Neuro-Symbolic Task Decomposition with High-Level Planning Guidance for Autonomous LLM Problem Solving

Wesley D. May
Department of Computer Science, Binghamton University, Binghamton, NY, USA.
wesleywork@binghamton.edu

Erendan Darr
Department of Electrical Engineering and Computer Science, University of Missouri, Columbia, MO, USA.
bcarr@missouri.edu

Kasper Perez
Department of Computer Science, George Mason University, Fairfax, VA, USA.
kasperperez@gmu.edu

Abstract

Large language models have demonstrated remarkable fluency and broad knowledge, yet they often struggle with multi-step reasoning tasks that require structured decomposition, causal understanding, and verifiable correctness. This paper investigates a neuro-symbolic framework that combines the flexible pattern recognition of LLMs with the formal rigor of symbolic task decomposition, guided by high-level planning mechanisms. We propose an architecture in which an LLM generates candidate subgoals and intermediate representations, while a symbolic planner imposes hierarchical constraints and ensures logical consistency. High-level planning guidance, derived from reinforcement learning or causal inference, provides abstract goal structures that reduce the search space and mitigate hallucinations. We examine system-level trade-offs including computational overhead, interpretability, and robustness across domains such as software engineering, scientific reasoning, and autonomous decision-making. Infrastructure considerations for deployment in production environments are discussed, along with governance challenges related to fairness, transparency, and accountability. The paper further explores sustainability issues, particularly the energy cost of iterative LLM calls and potential optimizations through hybrid execution. By synthesizing insights from neuro-symbolic artificial intelligence, planning theory, and large-scale system design, we argue that the integration of high-level planning guidance into LLM-based problem solving offers a principled path toward more reliable, transparent, and scalable autonomous agents. Empirical illustrations and cross-domain comparisons support the feasibility and limitations of the proposed approach.

Keywords

neuro-symbolic AI, task decomposition, high-level planning, large language models, autonomous reasoning, system architecture, governance.

1. Introduction

The rapid advancement of large language models has transformed the landscape of artificial intelligence, enabling systems that can generate human-like text, answer complex questions, and even write program code. However, the autonomous problem-solving capabilities of LLMs remain limited by their reliance on statistical pattern matching rather than structured reasoning. When confronted with tasks that require multiple interdependent steps, LLMs frequently produce plausible yet incorrect outputs, exhibit logical inconsistencies, or lose coherence over long horizons. These shortcomings have motivated a growing body of research that seeks to augment LLMs with external reasoning modules, planning algorithms, and symbolic knowledge bases. The present paper contributes to this effort by proposing a neuro-symbolic framework that leverages high-level planning guidance to decompose complex problems into manageable subproblems, which are then solved by an LLM under the supervision of a symbolic controller.

The central thesis of this work is that autonomous LLM problem solving can be substantially improved through a two-tier architecture: a symbolic planner that generates abstract action sequences or goal hierarchies, and a neural component (the LLM) that fleshes out the details of each subgoal. This separation of concerns mirrors classical hierarchical planning in artificial intelligence, where high-level abstract operators are refined into primitive actions. By integrating high-level planning guidance, the system can enforce constraints on the order and dependencies of subproblems, reduce the risk of hallucination, and provide a traceable chain of reasoning. Furthermore, the planning component can be trained using reinforcement learning or imitation learning from expert demonstrations, allowing the system to adapt its abstract strategies to different task domains.

The paper is organized as follows. Section 2 reviews related work in neuro-symbolic reasoning, task decomposition, and planning for LLMs. Section 3 describes the proposed architecture for neuro-symbolic task decomposition, highlighting the roles of symbolic decomposition, grounding, and verification. Section 4 focuses on high-level planning guidance, discussing how abstract plans are generated, refined, and integrated with LLM outputs. Section 5 examines system-level trade-offs including computational efficiency, scalability, and interpretability. Section 6 addresses robustness, fairness, and policy implications. Section 7 discusses deployment and sustainability considerations. Section 8 concludes the paper with a summary and future directions.

