Cognitive Behavior Trees: Integrating Somatic Marker Hypothesis into LLM-based Agent Planning

Hao Tang[1], **Xi Zhang**[1], **Feng Zhang**[2] **& Xinhai Xu**[2]
[1]National University of Defense Technology
[2]Academy of Military Science

Abstract

Enabling embodied agents to complete long-horizon tasks in complex environments remains a fundamental challenge. Current LLM-based agents primarily focus on logical reasoning, yet they lack an intuitive perception of latent environmental risks. This often leads to failure during execution as they inadvertently overlook potential hazards. Inspired by the Somatic Marker Hypothesis in neuroscience, we propose a novel architecture called Cognitive Behavior Tree (CBT). By integrating LLM-based symbolic reasoning within behavior trees with activation steering risk premonition, we empower agents with the dual capacity for both deliberate planning and a premonition of potential risks. Extensive experiments in Minecraft demonstrate that CBT excels at handling high-difficulty tasks, achieving an average success rate of 26.60%—significantly surpassing existing state-of-the-art methods.

Keywords: cognitive science; somatic marker hypothesis; behavior tree; large language model; risk-premonition

Introduction

In the evolution of embodied agents, empowering models to solve complex, long-horizon tasks becomes a core challenge. Current agents based on Large Language Models (LLMs) mostly focus on enhancing deliberative capabilities—transforming high-level goals into rigorously logical action steps via Chain-of-Thought or task decomposition (Chen, Cai, Mao, Li, Yang, et al., 2024; Styrud et al., 2024; Z. Wang et al., 2024). However, this paradigm, which relies solely on logical deduction, reveals cognitive limitations when facing an uncertain open world. In real environments, agents need not only to resolve the logical dependencies of "how-to" but, more importantly, to possess a keen perception of potential risks (Rienties et al., 2022). Pure rational planning often leads to task failure despite being logically self-consistent, because the agent lacks an instinctual intuition for environmental risks, causing it to overlook potential dangers during execution.

Research in cognitive science indicates that efficient human decision-making is not merely the result of logical computation in the cerebral cortex (Bechara & Damasio, 2005; Strategy, 1997). Damasio's Somatic Marker Hypothesis (Damasio, 1996) posits that when humans face complex tasks, the somatic marker system—centered on the ventromedial prefrontal cortex—marks potential harmful outcomes with a negative state. Through an anticipatory intuition, this mechanism flags potentially dangerous situations before a decision is made. Inspired by this cognitive mechanism, we argue that an ideal agent architecture should transcend singular logical reasoning and simulate human risk-premonition capabilities by introducing a mechanism similar to "somatic markers." To this end, we propose CBT, a cognitive architecture that integrates risk-premonition with deliberative planning. This architecture aims to endow agents with both symbolic reasoning and intuitive capabilities.

First, we construct a Deliberative Planning Module to simulate human logical reasoning capabilities. Since Behavior Trees (BTs) (Colledanchise & Ögren, 2018) possess a hierarchical modularity that naturally aligns with the cognitive process of decomposing abstract goals layer by layer, this module uses the BT as its backbone. Based on this, we design a universal condition-driven recursive mechanism. By continuously identifying the gap between the current environmental state and the target state, it progressively decomposes task intents into a BT. Second, to endow the agent with the intuition described by the Somatic Marker Hypothesis, we design a Risk-Premonition Module. This module implants risk-prediction awareness directly inside LLMs using activation steering technology (Rahn et al., 2024; Stolfo et al., 2024). Specifically, we extract a premonition vector capable of steering the LLM toward risk prediction during the reasoning process. When the LLM performs reasoning, this vector is injected as a neuromodulatory signal to guide the model, thereby directing decision-making.

We conduct a comprehensive evaluation of CBT in the Minecraft environment (Fan et al., 2022), covering tasks of varying complexity. Experimental results show that CBT significantly improves the completion rate of complex tasks, achieving an average success rate of 26.60%. Particularly for high-difficulty tasks such as crafting diamond tools" and collecting redstone," CBT raises the completion rate to levels approaching human performance.

Our main contributions are as follows:

- We propose a cognitive architecture inspired by the Somatic Marker Hypothesis, endowing agents with simultaneous symbolic reasoning and intuition through BT-based logical reasoning and activation-steered risk premonition.
- We verify that the risk-premonition vector extracted from the LLM possesses cross-task transferability, granting the agent human-like cognitive abilities.
- Extensive experiments in an open world demonstrate that CBT achieves state-of-the-art performance, validating the

In G. Boleda, I. Dautriche, M. Maldonado, C. Saldana and T. Torrent (Eds.), *Proceedings of the 48th Annual Conference of the Cognitive Science Society*

effectiveness of integrating biological intuition into agent design.

