CTM-AI: Evolving Multi-Agent Intelligence Inspired by a Model of Consciousness

Haofei Yu[1*] **Yining Zhao**[1*] **Lenore Blum**[2] **Manuel Blum**[2] **Paul Pu Liang**[3]
[1]UIUC [2]CMU [3]MIT
haofeiy2@illinois.edu

Abstract

Despite remarkable advances, today's AI systems remain narrow in scope, falling short of the adaptive and continual intelligence that characterizes human capabilities. This gap has fueled longstanding debates about whether AI might one day achieve human-like generality or even consciousness. Flipping the debate, we present a concrete blueprint for a multi-agent AI system inspired by a formal machine model of consciousness called the Conscious Turing Machine (CTM). Together, CTM-AI contains a large collection of powerful processors ranging from specialized experts (e.g., language models, multimodal models, tools, and APIs) to unspecialized general-purpose learners poised to develop their own expertise. Crucially, for whatever problem must be dealt with, information from many processors is competitively selected, integrated, and exchanged appropriately to solve the task. Feedback mechanisms in CTM-AI enable the system to iteratively evolve its internal connections and communication. CTM-AI achieves state-of-the-art accuracy on MUStARD (72.28) and UR-FUNNY (72.13), outperforming multimodal and multi-agent frameworks. On tool-using and agentic tasks, CTM-AI achieves 10+ points of improvement on StableToolBench and WebArena-Lite. Overall, CTM-AI offers a principled and testable multi-agent AI system inspired by a model of consciousness. The code is available at https://github.com/consciousness-lab/ctm-ai.

1 Introduction

In recent years, progress toward AI models capable of human-like intelligence has inspired debates regarding whether today's AI and its future counterparts can one day display human-like levels of consciousness. Flipping the debate, we present a concrete blueprint for a multi-agent AI system based on a formal machine model of consciousness, the Conscious Turing Machine (CTM) [5, 7]. The CTM is a simple and formal model of consciousness inspired by Alan Turing's model of computation [43] and Bernard Baars' theater model of consciousness [3]. Critically different from other cognitive architectures and multi-agent LLM workflows, the CTM has no central executive – no conductor, no stage director [6]. Instead, the CTM employs a global workspace and distributed competition to integrate the power of an enormous collection of parallel independent cognitive, sensory,

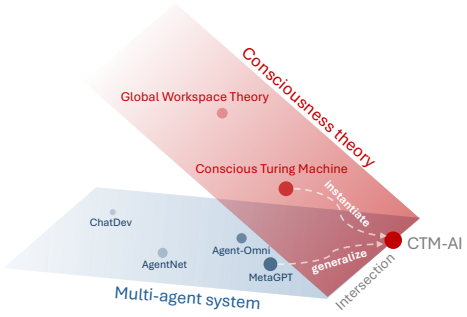


Figure 1: **Positioning of CTM-AI.** Existing research falls into two domains: models of consciousness (red) offer theoretical foundations but lack practical implementations, while multi-agent frameworks (blue) offer practical implementations but lack principled foundations. CTM-AI bridges this gap by *instantiating* the Conscious Turing Machine as a practical AI system.

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

motor, and extended long-term memory processors. When a problem needs to be solved, it becomes globally broadcast to all processors, eliciting help from those who might have the expertise, interest, and resources to tackle the problem, even though their talents and abilities might be unknown to a central executive.

While the CTM provides a theoretical foundation that is fully decentralized, modality agnostic, and architecturally modular, key mechanisms, including how processors broadcast information, form links, and combine expertise, and learn from feedback, are left abstract with no empirical validation. This raises a natural question: *Can the theoretical architecture of the CTM be instantiated into a practically working multi-agent AI system?* To answer this question, our work bridges the gap between theory and practice by instantiating the formal CTM model as a multi-agent framework, CTM-AI, operationalized with modern foundation models. Specifically, CTM-AI features: (1) multiple specialized, agentic processors operating in parallel, (2) a limited-capacity short-term memory workspace that enforces selective attention via up-tree competition, (3) down-tree global broadcasting that distributes information from the workspace to all processors, and (4) iterative evolution through dynamic communication linking between processors, enabling integration of distributed knowledge into higher-order representations.

Key contributions. The significance of CTM-AI is threefold: (1) it serves as the first practical instantiation of the CTM, translating a theoretical model of consciousness into an executable multi-agent AI system; (2) it naturally yields a highly modular and decentralized architecture where processors can be flexibly added or removed, free from the rigid orchestrators of current agentic workflows; and (3) it enables iterative multi-agent evolution through decentralized collaboration and adaptive link formation, expanding reasoning and agentic capabilities beyond static architectures.

Main results. To evaluate CTM-AI's ability to coordinate multiple processors, modalities, and tasks, we present quantitative results across multimodal perception, tool use, and agentic environments. These benchmarks require systems to utilize external APIs, integrate and reason over multimodal data, and solve complex, multi-step problems. CTM-AI achieves state-of-the-art accuracy on MUStARD (72.28%) and UR-FUNNY (72.13%), outperforming unified multimodal baselines and multi-agent frameworks like MoA and MetaGPT. Furthermore, CTM-AI generalizes to tool-use and agentic tasks, yielding an absolute improvement of >10% in pass rate on both StableToolBench and WebArena-Lite. Finally, our analysis demonstrates that: (1) CTM-AI can iteratively self-improve by adapting its processor links based on task complexity; (2) it integrates seamlessly with existing reasoning paradigms; and (3) its core dynamics are robust to hyperparameters and not over-engineered.

2 Related Works

Models of consciousness. Computational models of consciousness seek to formalize how the human brain selects, integrates, and distributes information [8]. Alongside Global Workspace Theory (GWT) [3], several models have been developed with distinct emphases, including Integrated Information Theory [42], Higher-Order Theories [39], and cognitive architectures such as ACT-R [2] and SOAR [22]. Within the GWT lineage, LIDA [16] implements GWT's cognitive cycle symbolically, the Global Neuronal Workspace [30] formalizes it at the neural level, and the Global Latent Workspace [44] proposes a deep-learning roadmap for GWT-style integration. The Conscious Turing Machine (CTM) [5, 7] formalizes GWT in the framework of Turing computation, defining mechanisms including up-tree competition for workspace access and down-tree broadcast [6], but has remained purely theoretical. CTM-AI provides a finite-scale operationalization of the CTM's competition, broadcast, and link-formation mechanisms using modern AI technologies.

Multi-agent frameworks. Most recent multi-agent systems [21, 40] share two structural properties: a central executive that orchestrates information flow (e.g., the manager in MetaGPT), and task-specific workflows where each agent is bound to a predefined role and execution order. These design choices make such systems effective within their target tasks like coding [20, 36] and multimodal understanding [14, 23, 24, 28], but difficult to generalize. CTM-AI differs on both axes: (1) instead of a central executive [20, 36, 40], CTM-AI uses up-tree competition and down-tree broadcast to determine information flow without a learned central orchestrator; (2) instead of fixed workflow [20, 36] or predefined tool-calling protocols [17, 37], processors in CTM-AI have equal priorities is determined dynamically by competition at each iteration. CTM-AI also supports within-episode

adaptation of processor memories and links with iterations, aligning with the recent progress of self-evolving agents [11, 32, 38, 49, 53], but grounded in a cognitive architecture.