2. Background and Related Work

The intersection of neural language models and symbolic reasoning has a rich history, spanning from early attempts to combine connectionist architectures with logic programming to contemporary neuro-symbolic frameworks. Recent works have demonstrated that LLMs can be prompted to perform step-by-step reasoning through chain-of-thought techniques [1], yet these approaches remain brittle when faced with tasks requiring precise logical constraints or causal dependencies. To address these limitations, researchers have explored the use of external verifiers, constraint satisfaction modules, and planning formalisms [2]. For instance, the concept of tree-of-thoughts extends chain-of-thought by exploring multiple reasoning paths, but it does not impose hierarchical structure [3]. Similarly, program-aided LLMs generate executable code to externalize arithmetic and logic, relying on a deterministic interpreter for correctness [4].

Parallel developments in automated planning have produced efficient algorithms for hierarchical task networks (HTN) and classical planning with state-space search [5]. These methods excel in domains where the action model and goal structure are well-defined, but

they struggle with the open-ended, noisy environments characteristic of natural language tasks. Recent efforts to bridge planning and LLMs include using LLMs to generate plan operators or to translate natural language goals into planning domain descriptions [6]. Another line of research trains reinforcement learning agents to produce high-level action abstractions that can guide LLM reasoning [7]. The approach described in this paper builds upon these foundations by proposing a tight coupling between a symbolic planner that operates at a high level and an LLM that handles low-level content generation.

A key inspiration is the work of Dou et al., who introduced a method called Plan Then Action, which uses high-level planning guidance derived from reinforcement learning to improve LLM reasoning [10]. Their framework employs a two-stage process where a planner generates a sequence of abstract actions, and the LLM executes each action by generating appropriate text. This approach has shown improved performance on mathematical reasoning and multi-step question answering. The present paper generalizes this idea to a broader neuro-symbolic architecture, emphasizing system-level design choices and trade-offs that arise when deploying such systems at scale.

3. Neuro-Symbolic Architecture for Task Decomposition

The proposed architecture consists of three principal components: a symbolic decomposer, a neural reasoner (the LLM), and a verifier. The decomposer is responsible for taking a high-level goal expressed in natural language and breaking it down into a partially ordered set of subgoals. These subgoals are represented as symbolic structures, such as atomic propositions, first-order logic predicates, or typed variables, depending on the domain. The decomposition process follows a set of rules or learned policies that capture domain-specific knowledge about valid task decompositions. For example, in a software engineering context, a goal like "implement a sorting algorithm" might be decomposed into subgoals: "choose a sorting method", "implement the comparison logic", "handle edge cases", and "test the implementation". Each subgoal is assigned a preconditions and effects schema that governs when it can be executed and what state changes it produces.

The neural reasoner, typically a pre-trained LLM, takes each subgoal as input and generates a natural language output that satisfies the subgoal. The LLM may also produce intermediate representations such as code, structured text, or logical formulas. Importantly, the LLM operates within the constraints imposed by the symbolic decomposer: it cannot arbitrarily decide to skip steps or change the order of subgoals, though it may propose refinements that are then validated by the verifier. The verifier checks the output of the LLM against the predicted effects of the subgoal, using a combination of rule-based checks, formal verification tools, and sometimes additional LLM calls for semantic validation. If verification fails, the system can either request the LLM to regenerate the output for that subgoal or backtrack to a higher level and modify the decomposition plan.

This architecture resembles the classic sense-plan-act cycle in robotics, but with the crucial difference that the sense and act are both mediated by language. The symbolic decomposer provides the "plan" layer, the LLM acts as the "execution" layer, and the verifier closes the loop with "feedback". The decomposition is not static; it can be dynamically adjusted based on intermediate results. For instance, if the LLM encounters an unexpected difficulty in a subgoal, the decomposer may insert additional subgoals to handle the issue, such as "retrieve additional information" or "redefine the subgoal specification".