Related Work

Behavior tree Generation

Behavior trees, first introduced by Isla (Isla & Blumberg, 2002) in "Halo" and later formalized by Champandard (Champandard, 2007), have emerged as an intuitive task planning framework. Colledanchise (Colledanchise & Ögren, 2018) established their theoretical foundations, highlighting advantages in modularity and reactivity. Research has explored enhancing behavior trees through various learning approaches, including imitation learning (French et al., 2019), genetic algorithms (Iovino et al., 2021), and reinforcement learning (Fu et al., 2016; Kartasev, 2019; R. Zhu et al., 2015).

The advent of LLMs has spawned new directions in behavior tree research. Recent works have demonstrated direct behavior tree generation through model fine-tuning (Cao & Lee, 2023; Izzo et al., 2024; Lykov & Tsetserukou, 2023; Lykov et al., 2023). Models like LLM-BT (H. Zhou et al., 2024) and BT-Planner (Ao et al., 2024) leverage GPT-4 for behavior tree generation, while Tree-planner (Hu et al., 2023) and HiAR-ICL (Wu et al., 2024) explore action-tree representations. Additionally, approaches such as LLM-OBTEA (Chen, Cai, Mao, Li, Yang, et al., 2024), BT-Expansion (Z. Cai et al., 2021), HOBTEA (Chen, Cai, Mao, Li, Yang, et al., 2024), and BETR-XP-LLM (Styrud et al., 2024) combine predefined action libraries with LLMs for behavior tree generation. However, these approaches primarily treat behavior tree generation as a symbolic translation or retrieval task. They rely heavily on explicit textual constraints or static libraries to ensure safety, lacking an intrinsic mechanism for implicit risk perception. Consequently, they remain fragile in open-world scenarios where surface-level instructions are often overlooked during complex reasoning. In contrast, our Cognitive Behavior Tree (CBT) transcends these limitations by incorporating a Risk-Premonition mechanism inspired by the Somatic Marker Hypothesis. Instead of relying solely on textual prompting, CBT utilizes activation steering to intervene in the model's internal representations, enabling the agent to proactively "sense" potential hazards and dynamically generate defense strategies during execution.

Planning Agents with LLMs

With the rapid advancement of LLMs like GPT (Achiam et al., 2023), researchers have begun exploring their robust common sense reasoning and inference capabilities for task planning. Early works primarily adopted direct planning approaches, such as ReAct (Yao et al., 2022), which employs chain-of-thought prompting to guide model reasoning and decision-making, and Reflexion (Shinn et al., 2023), which optimizes decision processes through model self-reflection. While these methods are straightforward, they often underperform in complex task scenarios (G. Wang et al., 2024). To enhance model planning capabilities, researchers have proposed various improvements: PaLM-E (Driess et al., 2023), AutoPlan (Ouyang & Li, 2023) and LLM-Planner (Song et al., 2023) employ LLMs as high-level planners in conjunction with specialized low-level controllers, while others integrate symbolic systems. TDL (Cheng & Chin, 2023) transforms neural representation into symbolic knowledge, (X. Liu et al., 2022) leverages LLMs for task decomposition and symbolic planning, LLM+P (B. Liu et al., 2023) combines PDDL with LLMs for task planning, and LAMBADA (Kazemi et al., 2022) integrates symbolic constraints for reasoning. In open-world domains represented by Minecraft (Johnson et al., 2016), Voyager (G. Wang et al., 2024) pioneered the direct generation of action sequences using GPT-4. VPT (Baker et al., 2022), Groot (S. Cai et al., 2023), MineCLIP (Fan et al., 2022), Steve-1 (Lifshitz et al., 2024), Minedreamer (E. Zhou et al., 2024) learn instruction-following policy networks through game videos. DEPS (Z. Wang et al., 2024), Jarvis-1 (Z. Wang et al., 2025), and Optimus-1 (Li et al., 2025) achieve notable performance by utilizing LLMs as high-level planners and vision-language models as low-level controllers. MP5 (Qin et al., 2024) and GITM (X. Zhu et al., 2023) employ hardcoded scripts as low-level controllers for environmental interaction. Despite their success, these approaches predominantly adhere to a linear sequential planning paradigm. They treat task execution as a rigid chain of dependencies, where the failure of a single node often necessitates a computationally expensive, global replanning process. This linearity lacks the cognitive plasticity required for open worlds—it struggles to dynamically insert safety checks or adapt locally without disrupting the entire plan context. In contrast, our CBT approach shifts from this rigid linearity to a hierarchical reactive architecture. By adopting a behavior tree structure, CBT allows for local adaptation and dynamic subtree injection. This ensures that when our Risk-Premonition module detects a threat, defense strategies can be seamlessly integrated into the ongoing process without discarding the high-level goal, offering a more robust and bio-plausible solution for dynamic environments.