3 CTM-AI: The Conscious Turing Machine with Modern AI Models

In this section, we present background on the Conscious Turing Machine (CTM) (§3.1), then describe how CTM-AI instantiates CTM's abstract architecture (§3.2) and dynamics (§3.3) as a concrete and working system.

3.1 Background on the Conscious Turing Machine

The CTM is a simple and formal model of consciousness [5, 7] inspired by Alan Turing's model of computation [43] and Bernard Baars' theater model of consciousness [3]. However, CTM differs from Turing machines and Baars' model in several key ways. While Baars describes consciousness via the activity of actors performing on a stage directed by a stage director, the CTM has no stage director or central executive. Designing a central executive can be prohibitive since we often do not know how such an executive operates. Consider trying to recall the name of a person you've previously met. Although we may recall their name eventually, we do not know which processors are relevant and how to combine processor outputs beforehand. Rather, a federation of processors runs simultaneously, recalling different locations, events, and memories, before deciding which outputs are salient and integrating them to form the final answer. CTM employs a global workspace and distributed competition that determines which information from its vast collection of "unconscious" processors gets admitted to the "conscious" arena. When a problem needs to be solved, it becomes globally broadcast to all processors, eliciting help from those who might have the expertise, interest, and resources to tackle the problem, even though their talents and abilities might be unknown to a central executive. These features set the stage for its capability to be a model for general AI [6].

3.2 CTM-AI Architecture

The formal definition of the CTM is a 7-tuple < STM, LTM, Up-Tree, Down-Tree, Links, Input, Output >. We provide a brief explanation for each of them here:

- CTM is born at time 0 and has a finite lifetime T, measured in discrete clock ticks, $t = 0, 1, 2, ..., T \approx 10^{10}$.
- STM (short-term memory) is a small memory capable of holding a single chunk of information at each time t.
- LTM (long-term memory) is a collection of K powerful processors $\{p_1, p_2, ..., p_K\}$, K can be as large as 10^7.
- Up-Tree is an up-directed binary tree of height h with K leaves, one leaf in each LTM processor, and a (single) root in STM.
- Down-Tree is a simple down-directed tree of height 1 with a single root in STM and K edges directed from that root to the leaves, one leaf in each LTM processor.
- Links are the channels for transmitting information directly between processors.
- Input: $\mathbb{R}^d \rightarrow$ LTM carries information from the external (outer) world via sensors (*e.g.*, eyes, ears) to special LTM processors (*e.g.*, visual and auditory processors). $\mathbb{R}^d$ is CTM's external world where $\mathbb{R}$ represents the real numbers and d is a positive integer. It also includes a user intent, like a query about the external world.
- Output: LTM $\rightarrow \mathbb{R}^d$ carries information from special processors (*e.g.*, motor processor) that can be considered as feedback to the external (outer) world.

Long-term memory processors. The CTM-AI implementation comprises a configurable pool of LTM processors, each with its own expertise and memory. There are five broad families of LTM processors [9]:

- Sensory processors convert raw perceptual signals (e.g., vision, language) into representations.
- Extended or artificial processors wrap external tools and APIs (e.g., calculators, web search, weather services) so that they can be accessed as internal modules.
- Cognitive processors handle reasoning, inference, and planning for long-horizon problem solving.
- Motor processors generate outputs by mapping internal intents to external actions, including dialogue utterances, API calls, or embodied movements.

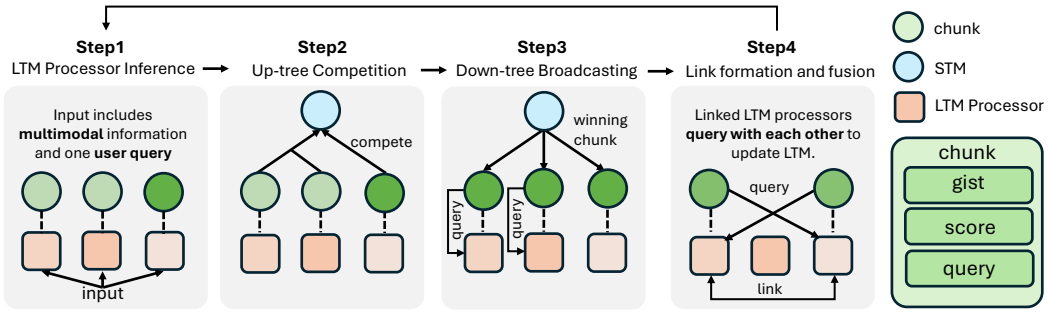


Figure 2: **Overview of CTM-AIdynamics**. (**1**) all specialized LTM processors run in parallel, each producing a chunk with a content gist, follow-up queries, and a self-assessed score; (**2**) an up-tree competition selects which chunk enters the limited-capacity STM, determining the system's conscious content; (**3**) A down-tree broadcast distributes this content to all processors, at which point the system becomes consciously aware of the selected information; (**4**) bidirectional links form between processors that hold complementary information, enabling direct unconscious communication that bypasses the STM, followed by fusion where linked processors exchange to enrich their memories. CTM-AI iterates with memories and links evolving across iterations.

- Unspecialized "free" processors are expandable slots for future role discovery.

Formally, an LTM processor p (with parameters θ) operates in a shared space $\mathcal{H}$ and maintains a private memory state $M_t \in \mathcal{M}$ updated over time. Such a memory works as the context for the processor. At step t, it receives an observation $o_t \in \mathcal{O}$ and a user query $q_t \in \mathcal{Q}$. We view the LTM processor at time t as a function $\mathrm{LTM}_t(\cdot)$ equipped with three operations: (1) *execute* produces a chunk based on the current observations and previous memory; (2) *read* returns a view of its memory at a specified timestamp; and (3) *write* integrates one or more chunks into its memory:

$$\textbf{execute: } \mathrm{LTM}_t(o_t, q_t) = \mathbf{c}_t \quad \textbf{read: } \mathrm{LTM}_t(\cdot) = \mathrm{M}_t \quad \textbf{write: } \mathrm{LTM}_t(\mathbf{c}_t) = \mathrm{M}_t \oplus \mathbf{c}_t = \mathrm{LTM}_{t+1}(\cdot) \tag{1}$$

A chunk $\mathbf{c}_t$ produced by processor p at step t is defined as a tuple: $\mathbf{c}_t = \langle \mathrm{addr}(p),\, t,\, h_t,\, q_t,\, s_t \rangle$. Each chunk stores its unique identifier $\mathrm{addr}(p)$, the timestep t, a gist $h_t \in \mathcal{H}$ in English language that summarizes information relevant to the user's query (e.g., information from audio like laughter detected and likely humorous), one or more follow-up query $q_t \in \mathcal{Q}$ that the processor proposes to other processors if answering it could improve the final answer (e.g., a language processor can ask the vision processor for facial expressions), and a self-reported score s_t indicating the processor's confidence/utility for how useful the gist is to answer the query.