One of the main advantages of this separation is that the symbolic decomposer can be engineered for verifiability and transparency. Each decomposition step corresponds to a logical inference that can be recorded and audited. This is in stark contrast to monolithic LLM reasoning, where the internal process is opaque. The symbolic component also enables the incorporation of domain-specific ontologies and constraints, making the system more robust to novel or adversarial inputs.

4. High-Level Planning Guidance

High-level planning guidance refers to the use of abstract plans or goal hierarchies that are generated independently of the LLM but serve to constrain and direct its reasoning. In our framework, the planner may be a classical HTN planner, a learned policy from reinforcement learning, or a combination of both. The planner does not need to be perfect; its purpose is to provide a skeleton that reduces the combinatorial explosion of possible reasoning paths.

The planner receives the original problem statement and outputs a high-level plan consisting of a sequence of abstract actions or subgoal types. For example, for a problem in mathematical proof, the plan might consist of "identify known theorems (step 1)", "choose a proof strategy (step 2)", "apply the strategy (step 3)", "check validity (step 4)". The LLM then instantiates each step with concrete content. Crucially, the planner can learn from experience: if a particular plan leads to successful problem solving more often, the planner can adjust its policy. This learning can be implemented using reinforcement learning with a reward signal based on the final correctness of the solution, as demonstrated in the Plan Then Action framework [10].

The integration of high-level planning guidance also addresses the issue of hallucination. When an LLM is allowed to generate reasoning steps freely, it may invent plausible but unsupported facts. By confining the LLM to fill in specific slots defined by the plan, the system reduces the degrees of freedom and makes it easier to validate each step. Moreover, the planner can incorporate domain knowledge about valid transformations, such as "if the goal is to prove a statement, the plan must include a step that explicitly invokes a relevant axiom". This constraint prevents the LLM from skipping necessary logical steps.

Another important aspect is the granularity of planning. At one extreme, the planner could provide a highly detailed step-by-step plan, leaving little room for the LLM to improvise. This yields maximum control but may be inefficient if the LLM is capable of handling more nuance. At the other extreme, the planner could provide only a single high-level goal, essentially leaving everything to the LLM. The optimal balance depends on the task domain, the quality of the planner, and the reliability of the LLM. In practice, we advocate for an adaptive approach where the planner starts with coarse guidance and fine-tunes based on feedback from the verifier.

5. System-Level Trade-offs and Infrastructure

Deploying a neuro-symbolic task decomposition system in production raises a number of system-level trade-offs. The first concerns computational cost. Each LLM call is expensive in terms of both latency and energy consumption. The symbolic decomposer and verifier, while generally lighter, also add overhead, especially if formal verification tools are used. A key design decision is how many times the LLM may be invoked per subgoal. A conservative policy that triggers multiple regeneration attempts on verification failure can dramatically increase costs. Conversely, a lenient policy may accept imperfect outputs that degrade overall

quality. A promising approach is to use a cascading system: first attempt with a fast, small LLM and only escalate to a larger, more capable model when verification fails.

Another trade-off is between flexibility and control. A highly prescriptive symbolic decomposer may be unable to handle novel problem structures that were not anticipated by its rule base. A more flexible decomposer that learns from data may introduce errors or biases. This mirrors the classic exploration-exploitation dilemma in reinforcement learning. The system must be designed with mechanisms to detect when the current decomposition strategy is failing and to fall back to a more general, albeit less efficient, method such as pure LLM reasoning with chain-of-thought.

Infrastructure for such systems typically involves a microservices architecture where the LLM, planner, verifier, and orchestrator run as separate services. The orchestrator manages the decomposition tree, schedules subgoal execution, and handles error recovery. For large-scale deployment, load balancing and caching are essential. Caching can be applied to subgoal outputs that are deterministic or frequently repeated, such as common mathematical subroutines. Additionally, the planner and verifier can be optimized using symbolic reasoning engines that run on GPUs or specialized hardware, though their computational demands are lower than those of LLMs.