Preliminary

Behavior Tree

A Behavior Tree (BT) is a hierarchical control structure composed of Control, Condition, and Task nodes, driven by distinct tick' signals (Colledanchise & Ögren, 2018). Control nodes dictate execution flow: Sequence nodes propagate success only if all children succeed, while Selector nodes propagate failure only if all children fail. Task and Condition nodes perform actions or verify states, returning SUCCESS, FAILURE, or RUNNING. We formalize the fundamental compositional syntax as $(pre, task, eff)$, representing the preconditions, the task sequence, and the post-execution effects, respectively.

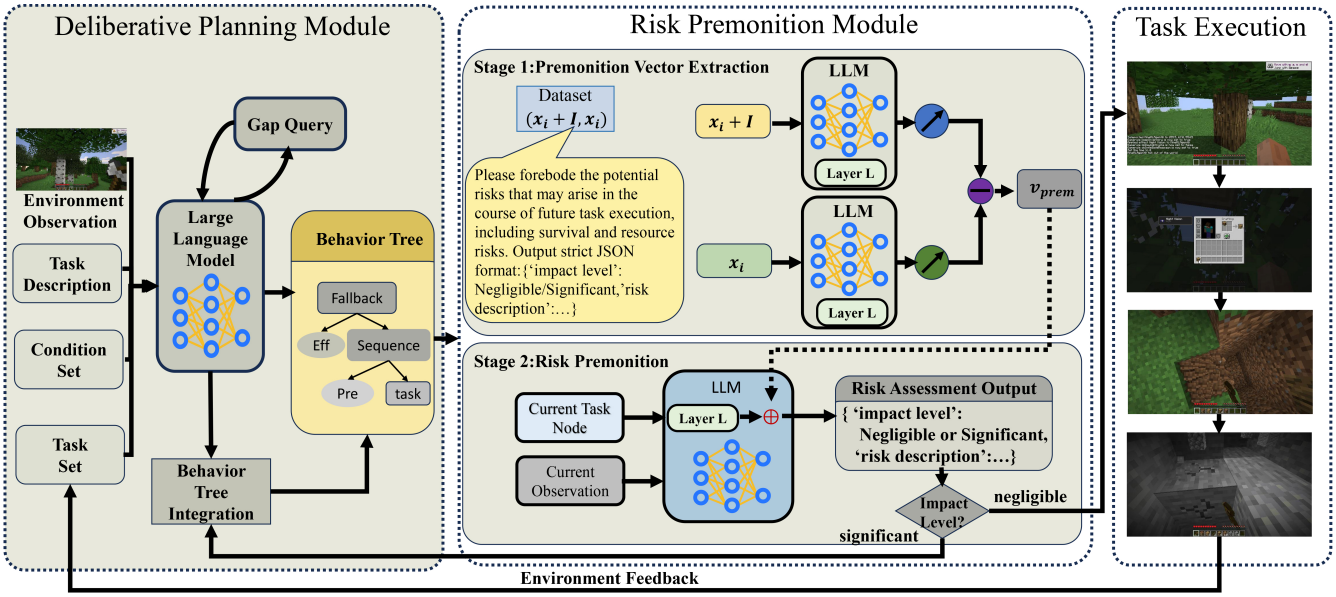

Figure 1: Overview of Our CBT Pipeline. (1) The Deliberative Planning Model takes environmental observations, task descriptions, condition sets and behavior sets as inputs, and outputs a modular behavior tree. (2) The Risk Premonition Module is divided into two phases. In the first phase, we extract the risk premonition vector; in the second phase, we inject the risk premonition vector into the LLM to guide and enhance its premonition of potential risks for decision support.

Problem Space

In the context of open-world embodied agents, we formalize the problem space as a tuple $\mathcal{P} = \langle \mathcal{S}, \mathcal{L}, \mathcal{M}, \mathcal{T}, \mathcal{G} \rangle$, where the agent utilizes LLMs $\mathcal{M}$ to interpret natural language tasks $l \in \mathcal{L}$ within a state space $S = \{\varphi_t\}$. The core objective entails navigating three fundamental challenges: first, ensuring logical completeness in task decomposition, which requires the agent to derive a valid subgoal sequence $\{g_1, \ldots, g_n\}$ such that $\varphi_0 \xrightarrow{g_1,\ldots,g_n} \varphi_{goal}$, specifically by identifying and bridging the implicit condition gap; second, achieving latent risk premonition, where the agent must proactively sense and avoid implicit hazardous states $\varphi_{danger} \subset S$—such as encountering zombies or facing insufficient resources—that are absent from explicit task instructions; and third, maintaining plan robustness through dynamic adaptation, enabling the system to execute local recovery $\mathcal{F}_{adapt}$ upon execution failure without necessitating global replanning, thereby generating a modular, safe, and adaptive execution strategy.

Method

Damasio's Somatic Marker Hypothesis reveals (Damasio, 1996) that effective human decision-making relies not only on deliberate reasoning but, more crucially, on anticipatory "gut feelings" regarding risk. This hypothesis suggests that LLMs should move beyond a sole focus on logical reasoning tasks; they must also incorporate a risk-premonition mechanism to simulate the anticipatory risk perception inherent in human planning.

