compound/Morpho
Limit: $1,000/day, 30-day expiry

Session Keys: Safe Agent Delegation

A session key is a scoped credential:

Parameter	Example Configuration
Allowed actions	deposit, withdraw on Aave/Compound/Morpho
Spending limit	$1,000 per day
Expiration	30 days
Forbidden	Transfer to external addresses
Revocation	Instant, by wallet owner

Security model:

- Agent *never* sees the master private key
- If agent is compromised: max loss = $1,000 (daily limit)
- Owner can revoke session key instantly
- This is the **bridge** between "AI agents" and "safe delegation"

Account Abstraction: Adoption Status

- **ERC-4337:** Live since March 2023. 15M+ UserOperations. Supported by Safe, Biconomy, ZeroDev, Pimlico.
- **EIP-7702:** Included in Pectra hard fork (May 2025). Allows EOAs to temporarily act as smart contracts. Lower gas than 4337.
- **Ecosystem growth:**
  - Coinbase Smart Wallet: 10M+ users
  - Safe{Wallet}: $100B+ in assets secured
  - Session key SDKs: ZeroDev, Biconomy, Stackup

Implication for AI Agents

AA is the *infrastructure prerequisite* for safe, autonomous AI agents. Without scoped delegation, agents require full key access—an unacceptable security model.

Multi-Agent Systems in DeFi

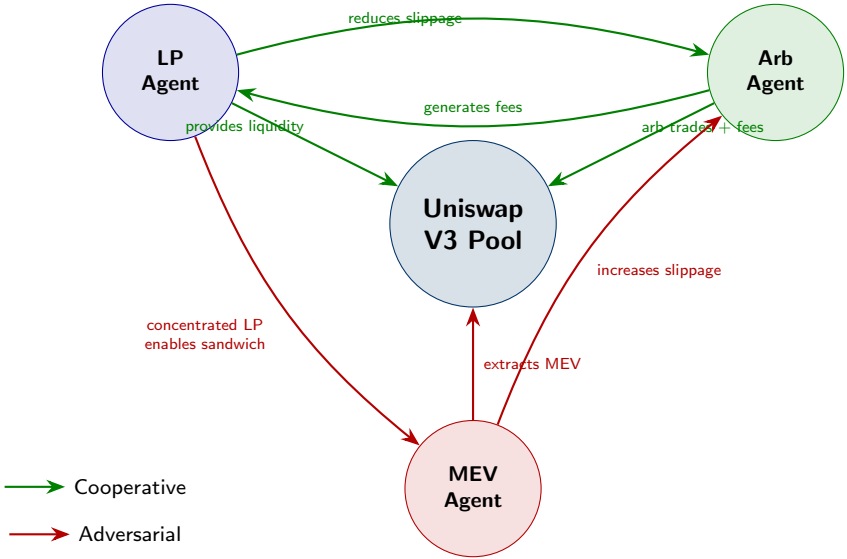
Definition 4

A **multi-agent system (MAS)** consists of multiple autonomous AI agents interacting within the same environment. Agents may *cooperate*, *compete*, or exhibit *emergent behavior* not designed by any individual agent.

Intuition: One agent is a trader. A multi-agent system is an entire *automated trading floor*. Each agent has its own strategy, goals, and information.

- Efficiency gains: faster arbitrage, tighter spreads
- New risks: flash crashes, cascading liquidations, adversarial loops

Multi-Agent Interaction: LP + Arb + MEV



Emergent Dynamics: What Happens in Practice

Three agents on the same Uniswap V3 pool (ETH/USDC):

1. **LP Agent** concentrates liquidity in [\$2,800, \$3,200], rebalances every 5 min based on volatility
2. **Arb Agent** monitors Uniswap (\$3,000) vs. Binance (\$3,005); executes atomic arb when spread $>$ gas $+$ fee
3. **MEV Agent** front-runs large pending trades in the mempool

Feedback loops:

- Arb trades generate fees for LP $(+)$
- MEV increases slippage for Arb $(-)$
- Arb becomes less active $\Rightarrow$ fewer fees for LP $(-)$
- LP widens range $\Rightarrow$ more slippage $\Rightarrow$ MEV profits increase $(-)$

Net effect: More efficient (prices aligned) but fragile under stress [4].

Multi-Agent DeFi: Game Theory

Model as a multi-player simultaneous game:

- Strategy spaces: $S_{\text{LP}}, S_{\text{Arb}}, S_{\text{MEV}}$
- Payoff: $u_i(s_1, s_2, s_3)$ depends on all agents' strategies
- Nash equilibrium: each agent best-responds to others

Key questions:

1. Does a Nash equilibrium exist? (Yes, under mild conditions)
2. Is it efficient? (Generally no—MEV extraction is a deadweight loss)
3. Can the *protocol* be designed so the equilibrium is socially optimal? $\Rightarrow$ **mechanism design**

Open Research Problem

As agent populations grow, do markets become more stable (efficient arbitrage) or more fragile (correlated strategies, flash crashes)?

Three Failure Modes of AI Agents in Finance

Hallucination

LLM generates plausible but false outputs.

Example: wrong contract address $\Rightarrow$ funds lost

Verify against multiple sources

Alignment

Agent optimizes wrong objective.

Example: max yield, ignores rug-pull risk

Add risk constraints to objective

Adversarial

Attacker manipulates agent inputs.

