Expert Systems

WILEY

Expert Systems

ORIGINAL ARTICLE OPEN ACCESS

Thinking Is Not Enough: The *B** Expansion Technique for Enhancing Autonomous LLM Agents

Sebastián Andrés Mayorquín Posadas | Julio Vega

Department of Telematic Systems and Computing, Rey Juan Carlos University, Fuenlabrada, Madrid, Spain

Correspondence: Julio Vega (julio.vega@urjc.es)

Received: 24 October 2025 | **Revised:** 13 May 2026 | **Accepted:** 8 June 2026

Keywords: action space | agents | AGI | *B** expansion | LLM | performance | prompt engineering

ABSTRACT

Most autonomous agents built on large language models (LLMs) focus on improving internal reasoning while assuming a fixed and complete set of primitive actions. In this work, we challenge this assumption and introduce the *B** expansion technique, an action-centric method that enables agents to iteratively construct new actions beyond an initial basis *B*, forming an expanded action space $B^* = B \cup A^*$. To formalize this paradigm, we present a unified framework for act-centric autonomy that explicitly models action space evolution, distinguishing our approach from prior frameworks such as ReAct, which operate over fixed action sets. Within this framework, *B** serves as a concrete instantiation that operationalizes dynamic action generation. We evaluate *B** in a non-trivial, fully observable environment—Conway's Game of Life on a 200×200 grid—where the objective is to maximize the number of live cells after 300 generations. Our results show that expanding the action space leads to improved exploration and performance, demonstrating that action space completeness is insufficient and that optimizing the action set itself can significantly enhance agent effectiveness, even surpassing state-of-the-art reasoning-based approaches.

1 | Introduction

Recent developments in large language model (LLM) agents have followed two closely related directions: improving internal reasoning capabilities and expanding interaction through external tools (Figures 1 and 2).

On the reasoning side, progress has been driven by large reasoning models (LRMs) and advanced 'thinking' strategies, including OpenAI o1/o3/o4 (OpenAI 2024a, OpenAI 2024b), Claude Sonnet 4 (Anthropic 2025) and DeepSeek R1 (DeepSeek Inc. 2025). These approaches improve performance by increasing test-time computation and refining step-by-step reasoning.

In parallel, the development of tool ecosystems has significantly extended the capabilities of LLM-based agents. Modern agents can now interact with heterogeneous external systems, ranging from simple calculators to complex APIs such as Notion, Google Calendar or Python execution environments. Recent standardization efforts further accelerate this trend by enabling more seamless integration of tools across systems.

Together, these two directions have led to hybrid reasoning-and-acting frameworks such as ReAct, which combine step-by-step reasoning with tool use to improve reliability and reduce hallucinations. However, these approaches still operate over a fixed and predefined set of actions, leaving open the question of how the structure and size of the action space itself influences agent performance.

There is a clear synergy between these trends, studied and applied since the publication of ReAct (Yao, Zhao, et al. 2023), a strategy that combines reasoning and acting to reduce hallucinations in LLMs. However, as the number and diversity of

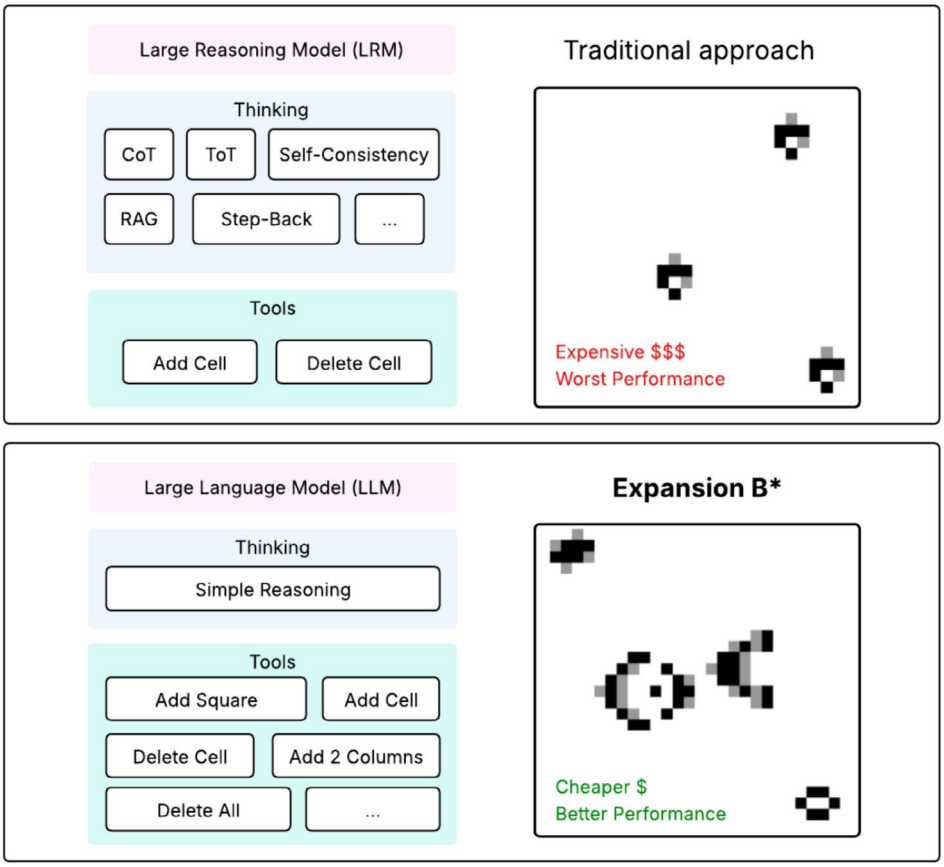

FIGURE 1 | Comparison between traditional LRM-based agents and the proposed B^* expansion framework. The left side illustrates an LRM operating with a fixed toolset, while the right side shows an LLM augmented with an expanded action space B^* applied to the Game of Life environment.