Short-term memory. Short-term memory (STM) is a small memory that holds a single chunk of information at each time step t. After up-tree competition, the winning chunk $\mathbf{c}_t^{i^\star}$ is stored in the STM. If its score $s_t^{i^\star}$ exceeds a threshold τ, the chunk is considered as the conscious output of CTM-AI and sent as the output of the CTM-AI; otherwise, the STM would be broadcast to all LTM processors via down-tree broadcast, and the system proceeds to the next iteration to gather more information.

3.3 CTM-AI Dynamics

Based on this architecture, the following learning dynamics govern prediction and learning in CTM:

1. Different LTM processors perform distinct functions, *e.g.*, cognitive, sensory, or motor. Some processors may be "off-the-shelf" while others' functionalities are realized over time. While individual processors may have their own internal language, communication within the CTM is in a common multimodal language we call Brainish. All processors start as independent entities.
2. Conscious communication between processors is conducted via an Up-Tree competition that decides whose chunk of information gets into STM.
3. The winning chunk (CTM's conscious content) is immediately globally broadcast to all processors via the Down-Tree, which causes the CTM to pay conscious attention to this information.
4. Links between processors form over time as one processor views another as having relevant information, enabling unconscious communication to integrate their knowledge into higher-order information (*e.g.*, learning to ride a bike requires conscious communication between sight and movement, after a while, links form, enabling unconscious communication).

5. Through continuous interaction, feedback, and learning from its external world via sensory inputs, predictions, actuators, and feedback, the CTM updates its individual processors, processor links, and multiprocessor integration to improve over time.

To implement these learning dynamics using modern AI architectures, we translate the first four CTM principles into a four-stage inference process for CTM-AI. The final principle dictates the overarching iteration loop. A formal description of each computational mechanism follows:

▷ **Step 1: LTM processor chunk inference**. At each iteration t, all K LTM processors run in parallel on the observation o_t and query q_t, each conditioned on its private memory M_t^i. Every processor jointly produces three outputs: a content gist h_t^i summarizing its findings, a set of follow-up queries q_t^i for potential cross-processor consultation, and a self-assessed score s_t^i. The resulting chunk $\mathbf{c}_t^i$ is:

$$\mathrm{CTM}_{\mathrm{collect}}(o_t, q_t) = \left\{\mathrm{LTM}_t^i(o_t, q_t)\right\}_{i=1}^K = \{\mathbf{c}_t^i\}_{i=1}^K \tag{2}$$

We use $\mathrm{score}(\mathbf{c}_t^i) = s_t^i$ and $\mathrm{addr}(\mathbf{c}_t^i) = i$ to access the score and processor index stored in a chunk.

Chunk score calculation. Here we provide more information about the score function. Directly inspired by the CTM, the score s_t^i is decomposed into three interpretable sub-scores: weight (how relevant the chunk addresses the query), intensity (the processor's confidence in its output), and mood (whether the chunk contains unexpected information) as an interpretable heuristic. These sub-scores are elicited alongside the chunk's gist via structured prompting. The final score is computed as a linear combination:

$$s_t^i = \alpha_1 s_{\mathrm{weight},t}^i + \alpha_2 s_{\mathrm{intensity},t}^i + \alpha_3 s_{\mathrm{mood},t}^i \tag{3}$$

where we set $\alpha_1 = \alpha_2 = 1$ and $\alpha_3 = 0.2$, down-weighting mood to prioritize reliable and on-topic chunks in the subsequent up-tree competition.

▷ **Step 2: Up-tree competition into STM**. After collecting chunks from all LTM processors, an up-tree competition selects the final chunk for STM's limited-capacity workspace. This competition is hierarchical and local: sibling chunks compete in groups via an additive competition function, ensuring the winning probability is independent of a processor's tree position. Theoretically, there can be a very large number of processors. In practice, only a few (< 10) LTM processors are typically active during inference, so we adopt a simplified global competition that samples an STM entry from the chunk scores. Scores are normalized into a categorical distribution (with temperature β). Here $\mathbf{s}_t = (s_t^1, \ldots, s_t^K)$ contains the self-reported scores already stored in the collected chunks.

$$i^\star \sim \mathrm{Cat}\big(\mathrm{Softmax}(\mathbf{s}_t/\beta)\big), \quad \mathrm{CTM}_{\mathrm{up}}(\{\mathbf{c}_t^i\}_{i=1}^K) = \mathbf{c}_t^{i^\star}. \tag{4}$$

▷ **Step 3: Down-tree broadcast**. Once the up-tree competition selects the winning chunk $\mathbf{c}_t^{i^\star}$, it is written into the STM. If the stopping criterion is not met, the winning chunk is broadcast to all LTM processors via the down-tree. The system becomes consciously aware of this information upon reception by all processors [5]. Operationally, each processor receives the broadcast and applies its own **write** function to update its private memory:

$$\mathrm{CTM}_{\mathrm{down}}(\mathbf{c}_t^{i^\star}) = \left\{\mathrm{LTM}_t^i(\mathbf{c}_t^{i^\star})\right\}_{i=1}^K = \left\{\mathrm{LTM}_{t+1}^i(\cdot)\right\}_{i=1}^K \tag{5}$$

After this step, all processors share the same conscious content in their updated memories, preparing the system for cross-processor integration in Step 4.

▷ **Step 4: Link formation and fusion**. While steps 1–3 represent *conscious* communication in the CTM through STM broadcasts, step 4 enables *unconscious* communication where LTM processors form links and exchange information with each other [5].

Link formation. Triggered by the down-tree broadcast in Step 3, the winning processor $i^\star$ forms bidirectional links with other processors based on their previously reported chunk scores. Specifically, a link is established between processors $i^\star$ and j if $\mathrm{score}(\mathbf{c}_t^j) > \eta$, where η is the link threshold. The corresponding entries in the adjacency matrix are updated as $L[i^\star, j], L[j, i^\star] \leftarrow 1$. Since these scores are already generated during chunk inference, link formation requires no additional LLM calls or relevance evaluations. For example, in sarcasm detection, vision, text, and audio processors detect distinct cues (a sad face, an angry tone, exaggerated speech) and form links to share complementary evidence. Links formed for a given datapoint remain permanent across iterations.

Link fusion. Once links are established, each processor LTM_i consults its linked neighbors $\mathcal{N}(i)$ in parallel. It uses the follow-up queries q_t^i generated in Step 1 to consult its linked neighbors; the neighbors respond via their **execute** function, and the initiating processor integrates these responses via **write**:

$$\mathrm{CTM}_{\mathrm{fuse}}(o,\, L)=\big\{\, \widehat{\mathrm{LTM}}_t^{\,i} \,\big\}_{i=1}^{K}=\big\{\mathrm{LTM}_{t+1}^{i}(\cdot)\big\}_{i=1}^{K},\;\; \widehat{\mathrm{LTM}}_t^{\,i}=\mathrm{LTM}_t^{i}\Big(\big\{\mathrm{LTM}_t^{j}(o,\, q^j)\big\}_{j\in\mathcal{N}(i)}\Big) \quad (6)$$

This unconscious cross-processor integration discovers richer, synergistic information that no single processor could produce alone [27, 35]. The enriched long-term memories are carried into the next iteration of Step 1, closing the inference loop.

