**AI Research Engineer (agentic reasoning & tool use)**

**The assignment has been done with complete dedication utilizing X number of hours following all the instruction from assignment** [link](https://shopix.notion.site/assignment-8-ai-research-engineer-agentic-reasoning-tool-use-7d6999daa0e24e68b83987b598c35784)**.  
  
Research papers engaged with:**

1. [ReAct: Synergizing Reasoning and Acting in Language Models](https://paperswithcode.com/paper/react-synergizing-reasoning-and-acting-in)
2. [Toolformer: Language Models Can Teach Themselves to Use Tools](https://paperswithcode.com/paper/toolformer-language-models-can-teach)
3. [ReST meets ReAct: Self-Improvement for Multi-Step Reasoning LLM Agent](https://paperswithcode.com/paper/rest-meets-react-self-improvement-for-multi)
4. [Chain of Tools: Large Language Model is an Automatic Multi-tool Learner](https://paperswithcode.com/paper/chain-of-tools-large-language-model-is-an)
5. [Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models](https://paperswithcode.com/paper/language-agent-tree-search-unifies-reasoning)
6. **Non-Coding Component**

**A. Conceptual Map**

1. How LLMs reason and use external tools?

**Reasoning in LLMs**

LLMs reason by breaking down complex tasks into logical steps, using contextual information, and iteratively refining their responses. The reasoning process can be reactive (responding step-by-step to new information) or deliberative (planning a sequence of steps before execution).

LLMs use different reasoning strategies, including:

* **Chain-of-Thought (CoT) Reasoning** – LLMs explicitly generate intermediate steps before reaching a conclusion, improving logical consistency. (*Used in COT, ReAct, LATS*)
* **Search-Based Reasoning** – Instead of relying solely on internal knowledge, LLMs explore multiple paths using methods like **Monte Carlo Tree Search (MCTS)**. (*Used in LATS*)
* **Self-Reflection and Correction** – LLMs evaluate past actions, detect errors, and refine their approach iteratively. (*Used in ReST, LATS*)
* **Hybrid Reasoning (Reason + Act)** – LLMs combine **verbal reasoning** (thought generation) with **external tool interactions** for more informed decisions. (*Used in ReAct, ReST*)

**Using External Tools**

LLMs utilize tools through structured workflows that integrate reasoning, acting, and planning.  
This follows a structured workflow:

* **Tool Selection** – The LLM decides which tool to use based on the task.
* **Invocation** – It formulates a query, passing relevant arguments to the tool.
* **Execution** – The tool processes the request and returns a response.
* **Observation & Integration** – The LLM analyzes the tool’s output and incorporates it into reasoning.
* **Iteration & Correction** – If necessary, the LLM refines the process and calls additional tools.

Some models **autonomously learn** tool usage by **probing APIs**, discovering new tools, and refining their usage over time. (*Used in Toolformer, COT*)

2. Highlighting core workflows and interactions

1. Chain of Tools (COT)

* Workflow:  
  - Introduces Automatic Tool Chain (ATC) to allow LLMs to use multiple tools sequentially.  
  - Uses black-box probing to enable LLMs to discover, document, and learn new tools autonomously.  
  - Proposes ToolFlow, a benchmark for evaluating LLMs in long-term planning with complex tool dependencies.
* Interactions:  
  - LLMs interact with tools using programmatic tool chaining, where execution is structured as a sequence of dependent steps.  
  - Uses reflection mechanisms to debug and refine tool interactions dynamically.  
  - Moves beyond manually designed workflows by enabling self-learning of tool usage.

2. Language Agent Tree Search (LATS)

* Workflow:  
  - Introduces Monte Carlo Tree Search (MCTS) to explore different reasoning and acting paths before selecting the best action.  
  - Expands ReAct (Reason + Act) into a search-based decision-making framework.  
  - Uses an environment with external feedback to refine actions iteratively.
* Interactions:  
  - LLMs interact with external tools through search-based planning, instead of executing actions sequentially.  
  - Introduces LM-powered value functions to score and select the best reasoning paths.  
  - Uses self-reflection to optimize decision-making dynamically.

3. ReAct (Reason + Act)

* Workflow:  
  - Interleaves reasoning (thought generation) and acting (tool use) to improve decision-making.  
  - Enhances decision-making through tool interactions, allowing LLMs to gather new information before making final decisions.  
  - Uses step-by-step execution, where each action is based on the outcome of the previous step.
* Interactions:  
  - LLMs interact with external APIs (e.g., search engines) by reasoning about their needs first, then acting accordingly.  
  - Avoids the pitfalls of static chain-of-thought (CoT) reasoning by allowing dynamic, real-time interactions with tools.  
  - Increases interpretability and trustworthiness by making LLMs' decision-making explicit.