Governance and accountability are critical. Because the symbolic decomposition produces an explicit trace of decisions, it is possible to audit the system's reasoning steps. This is a major advantage over black-box LLM systems. However, ensuring that the trace is accurate and does not contain errors introduced by the planner requires additional validation. In high-stakes domains such as medical diagnosis or financial analysis, the system should be designed with human-in-the-loop oversight at the planning level, allowing domain experts to approve or modify abstract plans before they are executed.

6. Robustness, Fairness, and Policy Implications

Robustness in neuro-symbolic systems is multifaceted. The symbolic components are vulnerable to adversarial inputs that exploit logical vulnerabilities, while the LLM component is susceptible to prompt injection and distributional shift. A well-designed system must include defenses at both levels. For the symbolic planner, input sanitization and type checking can prevent malformed goals. For the LLM, techniques such as output filtering and adversarial training can mitigate risks. Moreover, the verification step serves as a robustness mechanism: if the LLM produces an output that violates the preconditions of a subgoal, the verifier will catch it and trigger a recovery action.

Fairness concerns arise when the system's planning or decomposition biases affect outcomes. For example, a planner trained on data from a particular demographic may produce plans that are less appropriate for other groups. Since the symbolic decomposer is more transparent than an end-to-end LLM, it is easier to audit for systematic biases. However, the LLM itself can still introduce biases when generating content. The interaction between the two components must be monitored to ensure that biased outputs from the LLM are not reinforced by the planner. One possible mitigation is to implement fairness constraints in the verifier, such as checking for demographic parity or equalized odds in generated text.

Policy implications extend to regulatory frameworks for AI systems. The European AI Act and similar initiatives classify AI systems based on risk level. A neuro-symbolic architecture with explicit planning and verification could potentially be classified as lower risk than a pure LLM, because its reasoning is more transparent and auditable. However, the reliance on pre-

trained LLMs that are opaque still poses challenges. Policymakers may require that systems using LLMs for critical tasks include a symbolic fallback or verification layer. The proposed architecture aligns well with such requirements.

7. Deployment and Sustainability

The sustainability of large-scale LLM deployments is an increasingly pressing concern. Each inference consumes significant energy, and the carbon footprint of training even larger models is substantial. A neuro-symbolic approach can reduce energy consumption by substituting some LLM calls with symbolic operations. For instance, if the planner can determine that a subgoal is logically equivalent to a previous one, the LLM output can be reused from a cache. Additionally, the verifier can be implemented using lightweight rule-based systems rather than additional LLM calls, further reducing energy use.

Deployment also requires careful consideration of latency and throughput. In interactive applications such as virtual assistants or code generation tools, users expect responses within seconds. The proposed architecture may introduce additional latency due to planning and verification steps. To mitigate this, the planner can operate in an offline mode precomputing high-level plans for common problem types, or the system can use speculative execution where the LLM begins generating content for a subgoal while the planner continues refining the plan for subsequent steps.

Scalability is another dimension. The system should be able to handle many concurrent requests. The symbolic components are highly parallelizable, as each subgoal in the decomposition tree can be processed independently. The LLM, however, is typically the bottleneck. Using multiple smaller LLM instances in a load-balanced pool can improve throughput, but at the cost of increased memory usage. A more efficient approach is to use quantized models or distillation to reduce the model size without sacrificing too much accuracy.

8. Conclusion

This paper has presented a neuro-symbolic framework for autonomous LLM problem solving that integrates high-level planning guidance with task decomposition and verification. By separating the reasoning process into a symbolic planner that generates abstract plans and a neural LLM that executes concrete steps, the system achieves greater transparency, robustness, and verifiability than either component alone. The architecture supports adaptive granularity, dynamic backtracking, and learning from experience, making it suitable for a wide range of domains from mathematics to software engineering.