▷ **Overall: Iterative inference loop**. The CTM theory prescribes a continuous cycle of *prediction, feedback, and learning* [5]. CTM-AI keeps this structure through Algorithm 1.

Prediction. All processors produce chunks from observations and memory, then compete via the up-tree to select STM content.

Feedback. For agentic tasks (e.g., web navigation tasks), motor processors translate the conscious content into actions on the external environment. The environment's response returns as a new observation o_{t+1}, providing feedback that informs the next step. For non-agentic tasks (e.g., multimodal perception), no external feedback is available, the system instead relies on iterative internal refinement.

Algorithm 1 CTM-AI Iterative Inference Algorithm

Require: LTMs $\{\mathrm{LTM}^k\}_{k=1}^{K}$; STM; link matrix $L \in \{0,1\}^{K\times K}$; query q; obs o; max rounds R; thresholds τ, η
Ensure: Output STM for answering query q

for $r = 1, \ldots, R$ **do**
  $\{\mathbf{c}^i\}_{i=1}^{K} \leftarrow \mathrm{CTM}_{\mathrm{collect}}(o, q)$ ▷ Eq. (2)
  $\mathbf{c}^\star \leftarrow \mathrm{CTM}_{\mathrm{up}}(\{\mathbf{c}^i\}_{i=1}^{K})$ ▷ Eq. (4)
  $\mathrm{STM} \leftarrow \mathbf{c}^\star$
  if $\mathrm{score}(\mathbf{c}^\star) > \tau$ **then**
    return STM
  end if
  $\{\mathrm{LTM}^i\}_{i=1}^{K} \leftarrow \mathrm{CTM}_{\mathrm{down}}(\mathbf{c}^\star)$ ▷ Eq. (5)
  $i^\star \leftarrow \mathrm{addr}(\mathbf{c}^\star)$
  $L[i^\star, j], L[j, i^\star] \leftarrow 1, \forall j \neq i^\star : \mathrm{score}(\mathbf{c}^j) > \eta$
  $\{\mathrm{LTM}^i\}_{i=1}^{K} \leftarrow \mathrm{CTM}_{\mathrm{fuse}}(o, L)$ ▷ Eq. (6)
end for
return STM

Learning. In the original CTM, learning is realized through the Sleeping Experts Algorithm, which adjusts processor weights based on prediction outcomes. To achieve multi-agent evolution, CTM-AI instead leverages in-context learning for self-reported score updates, requiring no parameter updates, through two evolution mechanisms: (1) *memory evolution*: broadcast chunks and fused responses are written into each processor's private memory, enriching the context window for future inference; and (2) *structural evolution*: new links form between processors with complementary information, progressively densifying the communication graph for richer information integration.

4 Evaluating the Capabilities of CTM-AI

We present quantitative results for CTM-AI on multimodal perception (MUSTARD [10] and UR-FUNNY [19]), tool use (STABLETOOLBENCH [17]), and web agent tasks (WEBARENA-LITE [52]).

4.1 Evaluation Settings

Unified model baselines. We compare against strong unified single-model systems across all three evaluation axes. Here, "unified" means that a single model models multimodal interaction internally or manages multi-step tool use within one end-to-end system, without explicit decomposition into multiple collaborating agents.

For *multimodal perception*, we include two types of baselines: (1) fine-tuned multimodal models, including MMoE [51], BLIP-2 [26], and ALBEF [25], which are trained to jointly model text and visual signals for multimodal interactions; and (2) prompting-based multimodal foundation models, including Qwen3-VL-8B-Instruct [4], Qwen3-VL-8B-Thinking [4], Qwen3-Omni-Flash [48], and Gemini-2.5-Flash-Lite [13].

For *tool-use*, we compare with strong LLM-based agents, including GPT-4o [1] and ToolLLaMA-v2 [37] under standard prompting strategies such as Chain-of-Thought and DFS-style planning. These baselines rely on a single LLM to plan and compose tool calls across multi-step tasks.

For *web agent tasks*, we adopt ReAct [50] with GPT-4o and Gemini-2.5-Flash-Lite as base models. Both the ReAct baselines and the CTM-AI-based agent receive identical observations.

Table 1: **Evaluation results on multimodal perceptions (MUStARD and UR-FUNNY)**. We report accuracy and macro-F1. Fine-tuned models and Qwen3-VL-series take vision and text as inputs. Qwen3-Omni and Gemini-2.5-flash-lite take vision, text, and audio. Details in Appendix §E.

	MUStARD		UR-FUNNY	
Method	Acc↑	F1↑	Acc↑	F1↑
Fine-tuned models				
ALBEF	54.49	48.51	66.77	68.67
BLIP2	53.75	62.65	70.43	74.31
MMoE	70.41	71.78	71.88	73.29
Prompting-based models				
Qwen3-VL-8B-Instruct	51.69	47.55	62.70	60.00
Qwen3-VL-flash	52.53	47.81	64.92	63.27
Qwen3-VL-8B-thinking	57.58	55.61	67.84	67.60
Qwen3-Omni-flash	54.78	50.47	70.16	69.62
+ CTM-AI	66.76	66.75	66.57	65.98
Gemini-2.5-fl	71.91	71.71	70.70	70.77
+ CTM-AI	72.28	72.23	72.13	72.13

Table 2: **Evaluation results on tool-using (StableToolBench).** We report the solvable pass rate score evaluated with GPT-4o (MirrorAPI-Cache setting) . We focus on multi-tool calling scenarios. I2-Cat. stands for I2-Category, I2/I3-Inst. stands for I2/I3-Instruction. Details in Appendix §E.

	StableToolBench		
Method	I2-Cat.↑	I2-Inst.↑	I3-Inst.↑
ToolLLaMAv2 CoT	$19.9_{\pm 1.0}$	$22.3_{\pm 0.4}$	$19.1_{\pm 0.8}$
ToolLLaMAv2 DFS	$22.8_{\pm 1.5}$	$19.2_{\pm 1.6}$	$18.6_{\pm 1.5}$
GPT-4o mini CoT	$24.5_{\pm 1.0}$	$22.3_{\pm 2.7}$	$20.8_{\pm 1.5}$
GPT-4o mini DFS	$25.8_{\pm 1.7}$	$25.8_{\pm 2.7}$	$20.2_{\pm 0.8}$
GPT-4o CoT	$32.5_{\pm 1.7}$	$29.6_{\pm 1.6}$	$27.9_{\pm 3.5}$
GPT-4o DFS	$32.8_{\pm 1.5}$	$28.3_{\pm 1.3}$	$23.0_{\pm 1.3}$
Gemini-2.5-fl CoT	$23.1_{\pm 2.1}$	$12.7_{\pm 0.6}$	$19.1_{\pm 0.8}$
+ CTM-AI	$47.4_{\pm 2.5}$	$34.9_{\pm 1.9}$	$46.7_{\pm 2.0}$
Qwen3-8B CoT	$29.8_{\pm 2.4}$	$26.2_{\pm 3.0}$	$33.6_{\pm 1.2}$
+ CTM-AI	$52.1_{\pm 1.9}$	$53.4_{\pm 0.7}$	$72.7_{\pm 1.0}$
Qwen3-8B-think CoT	$39.7_{\pm 0.9}$	$37.0_{\pm 0.7}$	$39.1_{\pm 0.8}$
+ CTM-AI	$56.5_{\pm 0.8}$	$43.1_{\pm 0.6}$	$66.4_{\pm 0.7}$

Multi-agent baselines. To situate CTM-AI among existing multi-agent paradigms at comparable inference cost, we further compare with several representative frameworks: multi-agent debate [15], centralized orchestra [41], multi-agent ensembling [46], MetaGPT [20], Mixture-of-Agents (MoA) [45], and AutoGen [31]. We instantiate all multi-agent baselines on the same backbone as CTM-AI for fair comparison, with each framework assigning dedicated agents to audio, video, text modalities, and tool use. Details are provided in Appendix §E.