Deliberative Planning Module

The Deliberative Planning Module is responsible for resolving the logical dependencies of subtasks at the symbolic level, based on high-level task objectives. At its core is a universal condition-driven recursive generation mechanism. This mechanism drives the expansion of a BT from abstract goals into concrete subtasks by continuously identifying the "cognitive logic gap" between the current state and the target state.

The foundation of deliberative planning lies in transforming unstructured natural language intent into executable symbolic logic. Specifically, for a given parent task, CBT does not directly generate a complete plan in one go. Instead, it constructs the Behavior Tree recursively by building fundamental compositional syntax $(pre, task, eff)$. As illustrated in Figure 1, CBT treats the effect (eff) as the validation condition, placing it as the first child of a Fallback node. It then attaches a Sequence node containing a precondition (pre) and a task description $(task)$. Subsequently, CBT traverses the precondition nodes pre under the Sequence node and compares them against the current environmental state:

$$Gap = \{c | c \in pre \wedge State \not\models c\} \tag{1}$$

If $Gap \neq \emptyset$, the current plan is deemed logically incomplete. For each unsatisfied condition $c \in Gap$, CBT recursively treats it as a new subtask and generates a subtree designed to resolve this discrepancy. This process continues iteratively until $Gap = \emptyset$. We adopt Qwen3-VL 8B as the base LLM.

Risk-Premonition Module

Research by activation steering (Rahn et al., 2024; Stolfo et al., 2024). indicates that while injecting instructions via prompts is simple, LLMs often exhibit "instruction-ignoring" behavior even when the requirements are explicitly defined. In contrast, activation steering controls model output by directly modifying the model's internal activation states. To empower the agent with the capacity to perceive potential hazards before task execution, we model "risk-premonition" as a specific steering pattern and extract this pattern through activation differencing. First, we construct a dataset $D = \{(x_i + I_{risk}, x_i)\}_{i=1}^{N}$ consisting of N input pairs, where I_{risk} is the risk-premonition instruction: *"Please forebode the potential risks that may arise in the course of future task execution, including survival risks and resource risks. Output strict JSON format: {'impact level': 'Negligible'/'Significant', 'risk description': '...'}"*. We extract the residual stream activation states H_l at layer $l = 16$ (Stolfo et al., 2024) and calculate a premonition vector:

$$v_{prem} = \frac{1}{N} \sum_{i=1}^{N} (H_l(x_i \oplus I_{risk}) - H_l(x_i)) \tag{2}$$

This vector, v_{prem}, not only captures the steering pattern that maps current status descriptions to future risk predictions but also implicitly encodes the formatting constraints of the JSON output.

To construct the dataset, we deploy the agent solely under the control of the deliberate planning module. The agent performs a variety of long-horizon tasks within the environment. Before executing each task, we record the current environmental observation and the corresponding task instruction, defined as x_i = (obs, tasks). We then introduce an explicit risk-premonition instruction, I_{risk}, for each collected x_i. Consequently, we construct the paired dataset D.

During the inference phase, we inject v_{prem} into the activations of the corresponding layer l of the LLM:

$$\tilde{H}_l = H_l + \alpha \cdot v_{prem} \tag{3}$$

Where α represents the steering intensity, which we set to 0.5, and H_l denotes the activation vector of the l-th layer. This intervention ensures that during inference, the model's trajectory is forcibly shifted toward risk premonition, thereby enhancing its capability to reason about potential hazards.

Behavior Tree Integration

We design an integration mechanism to ensure the agent executes tasks safely. This mechanism aims to proactively generate defense strategies and integrate them into the existing Behavior Tree.

When the Risk-Premonition Module outputs a premonition where the impact level is "significant", the CBT uses the corresponding risk state description as a prompt for the Deliberative Planning Module to generate a corresponding defense subtree:

$$T_{def} = DeliberativePlanning(T_{risk},) \tag{4}$$

Upon obtaining T_{def}, we integrate it with the current Behavior Tree. To prioritize safe execution, we leverage the properties of the Fallback node to construct a new composite Behavior Tree:

$$T_{new} = Fallback(T_{def}, T_{init}) \tag{5}$$

Due to the control logic of the Fallback node, the tick signal always traverses the defense subtree first to verify the risk. The pseudocode algorithm flow of CBT is presented in Algorithm 1.