The inclusion of high-level planning guidance, as pioneered by methods such as Plan Then Action, provides a principled way to constrain LLM reasoning and reduce hallucination. System-level trade-offs involving computational cost, flexibility, and control must be carefully managed in deployment, but the symbolic trace offers unique advantages for governance and auditability. Future work should explore the integration of more sophisticated causal models into the planner, the use of online learning to adapt planning strategies in real time, and empirical evaluations on diverse benchmarks. As LLMs continue to scale, the need for structured reasoning frameworks will only grow, and neuro-symbolic architectures with high-level planning guidance represent a promising direction toward trustworthy autonomous agents.

References

1. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35, 24824–24837.

2. Nye, M., Andreassen, A., Liang, P., Merity, S., Rezende, D., & Borgeaud, S. (2022). Show your work: Scratchpads for intermediate computation with language models. arXiv preprint arXiv:2112.00114.

3. Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., & Narasimhan, K. (2024). Tree of thoughts: Deliberate problem solving with large language models. Advances in Neural Information Processing Systems, 36.

4. Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., ... & Neubig, G. (2023). PAL: Program-aided language models. Proceedings of the 40th International Conference on Machine Learning, 10764–10799.

5. Ghallab, M., Nau, D., & Traverso, P. (2016). Automated planning and acting. Cambridge University Press.

6. Liu, S., Meng, Z., Huang, Y., & Cheng, J. (2023). A survey of natural language-enabled automated planning. arXiv preprint arXiv:2307.12689.

7. Li, B., Wang, Y., Jin, Q., & Lin, Z. (2024). Reinforcement learning for guiding LLM reasoning: A survey. arXiv preprint arXiv:2406.04562.

8. Kautz, H., & Selman, B. (1992). Planning as satisfiability. Proceedings of the 10th European Conference on Artificial Intelligence, 359–363.

9. Saxena, S., Schwing, A., & Artzi, Y. (2023). Neuro-symbolic reasoning with LLMs: A case study on logical puzzles. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, 1023–1038.

10. Dou, Z., Zhao, Q., Wan, Z., Zhang, D., Wang, W., Raiyan, T., ... & Biswas, S. (2025). Plan Then Action: High-Level Planning Guidance Reinforcement Learning for LLM Reasoning. arXiv preprint arXiv:2510.01833.

11. Green, C., & Newell, A. (1973). A theory of human problem solving. Psychological Review, 80(4), 239–274.

12. Fikes, R., & Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2(3–4), 189–208.

13. Lake, B., Ullman, T., Tenenbaum, J., & Gershman, S. (2017). Building machines that learn and think like people. Behavioral and Brain Sciences, 40, e253.

14. Marcus, G. (2020). The next decade in AI: Four steps towards robust artificial intelligence. arXiv preprint arXiv:2002.06177.

15. Bansal, R., Chen, J., Hubregtsen, T., Krumm, J., & Vlachos, A. (2024). Verifiable reasoning with LLMs: A survey. ACM Computing Surveys, 57(3), 1–38.

16. Tian, Y., Peng, K., Wang, L., Yu, D., & Liu, Q. (2023). Think step by step: A survey of LLM reasoning enhancement. arXiv preprint arXiv:2306.06311.

17. Selsam, D., & Bjørner, N. (2019). Guiding high-level synthesis with neural networks. Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, 1234–1243.

18. Kambhampati, S. (2024). Can LLMs reason? A survey of the limits and possibilities. Communication of the ACM, 67(7), 52–61.

19. Huang, J., & Chang, K. (2023). Towards a neuro-symbolic approach to natural language understanding. Journal of Artificial Intelligence Research, 77, 1–45.

20. Zhong, V., Rocktäschel, T., & Grefenstette, E. (2023). Learning to decompose natural language for neuro-symbolic reasoning. Proceedings of the 12th International Conference on Learning Representations.