CTM-AI backbone models. For multimodal interaction, we instantiate processors in CTM-AI with Gemini-2.5-Flash-Lite and Qwen3-Omni-Flash, both of which natively accept text, audio, and vision inputs, and initialize each processor with the same underlying model so that comparisons against unified model baselines reflect architectural gains rather than differences in model capacity. For tool use, we additionally evaluate with Qwen3-8B-Instruct and Qwen3-8B-Thinking backbones under the same principle. We also use Gemini-2.5-Flash-Lite as the backbone model for agentic tasks. Details of per-task backbone assignments are provided in Appendix §E.

4.2 Evaluation Results

CTM-AI consistently outperforms baselines across multimodal, tool-using, and agentic benchmarks. CTM-AI consistently outperforms same-backbone and prompting-based baselines across all three evaluation axes. As shown in Table 1, on multimodal perception tasks, we find that CTM-AI with Gemini-2.5-Flash-Lite achieves 72.28 F1 on MUStARD and 72.13 F1 on UR-FUNNY, matching or outperforming the strongest baselines. As shown in Table 2, on StableToolBench, we find that CTM-AI yields the most substantial gains, improving over the strongest single-model baseline (Qwen3-8B-thinking CoT) by up to +16.8 points on multi-tool scenarios. Similarly, as shown in Figure 3, on WebArena-Lite, we find that CTM-AI improves over the ReAct baseline with the same backbone model across all five website categories. These results demonstrate that our proposed single decentralized multi-agent coordination mechanism effectively generalizes across distinct task families, from multimodal fusion in perception to multi-step planning in tool execution.

CTM-AI achieves better performance compared with unified cross-modal reasoning by coordinating multi-agents. In our multimodal setting, each processor directly observes only a single modality observation, rather than the complete multimodal input. Nevertheless, CTM-AI achieves competitive or even superior performance to unified multimodal models through agentic coordination, without additional cross-modal joint training. Through competition, broadcast, and fusion, specialized processors exchange information to integrate complementary evidence across modalities. As shown in Table 1 and Table 2, CTM-AI outperforms unified baselines under the same Gemini-2.5-Flash-Lite backbone, achieving +0.5 F1 on MUStARD, +1.4 F1 on UR-FUNNY, and over +20 points across all StableToolBench splits. These improvements largely extend to Qwen3-Omni-Flash and Qwen3-8B variants. A notable exception is UR-FUNNY with Qwen3-Omni-Flash, where positive-class predic-

Table 3: **Comparison with multi-agent frameworks.** We compare methods on macro-F1, wall-clock time per instance (Δt), and number of API calls per instance across MUStARD and UR-FUNNY. All methods use Gemini-2.5-Flash-Lite as the backbone. R denotes the maximum number of inference rounds. All agents inside share the same prompt for inference. We further extend the same-backbone comparison to StableToolBench and WebArena-Lite (lower panel), reporting pass/success rate (%), latency (Δt, s), model calls (#API), and external interactions (#Tool: tool calls; #Act: environment actions), under a separate evaluation setting from Table 2. Details are in Appendix §E.

	Metric	Unified	Ensemble	Orchestra	Debate	MetaGPT	AutoGEN	MoA	CTM-AI (R=3)
MUStARD	F1↑	71.7	69.3	55.6	63.8	66.1	71.1	72.0	72.2
	Δt ↓	7.2	5.6	23.6	16.3	20.9	21.4	17.0	18.7
	#API↓	1.0	3.0	13.0	10.0	6.1	10.1	10.1	9.7
UR-FUNNY	F1↑	70.8	58.7	54.6	66.1	62.1	58.6	67.6	72.1
	Δt ↓	8.8	5.4	24.2	17.9	27.0	37.1	36.7	42.8
	#API↓	1.0	3.0	13.0	10.0	6.1	9.9	10.2	17.0
	Metric	ReAct	Ensemble	Orchestra	Debate	MetaGPT	AutoGEN	MoA	CTM-AI (R=3)
StableToolBench	Pass↑	32.9	38.1	56.5	38.0	46.3	44.0	58.3	62.9
	Δt ↓	85	160	171	316	102	98.9	356	186
	#API↓	5.2	16.5	14.7	32.4	6.4	9.4	19.4	12.5
	#Tool↓	3.6	10.8	7.6	22.0	3.9	5.6	10.7	10.6
WebArena-Lite	SR↑	12.1	20.7	16.4	17.9	16.4	17.9	20.0	25.1
	Δt ↓	122	118	128	152	112	145	130	106
	#API↓	7.7	28.6	38.4	49.4	27.4	47.9	31.3	38.2
	#Act↓	7.7	7.2	7.7	7.0	9.2	6.8	7.8	5.2

tions decrease from 64.0% to 36.9%, improving precision (0.652 to 0.715) but reducing recall (0.844 to 0.535) and macro-F1 (0.696 to 0.659).

CTM-AI outperforms other multi-agent frameworks with adaptive cost trade-offs. CTM-AI overcomes the typical accuracy-flexibility trade-off of modular multi-agent pipelines, outperforming six representative multi-agent frameworks under the same backbone. As shown in Table 3, we find that CTM-AI achieves the highest F1 on both MUStARD (72.23) and UR-FUNNY (72.13), outperforming the strongest multi-agent baseline (MoA) by +0.2 and +4.6 F1, respectively. With only two inference rounds ($R = 2$), CTM-AI achieves F1 scores of 69.8 and 70.0 on MUStARD and UR-FUNNY, respectively, using 6.9 and 9.4 API calls per instance. On StableToolBench and WebArena-Lite, CTM-AI ($R = 3$) further achieves a 62.9% pass rate and a 25.1% success rate, outperforming the strongest baselines by 4.6 and 4.4 percentage points, respectively. These gains also come at competitive inference costs: CTM-AI uses fewer API calls than MoA on StableToolBench (12.5 vs. 19.4) and achieves the lowest latency on WebArena-Lite (106 seconds). These results highlight two key insights. First, CTM-AI's dynamic coordination is particularly beneficial when contextual cues are sparse, as in UR-FUNNY (+4.6 F1 over MoA versus +0.2 on MUStARD). Second, iterative inference enables an adaptive accuracy–cost trade-off, allowing easier instances to stop early while harder ones receive additional refinement.

4.3 Discussions

Q1: Can CTM-AI keep improving its performance when the number of processors keeps increasing?