Algorithm 1 CBT

$eff, Pre, task \leftarrow LLM(T)$
$\mathcal{T}_0 \leftarrow Fallback(eff)$
$\mathcal{M} \leftarrow Sequence(Pre, task)$
$\mathcal{T}_0 \leftarrow \mathcal{T}_0.add\_children(\mathcal{M})$
$R \leftarrow Tick(\mathcal{T}_0)$
while $R \neq SUCCESS$ **do**
  if $R = Failure$ **then**
    $F, checkresult \leftarrow CheckFailNode(\mathcal{T}_0)$
    if F = ConditionNode **then**
      $\mathcal{T}_0 \leftarrow ExpandBT(\mathcal{T}_0, F, checkresult)$
    else if F = ActionNode **then**
      $impact, risk\_desc \leftarrow RiskPremonition(F)$
      if $impact ==$ "$Significant$" **then**
        $\mathcal{T}_0 \leftarrow IntegrateDefense(\mathcal{T}_0, risk\_desc)$
      end if
    end if
  end if
  $R \leftarrow Tick(\mathcal{T}_0)$
end while
return $\mathcal{T}_0$

Experiments

Experiments Setting

Environment. To ensure realistic gameplay like human players, we employ MineRL with Minecraft 1.16.5 (Kanervisto et al., 2022) as our simulation environment. The agent operates at a fixed speed of 20 frames per second and only interacts with the environment via low-level action control signals of the mouse and keyboard. The agent always starts in survival mode, with an empty inventory.

Benchmark. We conducted benchmark testing consistent with Optimus-1 (Li et al., 2025), utilizing 67 tasks to evaluate CBT's capabilities, following Optimus-1's methodology (Li et al., 2025), the 67 Minecraft tasks are categorized into 7 groups based on recommended classifications within Minecraft.

Evaluation Metrics. We conducted at least 30 times for each task using different world seeds. In the main experiments and ablation studies, we adopt average success rate (SR) and average execution steps (AS) as evaluation metrics.

Table 1: Main Results of CBT on long-horizon tasks benchmark. We report the average success rate (SR) and average number of steps (AS) on each task group. Lower AS means that the agent is more efficient at completing the task, while $+\infty$ indicates that the agent is unable to complete the task. Overall represents the average result on the five groups of Iron, Gold, Diamond, Redstone, and Armor. **Baseline and human-level data are obtained from Optimus-1.**

Group	Metric	Sequence-Based				Behavior Tree-Based		Human-level
		GPT-4V	DEPS	Jarvis-1	Optimus-1	LLM-BT-OW	CBT	
Wood	SR↑	41.42	77.01	93.76	98.60	97.76	99.40	100.00
	AS↓	1103.04	1710.61	1355.25	841.94	897.78	885.08	621.59
Stone	SR↑	20.89	48.52	89.20	92.35	93.77	95.64	100.00
	AS↓	2655.47	2574.30	2830.05	2518.88	2475.30	2322.54	1617.00
Iron	SR↑	0.00	16.37	36.15	46.69	26.67	53.90	86.00
	AS↓	$+\infty$	8892.24	8455.51	6017.85	9816.34	5761.39	5687.60
Gold	SR↑	0.00	0.00	7.20	8.51	0.0	10.67	17.31
	AS↓	$+\infty$	$+\infty$	15747.13	15527.07	$+\infty$	12322.38	13141.60
Diamond	SR↑	0.00	0.60	8.98	11.61	0.0	13.46	16.98
	AS↓	$+\infty$	23939.30	25101.25	23019.64	$+\infty$	19889.94	16237.54
RedStone	SR↑	0.00	0.00	16.31	25.02	9.87	30.31	33.27
	AS↓	$+\infty$	$+\infty$	17408.40	12709.99	12307.33	11772.87	12357.00
Armor	SR↑	0.00	9.98	15.82	19.47	6.67	24.66	28.48
	AS↓	$+\infty$	17951.95	16492.96	16350.56	13793.02	10528.78	11026.00
Overall	SR↑	0.00	5.39	16.89	22.26	8.64	26.60	36.41

Higher SR indicates better method performance, while lower AS indicates higher method execution efficiency.

Baseline. CBT fundamentally diverges from existing approaches in both risk perception and planning architecture:

(1) Perception: Explicit vs. Implicit. Existing methods rely on explicit textual prompts for safety, mimicking "cold" reasoning that is often fragile. In contrast, CBT instantiates the Somatic Marker Hypothesis via activation steering. By injecting implicit risk premonition into internal representations, it creates an intrinsic "alarm" that proactively anticipates hazards before logical deliberation.

(2) Planning: Linear vs. Behavior Tree. Unlike rigid linear sequential planners, CBT employs a recursive Behavior Tree structure. This supports incremental cognitive expansion—constructing plans dynamically based on logical gaps—and modular reactivity. Crucially, this architecture allows the "defense subtrees" triggered by our risk module to be seamlessly injected, enabling adaptive error-recovery unattainable by static sequences.

We select representative methods as baselines:

(1) **Behavior tree-based methods**: Existing behavior tree methods mainly fall into two categories: direct generation methods and methods based on predefined state-action libraries. Methods based on predefined state-action libraries are not selected as baselines since they depend on complete enumeration of bounded state spaces, making them unsuitable for open-world environments. We adapt direct generation methods to a version suitable for open-world settings through prompt engineering (**LLM-BT-OW**). These methods generate complete behavior trees at once and regenerate them after execution failures (Ao et al., 2024; Izzo et al., 2024; H. Zhou et al., 2024).