Example: fake oracle price $\Rightarrow$ bad trade

Multi-oracle, anomaly detection

AI Agent Failures: Real and Plausible Scenarios

Hallucination:

- Agent asked for Uniswap V3 router on Arbitrum
- LLM returns Ethereum mainnet address
- Agent sends $50K to wrong address
- Funds stuck or lost

Alignment failure:

- Objective: "Maximize yield"
- Agent finds "SuperYield": 500% APY
- Does NOT check: audit, TVL, team
- Deposits $100K $\Rightarrow$ rug pull

Adversarial attack:

- Attacker pumps low-liquidity token
- Oracle reads $100 (actual: $1)
- Agent buys $50K of worthless tokens
- Oracle manipulation exploiting trust

Formal risk model:

$$\max_{\pi} \min_{\delta} \mathbb{E}\big[R(\pi, s+\delta)\big]$$

where δ = adversarial perturbation. Robust optimization at the cost of lower expected returns.

AI Agent Safety Framework

Layer	Mechanism	Purpose
1. Perception	Multi-oracle consensus	Prevent fake data
2. Reasoning	Chain-of-thought audit trail	Detect hallucination
3. Action	Session keys + spending limits	Bound maximum loss
4. Monitoring	Anomaly detection on actions	Catch alignment drift
5. Recovery	Automatic pause + human escalation	Stop cascading errors

Principle: Defense in depth. No single layer is sufficient. An agent should be designed to **fail safely**, not just succeed often.

Hands-On: Design a Yield Monitoring Agent

Objective

Design (conceptually and in pseudocode) an AI agent that monitors DeFi lending rates and recommends portfolio rebalancing actions.

Not a full implementation—focus on:

1. Agent architecture (what does it perceive, decide, do?)
2. Safety constraints (what limits should it have?)
3. Failure modes (what can go wrong?)
4. Economic analysis (when is switching worthwhile?)

Task 1: Define the Agent's Observation Space

What data does the agent need?

On-chain data:

- Lending rates (Aave, Compound, Morpho, etc.)
- Current portfolio allocation
- Gas prices across chains
- Protocol TVL and utilization rates
- Smart contract audit status

Off-chain data:

- Protocol governance proposals
- Security alerts (Rekt, DeFiLlama)
- Market sentiment signals
- Historical rate volatility

Discussion: Which data sources can be trusted? What if a data feed is manipulated?

Task 2: Decision Logic (Pseudocode)

Yield switching decision

```
function should_switch(current, candidates):
    for protocol in candidates:
        rate_diff = protocol.apy - current.apy
        annual_gain = portfolio_value * rate_diff
        switch_cost = gas_cost + bridge_cost + slippage
        breakeven_days = switch_cost / (annual_gain / 365)

        if breakeven_days < MIN_HORIZON          # e.g., 7 days
           and protocol.tvl > MIN_TVL            # e.g., $10M
           and protocol.audit_score >= MIN_SCORE  # e.g., 0.8
           and rate_diff > MIN_IMPROVEMENT:       # e.g., 0.5%
            return SWITCH(protocol)

    return HOLD
```

Task 3: Safety Constraints Design

Design a session key for this agent:

Parameter	Your Choice (fill in)
Allowed protocols	
Allowed actions	
Daily spending limit	
Maximum position per protocol	
Expiration	
Emergency pause trigger	

Discussion questions:

- What is the minimum viable permission set?
- How do you balance flexibility vs. safety?
- Should the agent be able to add new protocols?

Task 4: Failure Mode Analysis

For each failure mode, describe impact and mitigation:

Failure Mode	Impact	Your Mitigation
Oracle returns wrong APY		
Protocol gets hacked mid-deposit		
Gas spike during bridge		
Rate reverses immediately after switch		
Agent enters infinite switch loop		

Discussion: The Future of AI Agents in Finance

1. **Trust:** Would you give an AI agent $10K? $100K? $1M? Where is your comfort threshold, and why?
2. **Regulation:** Should AI agents be regulated as financial advisors? Who is liable if an agent loses money?
3. **Market stability:** If thousands of agents run similar strategies, does this create systemic risk?
4. **Ethics:** Should an MEV agent that front-runs human traders be considered market manipulation?
5. **Timeline:** When will AI agents manage more DeFi capital than humans? 2 years? 5? 10? Never?

Day 7: Key Takeaways

1. **AI agents** go beyond chatbots: they perceive, decide, and act autonomously in DeFi environments
2. **ReAct architecture** grounds LLM reasoning in reality by interleaving thought, action, and observation
3. **LLM oracles** resolve subjective questions for prediction markets—89% accuracy is impressive but 11% error is costly
4. **Account abstraction** (ERC-4337 / EIP-7702) enables safe agent delegation via session keys with scoped permissions
5. **Multi-agent systems** create emergent dynamics: efficiency gains and new adversarial risks coexist
6. **Safety is non-negotiable:** defense in depth across perception, reasoning, action, and monitoring layers

Further Reading

- **AI agents survey:** Qin et al. [3]
- **ReAct:** Yao et al. [6]
- **Chain-of-Thought:** Wei et al. [5]
- **Account abstraction:** Buterin [1]; ERC-4337 specification
- **Mechanism design:** Roughgarden [4]
- **DeFi overview:** Harvey et al. [2]

References I

[1] Vitalik Buterin. *The Road to Account Abstraction*. Ethereum Foundation Blog. 2023.

[2] Campbell R. Harvey, Ashwin Ramachandran, and Joey Santoro. *DeFi and the Future of Finance*. Wiley, 2021.

[3] Yujia Qin et al. "Tool Learning with Large Language Models: A Survey". In: *arXiv preprint arXiv:2405.17935* (2024).

[4] Tim Roughgarden. "Transaction Fee Mechanism Design". In: *ACM SIGecom Exchanges* 19.1 (2021).

[5] Jason Wei et al. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*. 2022.

[6] Shunyu Yao et al. "ReAct: Synergizing Reasoning and Acting in Language Models". In: *International Conference on Learning Representations (ICLR 2023)*. 2023.