We vary the processor pool from K=1 to K=32 on StableToolBench while keeping everything else fixed. The macro-average of the pass rate rises steadily from 41.8 to 62.3, but with diminishing returns: the largest gain comes from K=2 to 4, while 16 to 32 adds only 1.0. Meanwhile, model calls nearly double with each doubling of K (5.6 → 108.0), latency grows from 67 to 427, and active links increase from 0 to 2.3. Notably, K=16 already reaches 98% of the best score with about half the model calls of K=32, offering a favorable accuracy–cost trade-off.

Table 4: **Scalability of processor number** K. We report the solvable pass rate on StableToolBench (I2-Cat., I2-Inst., I3-Inst.), model calls (#API), latency (Δt, s), and average active links (#Links) for different K.

	Pass rate↑					
K	I2-Cat.	I2-Inst.	I3-Inst.	#API	Δt	#Links
1	35.1	34.2	56.2	5.6	67	0.00
2	46.6	44.4	46.3	10.5	111	0.20
4	45.8	48.7	70.2	18.1	141	0.54
8	51.6	53.2	63.9	31.7	168	0.89
16	61.2	54.0	68.6	58.2	285	1.58
32	55.7	53.4	77.9	108.0	427	2.33

Q2: Without external feedback, why can self-reported scores in CTM-AI help self-improve?

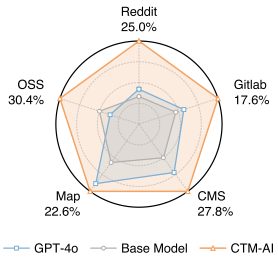


Figure 3: **Evaluation results on agentic tasks (WEBARENA-LITE).** Base model represents ReAct-style Gemini-2.5-flash-lite and CTM-AI uses the same backbone model. We report the success rate across 5 subdomains in web agent tasks.

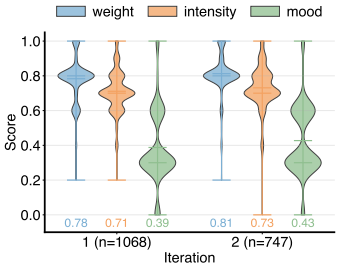


Figure 4: **Distribution of self-reported scores.** We summarize the score distribution of weight, intensity, and mood in each iteration, using Gemini-2.5-Flash-Lite as the backbone on the MUStARD dataset. n is the processor number.

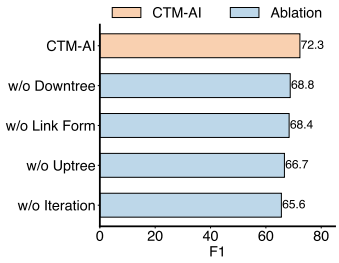


Figure 5: **Ablation on CTM-AI dynamics.** We isolate the contribution of each CTM-AI mechanism by ablating Step 2-4 and the iterative loop individually, using Gemini-2.5-Flash-Lite as the backbone of the MUStARD dataset.

Multi-dimensional score decomposition provides an effective routing signal. Rather than asking each processor for a single scalar score, we decompose chunk scoring into three interpretable subscores—weight, intensity, and mood (Eq. 3)—to yield a discriminative ranking signal for up-tree competition. Figure 4 shows that these sub-scores exhibit well-differentiated distributions, with both weight and intensity spanning 0.6 to 1.0. In addition, our ablation study in Appendix §G demonstrates that replacing CTM-AI's weighted sampling with minimum- or random-selection drastically degrades performance: on MUStARD, F1 drops from 72.23 to 62.92 (-9.3) under min-selection and to 66.76 (-5.5) under random-selection; on UR-FUNNY, it drops from 72.13 to 53.07 (-19.1) and 66.07 (-6.1), respectively. Our structured score also achieves a 62.9% downstream pass rate, outperforming self-prompted Yes/No logits (55.0%). Together, these results highlight two key insights: (1) pairing query alignment (weight) with a down-weighted tiebreaker (mood) reliably filters out low-quality processor chunks; and (2) effective multi-agent routing hinges on relative ranking utility rather than strict probabilistic calibration.

Iterative multi-agent evolution progressively refines processor outputs until convergence. Through global broadcast and link fusion, processors leverage cross-processor context across rounds to successively improve their predictions. Tracking the distribution of self-reported sub-scores across iterations on MUStARD (Figure 4) reveals that from iteration 1 to 2, weight ($0.78 \rightarrow 0.81$), intensity ($0.71 \rightarrow 0.73$), and mood ($0.39 \rightarrow 0.43$) all increase before stabilizing by iteration 3. This progression yields a key insight: even without external environmental feedback, global broadcast acts as an internal calibration signal, enabling processors to refine their generations, build confidence across rounds, and naturally converge to a stronger final output.

Q3: As an inference-time scaling method, how does CTM-AI compare with reasoning?

CTM-AI and reasoning scale along orthogonal, stackable axes. While standalone reasoning models improve over base models by $+10$ points on StableToolBench and $+5$ points on MUStARD/UR-FUNNY (Tables 1 and 2), they still underperform CTM-AI under a fixed backbone because chain-of-thought alone cannot coordinate across modalities and tools. Crucially, these two scaling paradigms are complementary: applying CTM-AI on top of Qwen3-8B-think yields an additional $+16.8$ to $+27.3$ points on StableToolBench over Qwen3-8B-think alone. Mechanistically, whereas internal reasoning deepens deliberation *within* each processor, CTM-AI scales coordination *across* processors. Stronger local reasoning produces higher-quality chunks and sharper queries throughout the cycle, causing the two gains to compound rather than overlap.

Upgrading a single processor with a reasoning model elevates the entire system. Upgrading only the text processor from Gemini-2.5-Flash-Lite to OpenAI o3 boosts overall MUStARD F1 from 72.23 to 78.11 ($+5.88$), surpassing standalone o3. Notably, the stronger text processor does not dominate the workspace: winning chunks remain balanced across audio (52.2%), video (25.3%), and text (22.5%), yet accuracy on text-winning samples surges to 82.46%. This result is because modularity allows stronger reasoning to be injected into a single modality without retraining the rest, uplifting peer processors rather than overriding them. Specifically, o3 not only calibrates its own chunk scores more selectively (hence the high precision at a 22.5% win rate), but also replaces generic broadcasts (e.g., *"What is the comedic tone?"*) with targeted cross-modal queries (e.g., *"Does Chandler roll*

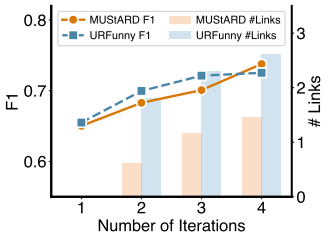


Figure 6: **Ablation on max iteration number** T. We use $\tau = 2.2$, $\eta = 0.9$ for MUStARD and $\tau = 2.2$, $\eta = 0.7$ for UR-FUNNY. When $T = 1$, no links are formed.

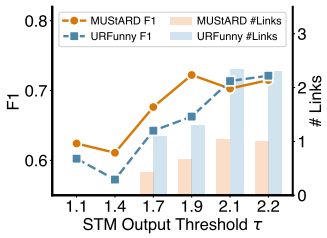


Figure 7: **Ablation on STM output threshold τ.** We keep $\eta = 0.9$ for MUStARD and $\eta = 0.7$ for UR-FUNNY, with 3 max iterations for MUStARD and UR-FUNNY.