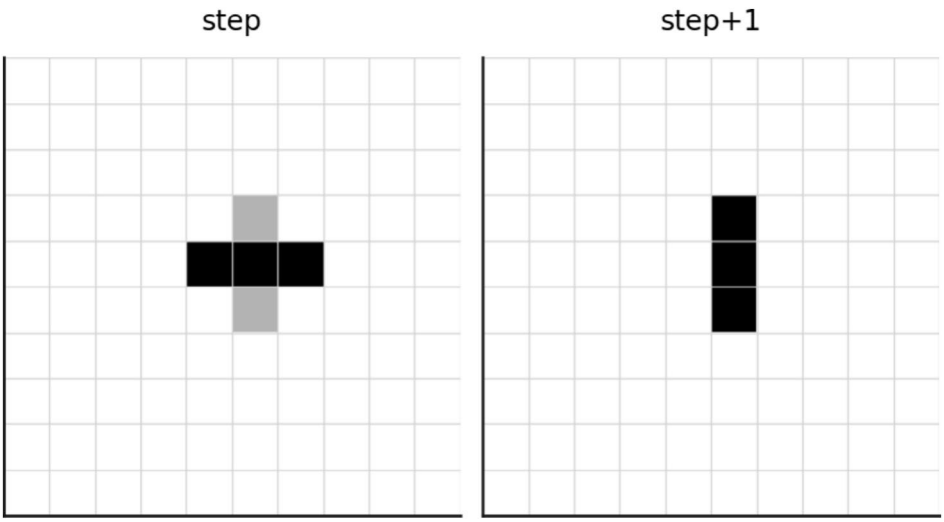

FIGURE 2 | Example of a single transition in Conway's Game of Life. The left panel shows an initial configuration at step t, while the right panel displays the resulting state at step $t + 1$. Grey cells in the left panel indicate positions that become alive in the next step according to the Game of Life update rules.

available tools increases, a new challenge emerges: the agent must not only reason correctly but also select the most appropriate action from an increasingly large and heterogeneous set.

In current approaches, this action selection process is typically handled implicitly by the model's internal reasoning, without explicitly addressing how the structure or size of the action space affects performance. As a result, even strong reasoning models may struggle when faced with large or poorly structured toolsets, leading to inefficient exploration or suboptimal decisions.

These observations suggest that reasoning and acting alone are not sufficient to fully characterize or optimize the performance of autonomous LLM agents. As tool ecosystems grow in scale

and diversity, the primary challenge shifts towards how actions are organized, generated and selected within an increasingly large and heterogeneous action space.

From this perspective, we argue that the structure of the action space itself becomes a first-order design variable in autonomous agent systems. However, existing approaches largely treat the action space as fixed and externally defined, relying on implicit selection mechanisms without explicitly modelling its impact on performance.

This article addresses this limitation from a perspective that contrasts with the dominant approach: instead of further optimizing the reasoning process, we focus on constructing and expanding the action space so that even agents with weaker reasoning capabilities can achieve strong performance. To operationalize this idea, we introduce B^* expansion and a supporting formal framework for action-space evolution.

To address these challenges, this paper makes the following contributions:

- We introduce B^* expansion, an action-centric technique that enables LLM agents to iteratively generate and incorporate new actions, transforming a fixed action basis into an evolving action space.
- We present a formal framework for act-centric autonomy that characterizes how action spaces can be structured, extended and analysed, providing a principled foundation for action-space optimization.
- We demonstrate through experiments in a controlled environment (Conway's Game of Life) that expanding and structuring the action space can significantly improve exploration and performance, allowing simpler models to match or surpass more advanced reasoning-based approaches.

This paper is structured as follows. Section 2 reviews related work on reasoning-based models and tool-augmented agents, highlighting current limitations in action-space design. Section 3 introduces the formal foundations of the proposed approach, including the definitions of problems, actions, and the cognition function, and presents the B^* expansion technique. Section 4 describes the experimental setup, including the Game of Life environment and the criteria for evaluating action-space expansion. Section 5 presents the experimental results, including the evolution of performance as the action space grows and a comparative analysis with reasoning-based models. Finally, Section 6 concludes the paper and discusses limitations and directions for future work.

2 | Related Work

2.1 | Tools and Agentic Standardization

Previously to November 2024, the most common approach to autonomous agents has been programming custom functions and binding them to the LLM by using function calling or structured tool calling (if supported by the LLM). Although this approach has been effective and opened the door to the development of autonomous (Fezari and Ali-Al-Dahoud 2023; TransformerOptimus 2023; Wu 2023; Reworkd 2023) and even embodied (Wang et al. 2023; Lifshitz et al. 2024; Wang et al. 2023) agents in virtual/real world, the limitation of this approach is that all the tools must be programmed manually by the developer, which is a time-consuming and error-prone task (Shi et al. 2025; Ma et al. 2025). Some standardized protocols were created to intercommunicate different tools and LLMs, accompanied by libraries that minimized the programming effort (Chase and Gola 2022; Oberhauser and Team 2019; LangChain Team 2024; CrewAI Inc. 2024). However, a sudden proliferation of tools emerged with the introduction of the Model-Context Protocol (MCP) (Anthropic 2024b) where all the tools now were unified under a single protocol, allowing developers to create agents that could easily integrate with a wide range of tools and services (Singh et al. 2025; Jordan and ElevenLabs 2025). This approach has also been extended to the standardization and intercommunication of Agents or prompts with the introduction of A2A (Agent to Agent) protocol (Surapaneni et al. 2025), which has been seamlessly integrated with MCP into agentic-based frameworks. This sudden growth of tools and agents has led to the creation of a new paradigm in autonomous agents, where the focus is on selecting the most appropriate tools for a given task and where the LLM cannot only hallucinate when trying to reason, but also when selecting the most appropriate tool for a given task.

2.2 | Reasoning Models

The most common approach to evaluate LRM has been the use of science or culture-based benchmarks (Yehudai et al. 2025; DeepSeek Inc. 2025; Qwen et al. 2025; OpenAI 2023). These benchmarks allowed comparison of how different models could be trained on different 'thinking'-based techniques in order to improve their performance on problem-solving tasks. These techniques included Chain of Thought (CoT), Tree of Thought (ToT) or ReAct, which enable trading test-time compute for better reasoning performance (Snell et al. 2024; Yao, Zhao, et al. 2023; Yao, Yu, et al. 2023; Sahoo et al. 2024). However, these approaches have also been tested on agentic-based environments (Shridhar et al. 2020; Wang et al. 2023; Shojaee et al. 2025) where the LLMs are susceptible to hallucinating and propagating the errors throughout the entire reasoning/acting process, not making the LRMs fully suitable for embodied or agentic applications. Additionally, LRMs have also been limited by the time and resource consumption required to execute these models, thus not allowing the use of LRMs in real-time or low-cost applications. Alternative techniques such as Retrieval-Augmented Generation (RAG) (Chen et al. 2024; Jiang et al. 2023) try to externalize the reasoning process to automated and efficient systems in order to improve the performance of LLMs without having them reason by themselves. This approach, however, is limited by the fact that the development of RAGs requires a lot of time and effort to create and maintain.