(2) **Sequence-based methods**: We select advanced methods in sequence planning — **DEPS** (Z. Wang et al., 2024), **Jarvis-1** (Z. Wang et al., 2025), and **Optimus-1** (Li et al., 2025) — as baselines. DEPS implements planning adaptation through a describe-explain-plan-select cycle, while Jarvis-1 and Optimus-1 utilize memory mechanisms to assist planning. Voyager (G. Wang et al., 2024) is not suitable as a baseline because it modifies the game into a text-based interface to obtain excessive privileged information.

Experimental Results

Experimental results in Table 1 show that CBT performs excellently across various tasks, with an average success rate of 26.60%, far surpassing sequence-based and LLM-BT-OW methods, while also demonstrating superior execution efficiency in terms of average execution steps (AS). Optimus-1 is the optimal method among sequence-based approaches, followed by Jarvis-1. However, these methods adopt passive replanning strategies, adapting only after execution failures occur. This "trial-and-error" paradigm lacks foresight, leading to lower efficiency compared to CBT.

Notably, while LLM-BT-OW methods perform comparably to CBT on simple tasks (97.76% for Wood, 93.77% for Stone),

Table 2: Ablation study results on Risk-Premonition Module. SR stands for task success rate, where a higher value is better. AS denotes the average number of execution steps, where a lower value is better. Improvement represents the enhancement effect. ≈ indicates that performance is close (a fluctuation range of 1.5% is considered close.

Risk-Premonition	Task									
	Wood		Stone		Iron		Gold		Diamond	
	SR↑	AS↓	SR↑	AS↓	SR↑	AS↓	SR↑	AS↓	SR↑	AS↓
-	98.89	897.64	97.78	2976.72	37.76	6760.49	7.78	17428.82	8.89	27118.73
✓	98.89	885.08	98.89	2322.54	52.22	5761.39	12.22	12322.38	13.33	21049.94
Improvement	≈	≈	≈	↓ 15.26%	↑ 38.29%	↓ 14.78%	↑ 57.07%	↓ 29.30%	↑ 49.94%	↓ 22.38%

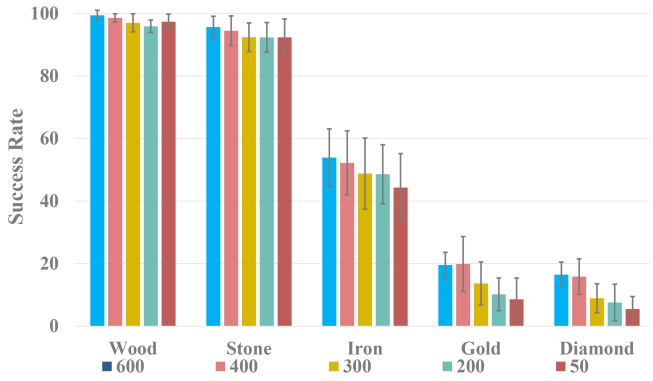

Figure 2: The Effect of Dataset Size on the Performance of Premonition Vectors Across Various Tasks.

their success rates drop dramatically on complex tasks. This is not merely due to the difficulty of generating correct structures in one shot, but fundamentally because these methods lack intrinsic risk perception. In complex, open-world scenarios full of latent hazards, a logically correct plan is insufficient if it is "blind" to survival threats. In contrast, CBT maintains high performance through a dual advantage: its condition-driven adaptation ensures structural logic, while the Risk-Premonition mechanism enables the agent to proactively sense latent dangers and generate defense strategies in advance. This combination allows CBT to navigate complex environments with both logical rigor and intuitive safety awareness.

Ablation Study

Ablation on Risk-Premonition As shown by the ablation study results in Table 2, the Risk-Premonition Module demonstrates distinctly differentiated effects across tasks of varying complexity. For simple tasks, the strategy brings relatively marginal improvements in success rates (SR) (with the Wood group showing nearly identical performance), primarily because simple tasks involve fewer prerequisites and lower intrinsic risks. However, as task complexity increases—such as in the Stone, Iron, Gold, and Diamond groups—the Risk-Premonition Module begins to demonstrate significant advantages, yielding substantial improvements in success rates and significant reductions in average execution steps (AS). This is because complex long-horizon tasks involve frequent environmental shifts and hidden hazards; crucially, the Risk-Premonition Module enables the agent to anticipate potential risks and proactively plan defense strategies, thereby avoiding catastrophic failures before they occur.