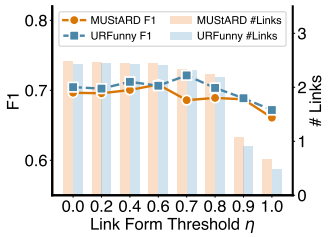


Figure 8: **Ablation on link form threshold η.** The higher η, the harder to form links. We set $\tau = 2.2$ and 3 max iterations for both MUStARD and UR-FUNNY.

his eyes?"), guiding the other processors in the CTM-AI, including audio and video ones, to extract more discriminative features (Appendix §H).

4.4 Ablation Studies

Ablation on CTM-AI dynamics. CTM-AI relies on five mechanisms: (i) processor inference, (ii) up-tree competition, (iii) down-tree broadcast, (iv) link formation and fusion, and (v) the iterative loop. Processor inference is always required, so we disable each of the other four in turn. As shown in Figure 5, removing any single mechanism degrades performance. Ranked by impact, the losses are: iterative loop (-6.7 F1), up-tree competition (-5.6), link fusion (-3.9), and down-tree broadcast (-3.5). Such results indicate that each mechanism has a distinct role, which explains the ranking. The iterative loop decides whether the system can refine its output at all. Up-tree competition makes sure the most informative chunk wins attention. Fusion integrates information across processors. Broadcast keeps all processors in sync. Even the smallest drop is substantial (-3.5 F1), so every mechanism is needed and CTM-AI is not over-engineered.

Ablation on max iteration number T. As shown in Figure 6, increasing T from 1 to 4 improves performance steadily on both datasets. MUStARD rises from 65.0 to 73.8 (+8.8), and UR-FUNNY rises from 65.5 to 72.5 (+7.0). The average number of inter-processor links grows along with performance. At T=4, UR-FUNNY forms 2.6 links on average, compared with 1.5 for MUStARD. These results indicate that extra iterations pay off primarily by enabling denser link formation. Notably, while performance trends are similar across both datasets, their link counts differ—demonstrating that CTM-AI adaptively scales its cross-processor interaction to task demands without manual tuning.

Ablation on thresholds τ and η. We test robustness to two hyperparameters. The first is the short-term memory (STM) output threshold τ: a higher τ means stricter output filtering. The second is the link formation threshold η: a higher η makes it harder for two processors to form a link. As shown in Figure 7, raising τ generally improves performance on both MUStARD and UR-FUNNY, together with more inter-processor links. As shown in Figure 8, performance is best for $\eta \in [0.4, 0.7]$ on both datasets. Both performance and the number of links drop sharply once $\eta > 0.8$. Such results indicate that a stricter τ stops the system from committing to an answer too early. This leaves more rounds for processors to form links and exchange evidence. η shows a clear trade-off: if it is too low, weak links add noise, and if it is too high, useful cross-processor fusion is cut off. The same good ranges hold on both datasets, so CTM-AI is reasonably robust to these hyperparameters.

5 Conclusion

In this paper, we introduced CTM-AI, a self-improving multi-agent framework that translates the theoretical principles of the Conscious Turing Machine (CTM) into a practical AI system. Through strong performance across multimodal perception, tool use, and agentic tasks, CTM-AI establishes a robust foundation for cognitively inspired multi-agent collaboration and iterative refinement. Moving beyond philosophical debates on machine consciousness, we hope our work inspires future research toward building more capable, decentralized, and self-improving multi-agent architectures.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] John R Anderson, Michael Matessa, and Christian Lebiere. Act-r: A theory of higher level cognition and its relation to visual attention. *Human–Computer Interaction*, 12(4):439–462, 1997.

[3] Bernard J Baars. *A cognitive theory of consciousness*. Cambridge University Press, 1993.

[4] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[5] Lenore Blum and Manuel Blum. A theory of consciousness from a theoretical computer science perspective: Insights from the conscious turing machine. *Proceedings of the National Academy of Sciences*, 119(21): e2115934119, 2022.

[6] Lenore Blum and Manuel Blum. A theoretical computer science perspective on consciousness and artificial general intelligence. *Engineering*, 2023.

[7] Manuel Blum and Lenore Blum. A theoretical computer science perspective on consciousness. *Journal of Artificial Intelligence and Consciousness*, 8(01):1–42, 2021.

[8] Patrick Butlin, Robert Long, Eric Elmoznino, Yoshua Bengio, Jonathan Birch, Axel Constant, George Deane, Stephen M Fleming, Chris Frith, Xu Ji, et al. Consciousness in artificial intelligence: insights from the science of consciousness. *arXiv preprint arXiv:2308.08708*, 2023.

[9] Stuart K Card, Thomas P Moran, and Allen Newell. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, 23(7):396–410, 1980.

[10] Santiago Castro, Devamanyu Hazarika, Verónica Pérez-Rosas, Roger Zimmermann, Rada Mihalcea, and Soujanya Poria. Towards multimodal sarcasm detection (an _obviously_ perfect paper). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4619–4629, 2019.

[11] Mert Cemri, Shu Liu, Shubham Agarwal, Monishwaran Maheswaran, Zhifei Li, Qiuyang Mang, Ashwin Naren, Kurt Keutzer, Alexandros G. Dimakis, Koushik Sen, Matei Zaharia, and Ion Stoica. AdaEvolve: Adaptive LLM driven zeroth-order optimization. *arXiv preprint arXiv:2602.20133*, 2026.

[12] Xiusi Chen, Shanyong Wang, Cheng Qian, Hongru Wang, Peixuan Han, and Heng Ji. Decisionflow: Advancing large language model as principled decision maker, 2025. URL `https://arxiv.org/abs/2505.21397`.

[13] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[14] Wei Dai, Peilin Chen, Chanakya Ekbote, and Paul Pu Liang. Qoq-med: Building multimodal clinical foundation models with domain-aware grpo training. *arXiv preprint arXiv:2506.00711*, 2025.

[15] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate, 2023. URL `https://arxiv.org/abs/2305.14325`.

[16] Stan Franklin, Tamas Madl, Sidney D'mello, and Javier Snaider. Lida: A systems-level architecture for cognition, emotion, and learning. *IEEE Transactions on Autonomous Mental Development*, 6(1):19–41, 2013.

[17] Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. *arXiv preprint arXiv:2403.07714*, 2024.

[18] Zhicheng Guo, Sijie Cheng, Yuchen Niu, Hao Wang, Sicheng Zhou, Wenbing Huang, and Yang Liu. Stabletoolbench-mirrorapi: Modeling tool environments as mirrors of 7,000+ real-world apis. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 5247–5270, 2025.

[19] Md Kamrul Hasan, Wasifur Rahman, AmirAli Bagher Zadeh, Jianyuan Zhong, Md Iftekhar Tanveer, Louis-Philippe Morency, and Mohammed Ehsan Hoque. Ur-funny: A multimodal language dataset for understanding humor. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2046–2056, 2019.