This context of LRMs and tools has led to the development of new strategies that try to balance the trade-off between reasoning and tool calling in order to improve the performance of LLMs and reduce the time or resource consumption required to execute them. It is important to note that this balance requires the tools to be as effective as possible, so that the LLMs do not

reduce performance on the overall environments by using invalid or irrelevant tools.

Despite these advances, a key limitation remains insufficiently addressed in the literature. Existing approaches largely assume that the set of available actions or tools is fixed and focus either on improving the reasoning capabilities of the model or on expanding the availability of external tools. However, little attention has been given to how the action space itself should be structured, adapted or expanded during execution.

In particular, current frameworks—including approaches such as ReAct (Yao, Zhao, et al. 2023)—rely on the implicit ability of the model to select from a predefined set of actions, without explicitly modelling how the size, composition or evolution of this action space affects performance. As a result, the problem of action-space optimization remains underexplored.

This gap motivates our work, where we shift the focus from reasoning optimization to action-space construction and evolution as a first-class design problem. That is why we propose B^* expansion as a method to explicitly construct and evolve the action space, enabling agents to generate new actions dynamically and improving performance beyond what can be achieved through reasoning or tool availability alone.

Compared to existing approaches, our work differs along a fundamental dimension. Methods such as Chain-of-Thought, Tree-of-Thought and ReAct focus on improving the reasoning process over a fixed set of actions, while tool-centric frameworks emphasize expanding the availability of external capabilities. In contrast, our approach treats the action space itself as dynamic, enabling agents to construct new actions during execution rather than selecting solely from a predefined set. This shift allows us to study how action-space evolution impacts performance, a dimension that is largely absent in prior work.

3 | Formal Foundations

In order to define the B^* expansion technique, we will first establish a formal set of definitions and axioms that will allow us to standardize the concepts used in this article. For this we will take the Action Space concept defined by ReAct (Yao, Zhao, et al. 2023) as the basis for our definitions, but we will simplify the cognition and reasoning process, and extend the Action Space to cover a more suitable definition for autonomous agents.

We emphasize that the following formalization is an idealized abstraction intended to support theoretical analysis of action-space structure in LLM-based agents. The assumptions introduced in this section should be interpreted as modelling conveniences rather than properties expected to hold in real-world environments.

3.1 | Problems, Actions and Solutions

To provide a formal characterization of action-space optimization, we introduce an idealized abstraction of problems, actions and solutions. This formulation is not intended to describe real-world environments exhaustively, but rather to isolate the structural properties relevant to autonomous agent behaviour.

We build upon the action-space formulation introduced in ReAct, while simplifying the reasoning process and focusing on the role of action composition and selection. Under this abstraction, we define the following sets.

We consider a potentially unbounded set of problems P, which allows us to avoid degenerate cases where the action space can be exhaustively enumerated. We assume:

$$\forall p_i, p_j \in P, p_i \neq p_j \Rightarrow \neg\left(p_i \equiv p_j\right), \tag{1}$$

where $\equiv$ denotes semantic equivalence under the cognitive interpretation.

We assume, for analytical convenience, that each problem $p \in P$ admits a well-defined representation up to semantic equivalence, allowing us to abstract away representational redundancy. So we can consider P as a set of really distinct problems, where we assume that solutions can be treated as equivalence classes of action sequences that satisfy the problem constraints. We will assume the set of problems is infinite to avoid the possibility of brute-forcing the solution space, which would lead to a trivial solution for any problem.

Let A denote the infinite set of actions (called as Action Space), where each $a \in A$ represents a fundamental interaction that an LLM can perform.

We denote the action space as infinite since the set of actions implemented by a developer could be extended indefinitely.

Let S be the solution space, defined as the set of all possible sequences of actions:

$$S = \left\{ s = \left(a_1, a_2, \ \ldots, a_n\right) \mid n \in \mathbb{N}^+, a_i \in A \right\}, \tag{2}$$

where each sequence $s \in S$ is immutable and order-sensitive. Thus, two sequences with the same actions in different orders are considered different solutions.

For each problem $p \in P$, let $S_p \subset S$ be the non-empty set of sequences that satisfy p:

$$s \in S_p \Leftrightarrow s \in S \wedge s \vDash p, \tag{3}$$

where $s \vDash p$ indicates that s resolves problem p under the system's semantics.

We further assume:

1. Mutual exclusivity of solutions. $\forall s_i, s_j \in S_p, s_i \neq s_j \Longrightarrow s_i \equiv s_j$.
2. Existence of solutions. $\forall p \in P$, $\exists s \in S_p$ such that $s \vDash p$.

These assumptions are introduced to simplify the theoretical analysis and to focus on the structure of the action space rather than representational ambiguity or multiple valid solution forms. At this point, this environment states that all problems are well-defined and can be solved by using a finite set of actions

$(a_1, a_2, \ldots, a_n)$ from the set of actions A, which are the actions that an LLM can perform.

3.2 | The Cognition Function

Let P denote the set of well-formed problems and A the set of actions available to an agent. For any finite action space $X \subseteq A$, we define the cognition function as a transformation that maps a given problem to a finite action sequence, using only elements from X, based on the knowledge latent in a fixed language model C.

Let C be a fixed (opaque) language model, $p \in P$ a problem instance and $X \subseteq A$ a finite set of available actions. The cognition function is defined as:

$$\mathbf{T_c}: (C, p, X) \mapsto (x_1, x_2, \ldots, x_k) \tag{4}$$

such that $\forall i, x_i \in X$ and $(x_1, \ldots, x_k) \in S$, where S is the set of candidate solutions formed by finite sequences of actions that satisfy the problem p under the constraints of using only actions from X.

3.2.1 | Semantic Domain

The output sequence $(x_1, \ldots, x_k)$ is interpreted as a candidate solution that, according to the model C, resolves the problem p under the constraint of using only actions from X. No assumptions are made about the internal mechanism of C, its architecture, the way it selects the sequence nor the reasoning mechanism used.