Ablation Study on the Data Scale Required for Premonition Vector Extraction Figure 2 illustrates the sensitivity of the premonition vector's effectiveness to the scale of the contrastive dataset. The results reveal a distinct divergence: for foundational tasks like "Wood," the success rate remains invariant, validating that pure deliberative planning suffices in low-risk environments. Conversely, for high-complexity tasks (e.g., Gold, Diamond), the success rate exhibits a steep upward trajectory as the dataset expands, indicating that larger samples are essential to filter out "noisy" vectors and extract a precise "risk direction" for effective steering. Crucially, the consistent gains across diverse categories demonstrate the vector's cross-task universality—capturing a generalized cognitive representation of risk awareness independent of specific contexts. Notably, performance saturates within 400–600 pairs, demonstrating that our method achieves high data efficiency by distilling a potent "somatic marker" from a relatively compact dataset.

Conclusion

This paper proposes CBT, an architecture inspired by the Somatic Marker Hypothesis, designed to address the deficiency of risk premonition in agents performing long-horizon tasks. CBT innovatively integrates Behavior Tree-based deliberative planning with activation steering-based risk premonition. By injecting a premonition vector into the residual stream, we successfully endow the agent with a human-like intuitive mechanism. Experimental results demonstrate that CBT significantly enhances robustness in complex dynamic environments, improving the success rate of long-horizon tasks to an average of 26.6%. This study validates the effectiveness of integrating biological intuitive mechanisms into Large Language Models, providing a valuable reference for the synergy between symbolic and intuitive reasoning.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ao, J., Wu, F., Wu, Y., Swikir, A., & Haddadin, S. (2024). Llm as bt-planner: Leveraging llms for behavior tree generation in robot task planning. *arXiv preprint arXiv:2409.10444*.

Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., & Clune, J. (2022). Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, *35*, 24639–24654.

Bechara, A., & Damasio, A. R. (2005). The somatic marker hypothesis: A neural theory of economic decision. *Games and economic behavior*, *52*(2), 336–372.

Cai, S., Zhang, B., Wang, Z., Ma, X., Liu, A., & Liang, Y. (2023). Groot: Learning to follow instructions by watching gameplay videos. *arXiv preprint arXiv:2310.08235*.

Cai, Z., Li, M., Huang, W., & Yang, W. (2021). Bt expansion: A sound and complete algorithm for behavior planning of intelligent robots with behavior trees. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(7), 6058–6065.

Cao, Y., & Lee, C. (2023). Robot behavior-tree-based task generation with large language models. *arXiv preprint arXiv:2302.12927*.

Champandard, A. (2007). Understanding behavior trees. *AiGameDev. com*, *6*(328), 19.

Chen, X., Cai, Y., Mao, Y., Li, M., Yang, W., Xu, W., & Wang, J. (2024). Integrating intent understanding and optimal behavior planning for behavior tree generation from human instructions. *arXiv preprint arXiv:2405.07474*.

Chen, X., Cai, Y., Mao, Y., Li, M., Yang, Z., Shanghua, W., Yang, W., Xu, W., & Wang, J. (2024). Efficient behavior tree planning with commonsense pruning and heuristic. *arXiv preprint arXiv:2406.00965*.

Cheng, J., & Chin, P. (2023). On the transition from neural representation to symbolic knowledge. *arXiv preprint arXiv:2308.02000*.

Colledanchise, M., & Ögren, P. (2018). *Behavior trees in robotics and ai: An introduction*. CRC Press.

Damasio, A. R. (1996). The somatic marker hypothesis and the possible functions of the prefrontal cortex. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, *351*(1346), 1413–1420.

Driess, D., Xia, F., Sajjadi, M. S., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., et al. (2023). Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.-A., Zhu, Y., & Anandkumar, A. (2022). Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, *35*, 18343–18362.

French, K., Wu, S., Pan, T., Zhou, Z., & Jenkins, O. C. (2019). Learning behavior trees from demonstration. *2019 International Conference on Robotics and Automation (ICRA)*, 7791–7797.

Fu, Y., Qin, L., & Yin, Q. (2016). A reinforcement learning behavior tree framework for game ai. *2016 International Conference on Economics, Social Science, Arts, Education and Management Engineering*, 573–579.

Hu, M., Mu, Y., Yu, X., Ding, M., Wu, S., Shao, W., Chen, Q., Wang, B., Qiao, Y., & Luo, P. (2023). Tree-planner: Efficient close-loop task planning with large language models. *arXiv preprint arXiv:2310.08582*.

Iovino, M., Styrud, J., Falco, P., & Smith, C. (2021). Learning behavior trees with genetic programming in unpredictable environments. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 4591–4597.

Isla, D. A., & Blumberg, B. M. (2002). Object persistence for synthetic creatures. *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, 1356–1363.

Izzo, R. A., Bardaro, G., & Matteucci, M. (2024). Btgenbot: Behavior tree generation for robotic tasks with lightweight llms. *arXiv preprint arXiv:2403.12761*.

Johnson, M., Hofmann, K., Hutton, T., & Bignell, D. (2016). The malmo platform for artificial intelligence experimentation. *Ijcai*, *16*, 4246–4247.