4. ReST (ReAct + Self-Training)

* Workflow:  
  - Extends ReAct by adding self-improvement through Reinforced Self-Training (ReST).  
  - Uses AI feedback and synthetic data to refine multi-step reasoning agents.  
  - Iteratively improves model performance by learning from past tool interactions and refining future steps.
* Interactions:  
  - Enhances tool interactions by integrating self-reflection to correct errors and improve reasoning quality.  
  - Uses automated feedback mechanisms to detect mistakes and refine responses without human intervention.  
  - Introduces self-distillation, where a smaller model is fine-tuned using lessons from a larger model's tool interactions.

5. Toolformer

* Workflow:  
  - Enables self-supervised tool learning, where LLMs independently learn when and how to call external APIs.  
  - Uses in-context learning to annotate training data with API calls, making tool interactions more autonomous.  
  - Fine-tunes LLMs on self-generated tool annotations to enhance their reasoning abilities.
* Interactions:  
  - LLMs interact with external tools by autonomously deciding which tool to use, when to use it, and how to process its response.  
  - Reduces dependency on human-annotated tool usage examples by using self-generated tool interaction data.  
  - Ensures that tool use is seamlessly integrated into the language model's natural workflow, without disrupting core language modeling abilities.

3. Conceptual Connections Across Papers

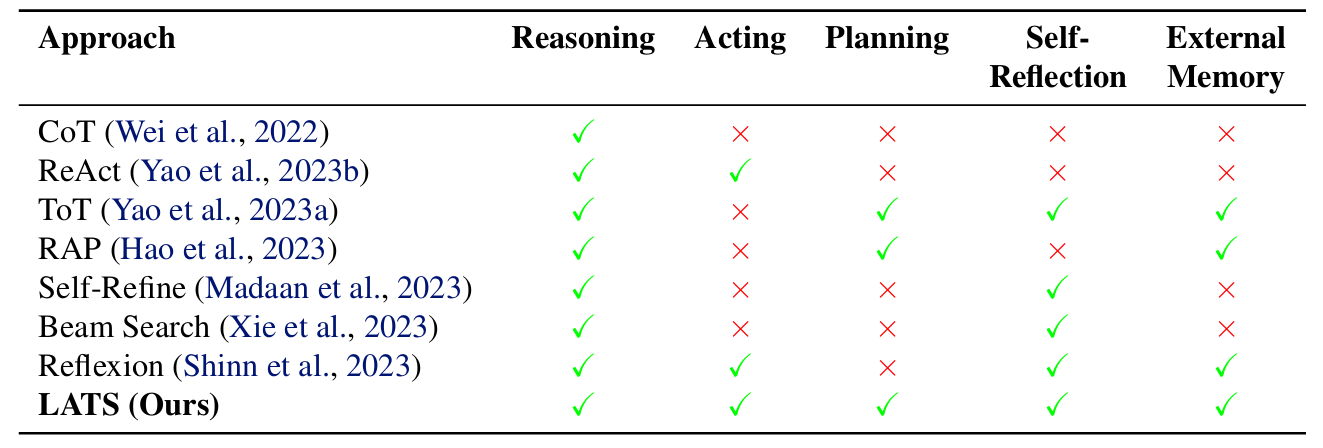
****

Figure : Summary of related work on reasoning, acting, and planning. (Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models Table 1)

Plan-Execute-Observe is a Core Theme: All papers use some form of stepwise decision-making, whether via tool chaining (COT), search-based planning (LATS), or interleaved reasoning and acting (ReAct, ReST).

Self-Reflection and Feedback Loops Improve Performance

* LATS and ReST integrate self-reflection, allowing models to adjust their actions based on external feedback.
* COT and Toolformer take this further by enabling LLMs to discover and refine tool usage on their own.

Search-Based vs. Execution-Based Decision Making

* LATS uses Monte Carlo Tree Search (MCTS) to explore multiple reasoning paths before acting.
* COT, ReAct, and ReST follow a sequential execution strategy, where decisions are refined based on stepwise observations.
* Toolformer enables LLMs to self-learn tool use without explicit stepwise execution logic.

Automated Tool Learning is a Future Trend

* Toolformer and COT are pioneering LLM-driven discovery and learning of tool usage, reducing reliance on predefined workflows.
* LATS and ReST enhance tool interactions by integrating feedback loops, but they do not yet enable completely self-directed tool learning.