We denote by $\mathcal{R}(X) \subseteq S$ the subset of solutions that C is capable of generating (called the reachability of the model) using only actions from X. The cognition function $\mathbf{T_c}$ thus produces outputs from $\mathcal{R}(C, X)$:

$$\mathbf{T_c}(C, p, X) \in \mathcal{R}(C, X) \tag{5}$$

3.2.2 | Non-Determinism

The function $\mathbf{T_c}$ is, in general, non-deterministic: repeated evaluations may yield different sequences, even for the same (C, p, X), due to the stochastic nature of LLMs.

The central empirical hypothesis of this work is that the cardinality and composition of X directly influence the reachability set $\mathcal{R}(C, X)$, and hence the quality and feasibility of the outputs of $\mathbf{T_c}$. This hypothesis is explored in depth in subsequent sections.

3.3 | Action Space and the B^* Expansion Technique

We now introduce a finite subset $B \subset A$, called the action basis, which serves as a generative core for the entire space A. The set B is assumed to be complete, in the following formal sense:

An action basis $B \subset A$ is complete if and only if every action $a \in A$ can be expressed as a finite composition of actions in B:

$$\forall a \in A, \exists (b_1, \ldots, b_k) \subset B \text{ such that } \texttt{comp}(b_1, \ldots, b_k) = a \tag{6}$$

where comp is a composition function mapping finite sequences over B to actions in A.

This implies that the basis B is, in principle, sufficient to construct any solution $s \in S$, although possibly in a suboptimal or excessively long form. This cannot be guaranteed for all environments, but can be assumed for some practical applications where the action space A can be mostly decomposed into a finite set of atomic actions B. For example, in the environment of controlling a computer, all the actions could be decomposed into a large but finite set of atomic actions such as 'click', 'type', 'scroll', and so forth.

This space B has been used by some autonomous agents (Shojaee et al. 2025; OpenAI 2025; Anthropic 2024a) to try to solve problems in domains where the action space is infinite but the implementation of low level actions can be enough to solve most of the problems proposed.

As stated, completeness ensures solvability but it does not imply optimality. In many cases, using only actions from B leads to solutions that are semantically correct but inefficient with respect to some performance measure (e.g., time, resource usage, compactness). Formally, for a problem $p \in P$, there may exist multiple solutions:

$$s_{\text{sub}} = (b_1, \ldots, b_m) \in B^m s_{\text{opt}} = (a_1, \ldots, a_k) \in A^k \tag{7}$$

such that both $s_{\text{sub}} \vDash p$ and $s_{\text{opt}} \vDash p$, but $s_{\text{opt}} \prec s_{\text{sub}}$ under some preference ordering $\prec$.

To approach this balance between completeness and optimality, we define A^*, the action space approximator, as the set of all actions that can be generated by a growing sequence action spaces $A_0^*, A_1^*, \ldots \subset A$, where:

$$A_0^* = \emptyset, \quad A_n^* \subset A_{n+1}^*, \quad \bigcup_{n=0}^{\infty} A_n^* = A \tag{8}$$

Each set A_n^* introduces new actions that were not present in previous stages. These actions act as useful abstractions that are compositions of actions in B, but could add different properties to the action space, such as higher-level definitions, or easier implementations of complex actions.

To ensure the validity of the expanded action space, each candidate action proposed for A^* undergoes a filtering process before inclusion. First, structurally invalid actions (e.g., malformed arguments or undefined operations) are discarded. Second, redundant actions—defined as actions functionally equivalent to existing elements in B^*—are removed using deterministic equivalence checks over action signatures. Only actions passing both criteria are incorporated into A_{n+1}^*.

In practice, the construction of A^* is implemented as an incremental generation process. At each stage n, candidate actions are proposed based on a combination of (i) observed failures or inefficiencies in $\mathcal{R}(C, A_n^*)$, (ii) decomposition of successful

action sequences and (iii) optional external specifications provided through tool interfaces such as MCP or A2A protocols.

These candidate actions are then validated and either incorporated into A^*_{n+1} or discarded based on their utility in improving solution quality or expanding coverage of the action space. This mechanism allows A^* to be generated in a semi-automated manner without modifying the underlying language model C.

In the context of autonomous agents, A^* represents a growing set of actions that could be created by a developer, agent or extracted from MCP/A2A protocols, which are not necessarily needed to solve any problem, but are useful to improve the quality of the solutions generated by the agent.

Finally, we define the expanded action space B^* as the union of the action basis B and a growing approximator $A^* \subset A$:

$$B^* = B \cup A^* \tag{9}$$

where $A^* = A^*_n$ at any fixed time step n. Thus, the expanded action set retains the completeness guarantee of B, while incrementally improving the reachability of the agent by granting access to higher-level or composite actions.

Since $B \subset A$, and $A^* \subset A$, it follows that $B^* \subseteq A$. However, unlike A, which is infinite and not practically accessible, B^* is a finite and incrementally extensible action set usable by real systems.

The development of an expanded action space, called as the expansion B^* technique enables the cognition function to:

- Guarantee solvability for any problem $p \in P$, due to the completeness of B if the problem p is reachable by the LLM (thus, if $T_c(C, p, B^*) \in \mathcal{R}(C, B^*)$)
- Improve the quality of generated solutions by incorporating actions from A^*, approximating optimal plans.
- Scale incrementally, by adding actions to A^* based on observed needs or performance feedback, without modifying C or retraining the model.

If the LLM C were ideal and could reason perfectly, then the expansion B^* would not be necessary, as the LLM could generate optimal solutions using only the actions in B. However, since LLMs are not perfect, the expansion B^* allows to improve the performance of the LLM by providing it with a growing set of actions that can be used to generate better solutions.

The effectiveness of the expansion B^* technique will be empirically evaluated in the next sections, where we will measure the growth of reachable solutions $\mathcal{R}(C, B^*)$ as A^* increases.

4 | Experimental Setup

All experiments use a fixed prompt template instructing the model to maximize the final number of live cells after 300 steps. The model is restricted to selecting actions from B^* via a structured tool-calling interface.

We set temperature = 0 for deterministic comparisons in LLM configurations, while LRM configurations use their default reasoning settings. The maximum number of tool calls per episode is fixed across all configurations to ensure fairness.

