### **Technical Research Report: A Unified Perception-Action-Reasoning Framework**

**Version:** 14.1 (Revised) **Date:** 2025-06-14

#### **Abstract**

This document establishes a new theoretical foundation for our agent architecture, adopting the general **Perception-Reasoning-Action (PRA)** model as its core principle. This unified framework describes all agentic behavior as a sequential decision process, which we formalize as an **LLM-policy agent** defined by a state space, action space, transition function, and a novel LLM-based policy function, π. This report then details our primary architectural implementation of the PRA framework: a multi-stage cognitive workflow named **"Plan, Synthesize, Review" (PSR)**. The PSR model is specifically engineered to apply the PRA loop to the complex task of generating high-fidelity, structured artifacts (e.g., reports, specifications, code). By formalizing our design under this unified theory, we create a more robust, principled, and extensible foundation for future development.

#### **1. The Unified Perception-Reasoning-Action (PRA) Framework**

The behavior of any advanced LLM-based agent can be abstracted into a powerful and continuous loop consisting of three phases: Perception, Reasoning, and Action. This cycle allows the agent to intelligently interact with its environment and its own internal state to achieve a goal.

* **Perception:** The agent ingests information from external sources. This includes the initial user prompt, data retrieved from knowledge bases or APIs, and observations resulting from its own previous actions.
* **Reasoning:** The agent processes the perceived information and its own internal memory. This is the core cognitive step where the LLM evaluates the current state, reflects on the goal, and decides on a course of action.
* **Action:** Based on its reasoning, the agent executes an action. These actions can be internal (modifying its own state) or external (interacting with tools).

##### **1.1. Mathematical Formalism of the LLM-Policy Agent**

To ground this framework, we formalize the agent as an **LLM-policy agent** operating in a sequential decision process. The agent can be described as a tuple (S,A,T,π), which builds upon the theory of Markov decision processes.

* S: The **State Space**. This represents the complete state of the agent at any given time, including its internal memory, dialogue context, and any observable environmental state. In our architecture, this corresponds to the GraphState object, which encapsulates the user prompt, retrieved documents, the working plan, and generated artifacts.
* A: The **Action Space**. This is the set of all actions available to the agent. A key innovation in this model is the distinction between two types of actions:  
  + **External Actions** (Aext​): These actions interface with the environment, such as querying a database, calling a search API, or running a piece of code. Each external action produces an observation o from the environment, which updates the agent's state.
  + **Internal Reasoning Actions** (Aint​): These are the agent's own thought steps, which update its internal state without querying an external source. Examples include creating or refining a plan, forming a hypothesis, or generating a self-critique. These "thought" actions are crucial as they influence all subsequent decisions.
* T: The **Transition Function**. This function defines the system's dynamics, describing how the state updates in response to an action, st+1​=T(st​,at​,ot​).
* π: The **LLM-Policy**. This is the core reasoning component of the agent. Unlike a traditional policy in an MDP, the policy π is realized by a large language model generating actions in natural language or structured text.

We can formally describe one iteration of the agent's PRA loop as follows: the agent, in state st​, samples an action at​ from the policy distribution generated by the LLM with parameters θ: at​∼πθ​(a∣st​). The subsequent state transition depends on the type of action taken:

1. If at​ is an **external action**, the environment produces an observation ot​=Env(st​,at​), and the agent's state updates to st+1​=f(st​,at​,ot​).
2. If at​ is an **internal reasoning action**, the state updates internally: st+1​=f(st​,at​). For example, a thought is appended to an internal scratchpad, modifying the state for the next step.
3. If at​ is a terminal **output action** (e.g., OUTPUT(final\_answer)), the loop terminates and presents the result to the user.

This loop can be viewed as the agent traversing a decision tree of states via actions until a terminal node is reached. This general architecture is agnostic to any specific library and can be implemented with raw API calls or higher-level frameworks like LangGraph, which explicitly models agent behavior as a state-directed graph.

#### **2. "Plan, Synthesize, Review" (PSR): A PRA Implementation for Artifact Generation**

Our primary architecture, PSR, is a structured and highly effective implementation of the PRA framework, tailored for generating complex, coherent documents. It applies the PRA loop at both a macro and a micro scale.

* **Macro-Level PRA Cycle (The Overall Workflow):**
  + **Perception:** The agent perceives the user's prompt and retrieves the initial knowledge base (retrieve\_context).
  + **Reasoning:** The agent reasons about the task to decide on a workflow (route\_task) and then creates a high-level strategic plan (create\_plan).
  + **Action:** The agent executes the entire synthesis and review process (the synthesis\_loop and holistic\_review), which is itself a complex action composed of many smaller PRA cycles. The final action is presenting the completed artifact to the user.
* **Micro-Level PRA Cycle (The Iterative Synthesis Loop):** The real power of the PSR model is in how it breaks down the monolithic "generation" task into a series of smaller, more manageable PRA cycles, one for each item in the plan. For each section of the document to be written:  
  + **Perception:** The agent perceives the specific sub-task from its plan and may perform a targeted retrieval of information relevant only to that sub-task.
  + **Reasoning:** The agent synthesizes the retrieved information and generates the text for that single section.
  + **Action (Internal):** The agent performs the internal action of saving the generated text to its working\_memory.

#### **3. Architectural Benefits**

Mapping our PSR workflow to the PRA framework provides several key benefits:

* **Theoretical Rigor:** It shows that our design is not ad-hoc but is grounded in a fundamental and widely applicable theory of agentic behavior.
* **Modularity and Extensibility:** By viewing each node in our graph as an implementation of Perception, Reasoning, or Action, it becomes easier to modify, replace, or add new capabilities. For example, adding a new external tool is simply adding a new possible Action.
* **Improved Debuggability:** Failures can be categorized more easily. Is the agent failing at **Perception** (not getting the right information), **Reasoning** (making a bad plan), or **Action** (failing to execute a step correctly)? This targeted diagnosis is crucial for complex systems.

By adopting this unified framework, we elevate our project from a specific solution to a principled architecture that is both powerful for its current task and readily extensible for future challenges.