**B. Analysis**

1. Agent design, reasoning steps, and tool use comparison

**a) Agent Design**

* COT and Toolformer focus on autonomous tool execution, with COT using a structured tool chaining approach and Toolformer allowing LLMs to self-learn tool usage without human supervision.
* LATS designs an agent around search-based decision-making, where multiple reasoning paths are explored before selecting an action.
* ReAct and ReST emphasize interactive and self-improving agents, where reasoning and tool use are dynamically adjusted in real-time. ReST further enhances this with reinforced self-training, allowing models to learn from past interactions.

**b) Reasoning Steps**

* COT follows a programmatic execution model, reasoning about tool dependencies in a structured workflow.
* LATS applies Monte Carlo Tree Search (MCTS) to evaluate multiple paths before deciding on an optimal reasoning sequence.
* ReAct alternates between verbal reasoning and tool actions, refining responses in real time.
* ReST extends ReAct by introducing self-reflection and iterative learning, ensuring the reasoning process improves over time.
* Toolformer embeds reasoning within the language modeling process, selecting tools based on how they improve token prediction.

**c) Tool Use**

* COT and Toolformer prioritize autonomous tool discovery and execution, enabling self-learning of tool usage.
* LATS selects tools strategically based on search-based exploration, ensuring optimized tool interactions.
* ReAct integrates tools within the reasoning loop, making tool use part of the thought process.
* ReST enhances this by allowing the model to reflect on past tool interactions and adjust future decisions.

**COT & Toolformer** → Best for automated, self-learning tool execution.

**LATS** → Best for complex decision-making requiring strategic tool use.

**ReAct & ReST** → Best for real-time interactive applications that involve reasoning-driven tool use.

2. Summary, Methodology contrast and real-world applicability

**Chain of Tools (COT)**  
COT equips LLMs to autonomously chain multiple external tools through a programmatic plan‐execute‐observe pipeline. It uses black-box probing to discover, document, and learn tool interfaces, making it effective for long-term planning tasks that involve complex, interdependent steps.

**Language Agent Tree Search (LATS)**  
LATS centres on search-based decision-making by integrating Monte Carlo Tree Search (MCTS) with language model reasoning. The agent explores multiple reasoning paths, leveraging LM-powered value functions and external feedback to select the most promising action. This strategy is geared toward environments where strategic planning and evaluating alternative pathways are critical.

**ReAct (Reason + Act)**  
ReAct introduces an interleaved framework where the LLM alternates between verbal reasoning (generating “thoughts”) and taking actions (tool use). This dynamic integration allows the agent to adjust in real time—gathering new external information via API calls and refining its internal thought process on the fly, much like human problem-solving.

**ReST (ReAct + Self-Training)**  
Building on ReAct, ReST incorporates self-improvement through Reinforced Self-Training. The agent uses AI feedback and synthetic data to iteratively refine its multi-step reasoning and tool usage. This approach emphasizes continuous learning and error correction, leading to progressively enhanced performance even in complex interactive settings.

**Toolformer**  
Toolformer enables LLMs to learn when and how to call external APIs using a self-supervised learning paradigm. By annotating its own training data with API calls and integrating these into the language modeling process, the model improves its ability to handle tasks such as factual lookups, arithmetic, and translations without requiring extensive human-labeled examples.

Contrasting Methodologies

**Autonomy vs. Strategic Planning:**

* *COT and Toolformer* prioritize **autonomous tool execution**. COT relies on explicit tool chaining and reflective debugging, while Toolformer uses self-supervised annotations to seamlessly embed tool usage within the language model’s predictions.
* In contrast, *LATS* emphasizes **strategic planning** through search-based methods (MCTS), evaluating multiple reasoning trajectories before committing to an action.

**Interactive Real-Time Adaptation:**

* *ReAct* and *ReST* focus on real-time, interactive decision-making by interleaving reasoning with action. ReAct provides a reactive framework, whereas ReST adds an iterative self-training loop to continually refine decisions based on past interactions.

**Learning Mechanisms:**

* *Toolformer* stands out by employing a **self-supervised learning** mechanism that enables the model to learn tool use from its own language modeling loss, requiring minimal human intervention.
* *ReST* leverages **reinforced self-training** to improve over time, showcasing a blend of immediate interaction (as in ReAct) and long-term learning.