To empirically validate the expansion technique B^*, it has been considered an environment that meets the following characteristics:

- *Non-triviality:* The environment must be sufficiently complex so that the use of a direct or obvious strategy is not enough to solve the problem. Additionally, this environment must be highly sensitive to the input used, in such a way that we can emulate real-world environments where the order and selection of tools cannot be easily predefined by a single prompt (such as in the case of fully controlling a computer).
- *Completeness:* The environment must contain an action space B that is finite and complete, that is, that allows solving any problem posed in the environment (a fundamental requirement for the expansion technique B^*).
- *Expandable:* The environment must allow the incorporation of new actions into the action space A^* incrementally, with these actions always being compositions of actions in B, and that can be used by the LLM to improve its performance in problem solving.
- *Measurable:* The environment must allow the performance of the LLM to be measured with a quantitative metric, so that we can compare the result precisely without relying on subjective interpretations
- *Initially Unreachable:* The environment must be complex enough so that the solution is not within the reach of the LLM through 'memorization' from other sources of information. This means that although the LLM may have notions of how to solve the problem, it should not know the solution directly, but rather be capable of solving it through the available actions.

In the experimental setup, the base action set B is defined as the minimal set of primitive operations available in the environment: (i) placing a live cell in a selected grid position, (ii) removing a live cell and (iii) querying the current state of the board. These actions are assumed to be sufficient to fully interact with the Game of Life environment at a low level, but not necessarily efficient for constructing complex configurations.

Given this set of requirements, the proposed environment is Conway's Game of Life: Conway's Game of Life is a two-dimensional environment where each cell can be alive or dead. At each step, the board evolves according to simple rules based on neighbouring cells, giving rise to complex and hard-to-predict behaviours. The tools available in B will be the basic actions of the Game of Life, that is, placing a live cell, removing a live cell, and viewing the state of the board. The actions in A^* will be composite actions that can be generated from the actions in B, such as 'place a row of live cells', 'remove a column of live cells', etc. In this environment, the agent may define an initial pattern within a 200 $\times$ 200 board where the number of live cells at the beginning of the game is less than or equal to 50. The overall goal will be to maximize the number of live cells at the end of the game, after 300 steps

have passed on the board and with the final cells being the result of the board's evolution from the initial pattern.

The interaction between the LLM and the environment follows a tool-calling loop, where the model C receives the current state of the board and a description of the task, and must select one or more actions from B^* to apply. Each action is executed in the environment, which deterministically updates the board according to the rules of the Game of Life. The updated state is then returned to the model for the next decision step.

Each episode follows the procedure:

1. Initialize Game of Life environment.
2. Provide initial state to model C
3. Model selects action $a \in B^*$ via prompt-based tool calling
4. Execute action in environment
5. Receive updated state
6. Repeat until termination (300 steps)
7. Compute final score

For A^* generation:

1. Sample failure or suboptimal trajectories
2. Extract candidate action patterns
3. Propose new action via LLM or decomposition
4. Validate and add to A^* if accepted

Note that the final result of the board will depend solely on the initial pattern and the rules of the game, where the agent will not be able to interact during intermediate steps, and the result will be highly sensitive to the pattern used.

This environment meets the aforementioned requirements and will be used with both LRMs and LLMs in order to compare the performance of the expansion technique B^* in both cases.

5 | Experiments and Results

The current section presents the results obtained from the experiments conducted in the proposed environment, where the performance of the LLMs will be evaluated as the action space A^* expands.

All experiments use a fixed prompt template that instructs the model to maximize the final number of live cells after 300 steps, while explicitly restricting action selection to the available set B^*. No task-specific prompt engineering beyond this instruction was used.

5.1 | Evolution of Performance

Given the proposed environment, the first experiment conducted consists of evaluating the performance obtained by an LLM as the action space A^* expands. For this, the OpenAI model `gpt-4o-mini` was used due to its balance between performance and cost. The tool calling functionality is used to interact with all the tools in B^*.

For the generation of actions in A^*, no set of manually developed or predefined actions was programmed; instead, the same LLM model was used to generate, at each iteration of the experiment, a new tool that would be added to the action space A^*. This was done to avoid possible biases in the action space that could violate the environment's non-triviality property. This action generation process was carried out completely independently and without information, memory or context from previous runs of the environment, so that the model creating the actions could not 'correct' errors made in previous iterations.

To mitigate potential bias in the construction of A^*, the generation of new actions is performed using the same model C but with strict separation between generation and execution contexts. Specifically, the tool-generation process has no access to performance outcomes from the experimental runs and is executed independently for each run without shared memory or cross-run information.

This design avoids direct optimization of A^* towards observed test-time performance and ensures that improvements in B^* arise from structural action-space expansion rather than closed-loop adaptation to specific trajectories.

In total, 1300 runs were performed. The first 500 were done with 20 players with a number of actions in A^* varying between 0 and 25 actions (always adding the action space B as a base). It is important to note that the actions in A^* are not the same among different players, meaning that each player generated a different set of actions, which allows evaluating the performance of the B^* expansion independently of the generated actions.

The remaining 800 runs were conducted in a second experiment with 20 players, and up to 40 rounds of expanded action space A^*, to evaluate the performance of the B^* expansion in an even larger space.

Figure 3 (left) shows the evolution of the median score as the size of the extended action space A^* increases. We observe a clear upward trend: during the initial rounds, the agent tends to stabilize around 50 points, which roughly corresponds to the number of alive cells it is allowed to place initially. This indicates that the LLM, when limited to the base action set B, primarily discovers configurations that do not leverage the full dynamical potential of the environment.

However, starting around round 7, there is an abrupt increase in performance. This inflexion point reflects the influence of higher-level actions via A^*, which appear to unlock novel strategies that go beyond simple cell placements. The trend stabilizes gradually, reaching median values close to 100 points (double the initial score) by round 25.

Figure 3 (right) complements this analysis by displaying the full distribution of scores across all rounds. In early stages (low $|A^*|$), score distributions are tightly concentrated, reflecting the agent's limited capacity to explore diverse outcomes. Most

scores cluster near the initial placement baseline, with minimal deviation.

As the action space expands, we observe a pronounced increase in score variance. This is expected in highly sensitive environments like Conway's Game of Life, where small variations in initial conditions can lead to radically different outcomes after 300 simulation steps. The LLM, now equipped with richer tools, is able to explore a broader range of strategies-some highly effective, others less so-resulting in both higher peak scores and greater dispersion. This behaviour aligns with the hypothesis that expanding the action space $B^* = B \cup A^*$ enhances the agent's effective reach $\mathcal{R}_C(B^*)$, not only in median performance but also in diversity of achievable solutions.