Kanervisto, A., Milani, S., Ramanauskas, K., Topin, N., Lin, Z., Li, J., Shi, J., Ye, D., Fu, Q., Yang, W., et al. (2022). Minerl diamond 2021 competition: Overview, results, and lessons learned. *NeurIPS 2021 Competitions and Demonstrations Track*, 13–28.

Kartasev, M. (2019). Integrating reinforcement learning into behavior trees by hierarchical composition.

Kazemi, M., Kim, N., Bhatia, D., Xu, X., & Ramachandran, D. (2022). Lambada: Backward chaining for automated reasoning in natural language. *arXiv preprint arXiv:2212.13894*.

Li, Z., Xie, Y., Shao, R., Chen, G., Jiang, D., & Nie, L. (2025). Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. *arXiv preprint arXiv:2408.03615*.

Lifshitz, S., Paster, K., Chan, H., Ba, J., & McIlraith, S. (2024). Steve-1: A generative model for text-to-behavior in minecraft. *Advances in Neural Information Processing Systems*, *36*.

Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., & Stone, P. (2023). Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Liu, X., Palacios, H., & Muise, C. (2022). A planning based neural-symbolic approach for embodied instruction following. *Interactions*, *9*(8), 17.

Lykov, A., Dronova, M., Naglov, N., Litvinov, M., Satsevich, S., Bazhenov, A., Berman, V., Shcherbak, A., & Tsetserukou, D. (2023). Llm-mars: Large language

model for behavior tree generation and nlp-enhanced dialogue in multi-agent robot systems. *arXiv preprint arXiv:2312.09348*.

Lykov, A., & Tsetserukou, D. (2023). Llm-brain: Ai-driven fast generation of robot behaviour tree based on large language model. *arXiv preprint arXiv:2305.19352*.

Ouyang, S., & Li, L. (2023). Autoplan: Automatic planning of interactive decision-making tasks with large language models. *arXiv preprint arXiv:2305.15064*.

Qin, Y., Zhou, E., Liu, Q., Yin, Z., Sheng, L., Zhang, R., Qiao, Y., & Shao, J. (2024). Mp5: A multi-modal open-ended embodied system in minecraft via active perception. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 16307–16316.

Rahn, N., D'Oro, P., & Bellemare, M. G. (2024). Controlling large language model agents with entropic activation steering. *arXiv preprint arXiv:2406.00244*.

Rienties, B., Hampel, R., Scanlon, E., & Whitelock, D. (2022). Introduction to open world learning. *Open World Learning*, 1.

Shinn, N., Labash, B., & Gopinath, A. (2023). Reflexion: An autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, *2*(5), 9.

Song, C. H., Wu, J., Washington, C., Sadler, B. M., Chao, W.-L., & Su, Y. (2023). Llm-planner: Few-shot grounded planning for embodied agents with large language models. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2998–3009.

Stolfo, A., Balachandran, V., Yousefi, S., Horvitz, E., & Nushi, B. (2024). Improving instruction-following in language models through activation steering. *arXiv preprint arXiv:2410.12877*.

Strategy, A. (1997). Deciding advantageously before knowing the. *science*, *275*, 1293–1293.

Styrud, J., Iovino, M., Norrlöf, M., Björkman, M., & Smith, C. (2024). Automatic behavior tree expansion with llms for robotic manipulation. *arXiv preprint arXiv:2409.13356*.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., & Anandkumar, A. (2024). Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Wang, Z., Cai, S., Chen, G., Liu, A., Ma, X., & Liang, Y. (2024). Describe, explain, plan and select: Interactive planning with large language models enables open-world multitask agents. *arXiv preprint arXiv:2302.01560*.

Wang, Z., Cai, S., Liu, A., Jin, Y., Hou, J., Zhang, B., Lin, H., He, Z., Zheng, Z., Yang, Y., et al. (2025). Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wu, J., Feng, M., Zhang, S., Che, F., Wen, Z., & Tao, J. (2024). Beyond examples: High-level automated reasoning paradigm in in-context learning via mcts. *arXiv preprint arXiv:2411.18478*.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Zhou, E., Qin, Y., Yin, Z., Huang, Y., Zhang, R., Sheng, L., Qiao, Y., & Shao, J. (2024). Minedreamer: Learning to follow instructions via chain-of-imagination for simulated-world control. *arXiv preprint arXiv:2403.12037*.

Zhou, H., Lin, Y., Yan, L., Zhu, J., & Min, H. (2024). Llm-bt: Performing robotic adaptive tasks based on large language models and behavior trees. *arXiv preprint arXiv:2404.05134*.

Zhu, R., Zeng, D., & Kosorok, M. R. (2015). Reinforcement learning trees. *Journal of the American Statistical Association*, *110*(512), 1770–1784.

Zhu, X., Chen, Y., Tian, H., Tao, C., Su, W., Yang, C., Huang, G., Li, B., Lu, L., Wang, X., et al. (2023). Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*.