Real-World Applicability

* **COT and Toolformer**:  
  These methods are best suited for automated, structured workflows where tasks involve predefined or discoverable tool interactions. Examples include business process automation and systems that require reliable multi-step operations with external tool integration.
* **LATS**:  
  With its focus on search-based, strategic planning, LATS is ideal for applications in complex decision-making environments—such as autonomous agents in simulations, robotics, or any scenario where exploring multiple potential action paths is crucial.
* **ReAct** and **ReST**:  
  These approaches excel in real-time interactive scenarios like AI assistants, tutoring systems, or customer service bots, where the ability to dynamically reason, act, and adapt on the fly is essential. ReST, with its iterative improvement loop, is particularly valuable in settings where continuous performance enhancement is desired.

**C. Open Questions**

Deploying LLM-based agents capable of reasoning and tool use presents exciting opportunities but also introduces significant challenges. As these models evolve from simple text predictors to autonomous decision-makers, their ability to scale efficiently, adapt to new tasks, handle errors, and integrate seamlessly with external systems becomes critical. While approaches like COT and Toolformer focus on autonomous tool execution, LATS emphasizes strategic planning, and ReAct and ReST prioritize real-time adaptability and self-improvement, each method faces unique hurdles in real-world deployment. Addressing these challenges requires advancements in efficiency, robustness, and interoperability, paving the way for more reliable and intelligent AI agents.

**A. Scalability**

**Computational Overhead**: Approaches like LATS (MCTS-based search) require extensive exploration, making them computationally expensive. ReST, with its iterative learning, also increases resource usage over time.

**Real-Time Processing**: Methods like ReAct and ReST, which interleave reasoning with actions, may struggle with high-frequency decision-making in real-time applications.

**Potential Improvement**:

* Implement hybrid models that combine search-based planning with efficient heuristics to reduce computational complexity.
* Use selective tool invocation, prioritizing critical tool calls to optimize efficiency.

**B. Adaptability**

Handling Novel Scenarios: COT and Toolformer enable LLMs to autonomously learn tool use, but they may fail in ambiguous or dynamic environments where tools evolve frequently.

**Generalization vs. Specialization**: Models trained in one domain may struggle to adapt to new tool interfaces or unseen reasoning tasks without fine-tuning.

**Potential Improvement**:

* Introduce meta-learning techniques, where models learn general strategies for tool use and decision-making across multiple domains.
* Develop continual learning architectures, allowing LLMs to retain past knowledge and dynamically integrate new tools.

**C. Error Handling and Reliability**

**Error Propagation**: COT’s multi-step tool chains can accumulate errors, leading to incorrect outputs. Similarly, ReST’s self-training may reinforce flawed reasoning patterns.

**Tool Failures and Unexpected Outputs**: LLMs may struggle to interpret tool failures, such as incorrect API responses or incomplete data.

**Potential Improvement**:

* Incorporate robust error detection mechanisms, such as confidence scoring or ensemble reasoning, to identify and correct errors before finalizing outputs.
* Develop fallback strategies, where the model can intelligently retry, reformulate, or switch tools when failures occur.

**D. Integration with External Systems**

**Standardized APIs and Protocols**: Toolformer and COT emphasize autonomous tool discovery, but real-world applications require structured APIs with well-defined schemas.

**Security and Data Privacy**: ReAct and ReST, which interact dynamically with external tools, pose risks if they process sensitive or unverified data.

**Potential Improvement**:

* Define universal tool-use protocols, where LLMs can interact with APIs in a structured, predictable manner.
* Implement security layers, including access control mechanisms and sandboxed environments to prevent unauthorized tool interactions.

**Future Research Directions**

**A. Efficient and Adaptive Planning Models**

* Explore hybrid planning architectures that combine search-based (LATS) and reactive decision-making (ReAct, ReST) to balance efficiency and depth in reasoning.
* Develop lightweight reasoning frameworks that optimize planning while maintaining adaptability.

**B. Self-Improving and Continual Learning Agents**

* Extend ReST’s self-improvement mechanism to dynamically fine-tune models over time without requiring manual retraining.
* Investigate memory-efficient architectures that allow LLMs to retain knowledge of past tool interactions.

**C. Error-Aware and Explainable AI for Tool Use**

* Develop error attribution techniques, allowing models to identify which step in a reasoning or tool-use process failed.
* Introduce transparent reasoning logs, where LLMs explain why they chose specific tools and how they arrived at decisions.

**D. Standardized Tool-Use Frameworks**

* Establish a benchmarking suite (expanding on ToolFlow) to evaluate LLM tool-use performance across different domains.
* Design cross-platform LLM APIs, enabling seamless integration with real-world applications.