When repeating and extending the same experiment to 40 rounds (Figure 4), we observe that the median performance maintains stable, indicating that the reach of the LLM has saturated under the current cognitive function T_c. This suggests that further gains are no longer driven by the growth of the action space B^*, but instead depend on the model's inherent ability C to reason over increasingly complex and abundant tools.

Nevertheless, a noteworthy trend emerges in the latter rounds: the dispersion of scores begins to narrow. While earlier stages showed a wide variance due to the chaotic nature of the environment and the exploratory nature of the model, rounds approaching iteration 40 display consistently high scores centered around 100, with reduced variability across runs. This indicates that as the action space becomes richer, the model not only retains high performance but also gains in robustness, consistently selecting more effective strategies with less reliance on stochastic search.

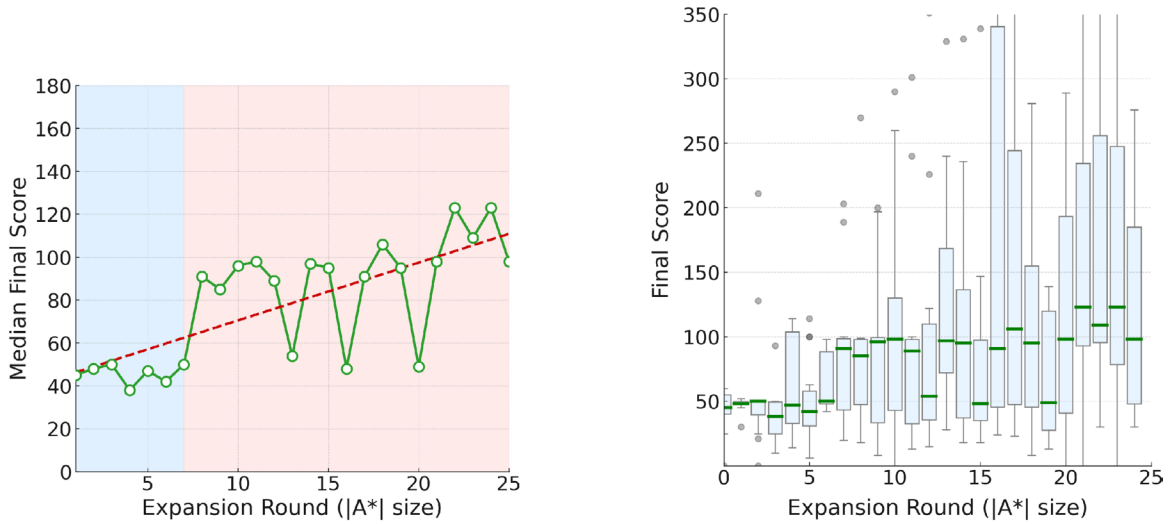

FIGURE 3 | Score evolution as the A^* action space increases. Left: Median final score over 25 expansion rounds. The green line shows the observed median values, while the red dashed line indicates a fitted trend over the rounds. Right: Distribution of final scores across the same trials, represented using box plots for each expansion round.

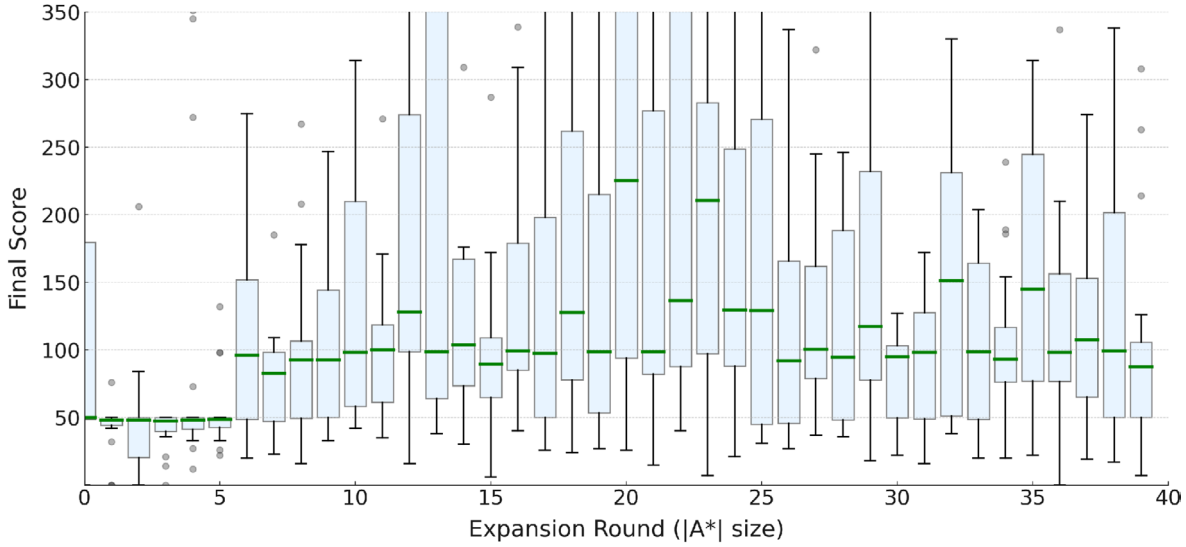

FIGURE 4 | Distribution of final scores across 40 expansion rounds. Each box plot represents the score distribution for a given size of the A^* action space.

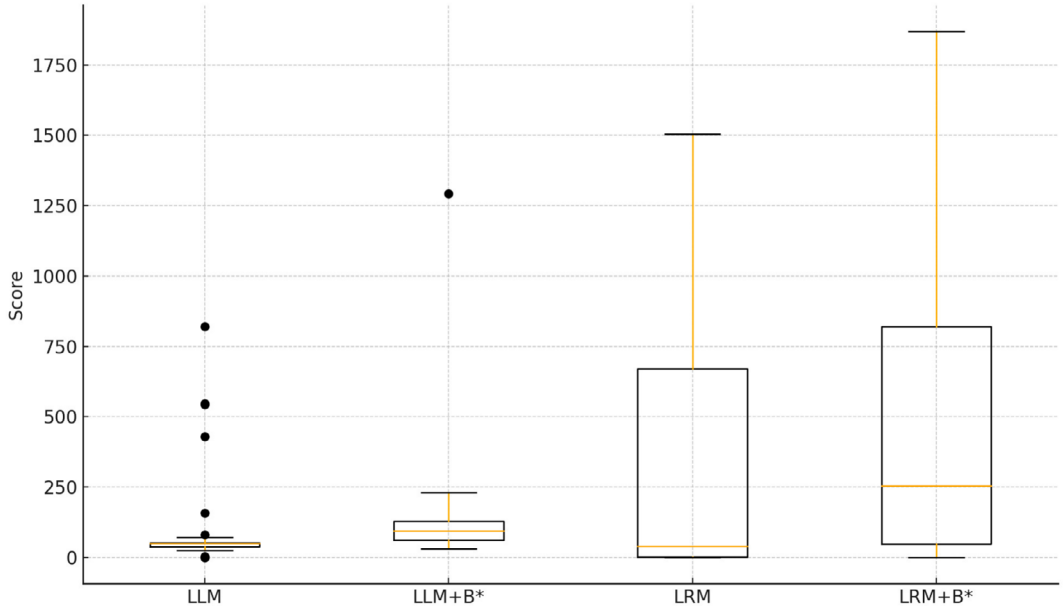

FIGURE 5 | Boxplot comparison of the four tested configurations. The figure shows the distribution of final scores for each setup. The *LLM+B** configuration exhibits a higher median score than the *LRM* configuration, although the *LRM* shows a wider score range. The *LRM+B** configuration achieves both higher median values and higher maximum observed scores compared to the other setups.

5.2 | Comparative Analysis With LRMs

To validate the effectiveness of the expansion B^* against traditional reasoning strategies, four configurations have been defined reusing the B^* sets generated in the previous experiment. The objective will be the same as in the previous experiment: to maximize the number of alive cells; only the language model used and the number of available actions (without including prior information or additional hints) will change.

First, an LLM with access only to the base set B will be considered, which is complete but requires the model to combine basic sequences by itself. Next, the same LLM is evaluated with access to the expanded set B^* (extracted at each evaluation from round 40 of a random player from the previous experiment), allowing it to use higher-level previously generated actions.

On the other hand, two reasoning variants (LRM) are compared: one operating only with B, and another that has the B^* set. These configurations allow analysing the combined effect of reasoning and expansion on the final performance. This will allow evaluating whether even an LRM can benefit from the B^* expansion or if, on the contrary, reasoning alone is sufficient to reach optimal performance.

Both LLM and LRM configurations use identical prompts, identical tool interfaces, and identical environmental constraints. The number of tool calls and episode length (300 steps) are fixed across all experiments to ensure controlled comparison. The only variables are the underlying model and the available action space (B vs. B^*).

In this second experiment, the `gpt-4o-mini` model was used as the base model and `o4-mini` as the LRM trained on the reasoning techniques previously described. One hundred twenty runs were performed (30 runs for each of the models configurations, both with and without expansion of the action space B^*). As seen in Figure 5, the LLM without reasoning, limited to the base set B, tends to get stuck in the initial configuration similarly to the previous experiment.

In contrast, the LRM applied over the same base set shows a markedly different dynamic: although it occasionally achieves much higher scores, it does so with extreme dispersion. The higher variance observed in LRM configurations suggests increased stochasticity in reasoning trajectories under constrained action spaces, leading to inconsistent exploitation of partially optimal strategies. Surprisingly, this reasoning variant is outperformed by the LLM without explicit reasoning thanks to the use of the B^* expansion technique, obtaining a higher median and a much more stable distribution of results. This behaviour points out that, in the absence of specialized tools, unstructured reasoning may degrade the overall quality of solutions, generating higher variance without providing systematic improvements.

Finally, enabling the B^* expansion shows that the LRM can also benefit significantly. The use of B^* tools drastically substantially raises the median, allowing not only higher results to be reached but also doing so more consistently.

Table 1 offers a simplified quantitative summary of the results obtained by each configuration. We observe that the B^* expansion consistently improves the median score across both LLM and LRM models, with the LLM achieving a median of 93 points and the LRM reaching 253 points.

To complement the summary statistics, we note that the reported ranges reflect the inherent stochasticity of both the environment and the action selection process. In particular, zero scores correspond to runs in which the agent fails to construct any stable or beneficial configuration within the initial steps, leading to rapid extinction of live cells under the Game of Life

TABLE 1 | Summary of Experiment 2. This table provides a simplified quantitative overview of the results obtained by each configuration.

Model	B^* expansion	Median	Total range
`gpt-4o-mini`	—	50	0–820
`gpt-4o-mini`	✓	93	30–1292
`o4-mini`	—	40	0–1504
`o4-mini`	✓	253	0–1870

dynamics. These cases are more frequent in configurations without B^*, and become progressively rarer as the action space expands.

To assess statistical significance, we perform pairwise non-parametric comparisons using the Mann–Whitney U test across all experimental configurations, applying Bonferroni correction to account for multiple comparisons. Specifically, we compare LLM versus LLM+B^*, LRM versus LRM+B^* and LLM+B^* versus LRM+B^*.

In addition to hypothesis testing, we report effect sizes using Cliff's Delta to quantify the magnitude of performance differences, together with 95% bootstrap confidence intervals for median scores to capture uncertainty in the estimates.

The results confirm that all configurations incorporating B^* exhibit statistically significant improvements over their non-expanded counterparts ($p < 0.01$ after correction), with medium-to-large effect sizes, and stable confidence intervals that do not overlap with baseline configurations.

6 | Discussion and Conclusions

Throughout this article, we have examined the current trend of optimizing autonomous agents by optimizing reasoning strategies, leaving these agents with a theoretically sufficient set of tools to solve the posed problems; however, later we have seen that completeness does not guarantee optimality. This optimality is clearly reflected in those problems where the action space is infinite, where LLMs present difficulties in selecting the most appropriate tools to solve the problem.

To address this issue, the expansion technique B^* is proposed, derived from a formal analysis of the action space and the cognition function, which has allowed us to improve the performance of both LLMs and LRMs in a complex and sensitive environment, such as Conway's Game of Life. This study aims to show the limitations of purely cognitive strategies and present an alternative in which it is not enough to have sufficient tools, but redundancy, composition, and even the creation of tools that only simplify the interaction with the environment can improve the performance of autonomous agents, even surpassing the performance of LRM based agents.

We believe this approach aligns with the adoption of tools such as MCP/A2A and the creation of more and more tools by autonomous agents, as these will allow agents to substantially improve the performance of general-purpose agents and take a step closer to AGI (Artificial General Intelligence).

7 | Limitations

Despite the observed improvements, the proposed B^* expansion technique introduces several practical limitations.

First, as the expanded action space A^* grows, the cognitive load on the language model increases due to the need to select among a larger set of candidate actions. This may introduce a 'paradox of choice' effect, where overly rich action spaces can degrade decision quality if not properly structured or ranked.

Second, the automatic generation of actions introduces additional computational overhead, as each expansion step requires either LLM inference or decomposition-based synthesis. While this process is amortized over multiple executions, it may still represent a non-trivial cost in real-world deployments.

Third, the effectiveness of B^* may depend on the structure of the task. In highly sequential or syntax-constrained domains (e.g., program synthesis or formal verification), naive action expansion may not yield improvements unless the generated actions preserve strict semantic correctness.

Finally, the current evaluation is limited to a single controlled environment (Conway's Game of Life). While this environment provides a measurable and highly sensitive testbed, it does not cover domains such as API planning, symbolic reasoning or real-world tool orchestration. Future work will extend B^* evaluation to additional benchmarks such as tool-use planning tasks and structured reasoning environments.

8 | Replicability

All the code used for generating the experiments and evaluating the results is available in the GitHub repository: https://github.com/SamthinkGit/Bstar-Expansion. More instructions on replicating the experiments can be found in the repository, which includes the scripts for environment creation and results evaluation.

Author Contributions

Conceptualization: S.A.M.P. Methodology: S.A.M.P. and J.V. Software: S.A.M.P. Validation: S.A.M.P. Formal analysis: S.A.M.P. and J.V. Investigation: S.A.M.P. and J.V. Data curation: S.A.M.P. Writing – review and editing: S.A.M.P. and J.V. Visualization: S.A.M.P. Supervision: J.V. Project administration: J.V. Funding acquisition: J.V. All authors have read and approved the final version of the manuscript.

Funding

The authors have nothing to report.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The datasets generated and/or analysed during the current study are publicly available and can be accessed via the following repository: https://github.com/SamthinkGit/Bstar-Expansion. No restrictions apply to the access or use of these data.

References

Anthropic. 2024a. "Claude 3.5 Models and Computer Use."

Anthropic. 2024b. "Introducing the Model Context Protocol."

Anthropic. 2025. "Introducing Claude 4."

Chase, H., and A. Gola. 2022. "Langchain. Open-Source Framework for LLM Applications."

Chen, J., H. Lin, X. Han, and L. Sun. 2024. "Benchmarking Large Language Models in Retrieval-Augmented Generation." *Proceedings of the AAAI Conference on Artificial Intelligence* 38: 17754–17762.

CrewAI Inc. 2024. "Crewai. Open-Source Python Framework for Building and Orchestrating Multi-Agent AI Systems; Also Offers Enterprise Tools Like Control Plane, Monitoring, and Flows."

DeepSeek, Inc. 2025. "DeepSeek-R1 Release."

Fezari, M., and A. A.-D. Ali-Al-Dahoud. 2023. "From GPT to AutoGPT: A Brief Attention in NLP Processing Using DL." Preprint.

Jiang, Z., F. F. Xu, L. Gao, et al. 2023. "Active Retrieval Augmented Generation." In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 7969–7992. Association for Computational Linguistics.

Jordan, L., and ElevenLabs. 2025. "Introducing ElevenLabs MCP."

LangChain Team. 2024. "Langsmith. Unified Observability and Evals Platform for LLM Apps."

Lifshitz, S., K. Paster, H. Chan, J. Ba, and S. McIlraith. 2024. "Steve-1: A Generative Model for Text-to-Behavior in Minecraft."

Ma, Z., Z. Huang, J. Liu, M. Wang, H. Zhao, and X. Li. 2025. "Automated Creation of Reusable and Diverse Toolsets for Enhancing LLM Reasoning." *Proceedings of the AAAI Conference on Artificial Intelligence* 39: 24821–24830.

Oberhauser, J., and n8n Team. 2019. "n8n. Fair-Code Workflow Automation Platform; Open-Source; Combines Visual No-Code and Custom Code (JS/Python)."

OpenAI. 2023. "GPT-4."

OpenAI. 2024a. "Introducing GPT-4o and GPT-4o Mini."

OpenAI. 2024b. "Presentamos OpenAI o1." https://openai.com/es-419/o1/.

OpenAI. 2025. "Introducing Operator."

Qwen, A. Yang, B. Yang, et al. 2025. "Qwen2.5 Technical Report."

Reworkd. 2023. "AgentGPT."

Sahoo, P., A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha. 2024. "A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications." arXiv Preprint arXiv:2402.07927.

Shi, Z., S. Gao, L. Yan, et al. 2025. "Tool Learning in the Wild: Empowering Language Models as Automatic Tool Agents."

Shojaee, P., I. Mirzadeh, K. Alizadeh, M. Horton, S. Bengio, and M. Farajtabar. 2025. "The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity."

Shridhar, M., X. Yuan, M.-A. Côté, Y. Bisk, A. Trischler, and M. Hausknecht. 2020. "Alfworld: Aligning Text and Embodied Environments for Interactive Learning." arXiv Preprint arXiv:2010.03768.

Singh, A., A. Ehtesham, S. Kumar, and T. T. Khoei. 2025. "A Survey of the Model Context Protocol (MCP): Standardizing Context to Enhance Large Language Models (LLMs)."

Snell, C., J. Lee, K. Xu, and A. Kumar. 2024. "Scaling LLM Test-Time Compute Optimally Can Be More Effective Than Scaling Model Parameters."

Surapaneni, R., M. Jha, M. Vakoc, T. Segal, and Google. 2025. "Announcing the Agent2Agent Protocol (A2A)."

TransformerOptimus. 2023. "SuperAGI."

Wang, G., Y. Xie, Y. Jiang, et al. 2023. "Voyager: An Open-Ended Embodied Agent With Large Language Models."

Wu, A. 2023. "MetaGPT."

Yao, S., D. Yu, J. Zhao, et al. 2023. "Tree of Thoughts: Deliberate Problem Solving With Large Language Models." *Advances in Neural Information Processing Systems* 36: 11809–11822.

Yao, S., J. Zhao, D. Yu, et al. 2023. "React: Synergizing Reasoning and Acting in Language Models." In International Conference on Learning Representations (ICLR).

Yehudai, A., L. Eden, A. Li, et al. 2025. "Survey on Evaluation of LLM-Based Agents."