[20] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

[21] Yubin Kim, Ken Gu, Chanwoo Park, Chunjong Park, Samuel Schmidgall, A Ali Heydari, Yao Yan, Zhihan Zhang, Yuchen Zhuang, Mark Malhotra, et al. Towards a science of scaling agent systems. *arXiv preprint arXiv:2512.08296*, 2025.

[22] John E Laird, Allen Newell, and Paul S Rosenbloom. Soar: An architecture for general intelligence. *Artificial intelligence*, 33(1):1–64, 1987.

[23] Hengzhi Li, Brendon Jiang, Alexander Naehu, Regan Song, Justin Zhang, Megan Tjandrasuwita, Chanakya Ekbote, Steven-Shine Chen, Adithya Balachandran, Wei Dai, et al. Puzzleworld: A benchmark for multimodal, open-ended reasoning in puzzlehunts. *arXiv preprint arXiv:2506.06211*, 2025.

[24] Jiulin Li, Ping Huang, Yexin Li, Shuo Chen, Juewen Hu, and Ye Tian. A unified multi-agent framework for universal multimodal understanding and generation. *arXiv preprint arXiv:2508.10494*, 2025.

[25] Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. Align before fuse: Vision and language representation learning with momentum distillation. *Advances in neural information processing systems*, 34:9694–9705, 2021.

[26] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[27] Paul Pu Liang, Amir Zadeh, and Louis-Philippe Morency. Foundations & trends in multimodal machine learning: Principles, challenges, and open questions. *ACM Computing Surveys*, 56(10):1–42, 2024.

[28] Huawei Lin, Yunzhi Shi, Tong Geng, Weijie Zhao, Wei Wang, and Ravender Pal Singh. Agent-omni: Test-time multimodal reasoning via model coordination for understanding anything. *arXiv preprint arXiv:2511.02834*, 2025.

[29] Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, et al. Visualagentbench: Towards large multimodal models as visual foundation agents. *arXiv preprint arXiv:2408.06327*, 2024.

[30] George A Mashour, Pieter Roelfsema, Jean-Pierre Changeux, and Stanislas Dehaene. Conscious processing and the global neuronal workspace hypothesis. *Neuron*, 105(5):776–798, 2020.

[31] Microsoft. Autogen. `https://devblogs.microsoft.com/autogen/`, 2024. Accessed: 2026-04-18.

[32] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

[33] OpenAI. GPT-4o system card. Technical report, OpenAI, 2024. URL `https://openai.com/index/gpt-4o-system-card/`.

[34] OpenAI. OpenAI o3 and o4-mini system card, 2025. URL `https://openai.com/index/o3-o4-mini-system-card/`.

[35] Sarah Partan and Peter Marler. Communication goes multimodal. *Science*, 283(5406):1272–1273, 1999.

[36] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.

[37] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

[38] Ao Qu, Han Zheng, Zijian Zhou, Yihao Yan, Yihong Tang, Shao Yong Ong, Fenglu Hong, Kaichen Zhou, Chonghe Jiang, Minwei Kong, Jiacheng Zhu, Xuan Jiang, Sirui Li, Cathy Wu, Bryan Kian Hsiang Low, Jinhua Zhao, and Paul Pu Liang. Coral: Towards autonomous multi-agent evolution for open-ended discovery, 2026. URL https://arxiv.org/abs/2604.01658.

[39] David Rosenthal. *Consciousness and mind*. Clarendon Press, 2005.

[40] Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using llm agents as research assistants. *arXiv preprint arXiv:2501.04227*, 2025.

[41] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

[42] Giulio Tononi. An information integration theory of consciousness. *BMC neuroscience*, 5:1–22, 2004.

[43] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937. doi: 10.1112/plms/s2-42.1.230.

[44] Rufin VanRullen and Ryota Kanai. Deep learning and the global workspace theory. *Trends in Neurosciences*, 44(9):692–704, 2021.

[45] Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024.

[46] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[47] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[48] Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, et al. Qwen3-omni technical report. *arXiv preprint arXiv:2509.17765*, 2025.

[49] Yingxuan Yang, Huacan Chai, Yuanyi Song, Siyuan Qi, Muning Wen, Ning Li, Junwei Liao, Haoyi Hu, Jianghao Lin, Gaowei Chang, et al. A survey of ai agent protocols. *arXiv preprint arXiv:2504.16736*, 2025.

[50] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[51] Haofei Yu, Zhengyang Qi, Lawrence Jang, Ruslan Salakhutdinov, Louis-Philippe Morency, and Paul Pu Liang Liang. Mmoe: Enhancing multimodal models with mixtures of multimodal interaction experts. *arXiv preprint arXiv:2311.09580*, 2023.

[52] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

[53] Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025.

A Impact Statement

This work utilizes publicly available datasets; no private or sensitive user data was collected, and all experiments were conducted in controlled research settings. By bridging the theoretical CTM framework with practical AI technologies, we aim to enhance AI capabilities in affective learning, decision-making, multi-step reasoning, and tool use, contributing to the development of more reliable and trustworthy general AI. Crucially, our goal is neither to build conscious AI nor to replicate human identity, thereby avoiding the ethical risks associated with deceptive anthropomorphization. We also acknowledge the inherent risks of deploying foundation models and AI agents, particularly the propagation of socio-cultural biases. Actively detecting, understanding, and mitigating these biases remains a central commitment of our ethical research framework.

B Limitations

Scale of processor federation. The original CTM theory envisions a federation of up to 10^7 LTM processors. Our principal tasks typically use fewer than ten active processors (three modalities or roughly six retrieved tool processors); a separate $K = 1$–32 study and a 512-processor stress test do not approach the theoretical scale. As a result, the simplified global competition we adopt in Step 2 (Eq. 4) approximates—but does not fully exercise—the hierarchical up-tree competition prescribed by the theory. Scaling CTM-AI to substantially larger and more heterogeneous processor pools, and studying whether hierarchical competition yields additional benefits at scale, remains an open direction. The 512-processor stress test requires approximately 1,022 model calls and 931k tokens for a query and is evidence of operational feasibility only, not scalable efficiency.

Inference cost. Although CTM-AI achieves competitive latency relative to other multi-agent frameworks (Table 3), its iterative inference loop and link-fusion stage still incur higher API cost than unified single-model baselines. On UR-FUNNY, CTM-AI$_3$ uses 17.0 API calls per instance versus 1.0 for the unified baseline. Reducing this overhead is a natural next step.

C The Use of Large Language Models (LLMs)

We used LLMs as a writing assistant to help us edit parts of the paper. Additionally, we utilize the power of CodePilot and Claude Code to help us code faster. All AI-generated writing and code are manually checked and modified. There is no fully AI-generated content in the paper.

D Artifact Details

D.1 Model License

GPT-4o [33] License: Proprietary (OpenAI)
OpenAI-o3 [34] License: Proprietary (OpenAI)
Gemini-2.5-flash-lite [13] License: Apache 2.0
Qwen3-VL-8B-Instruct [4] License: Apache 2.0
Qwen3-VL-8B-thinking [4] License: Apache 2.0
Qwen3-Omni-flash [48] License: Proprietary (Alibaba)

D.2 Software Versions

For web-agent evaluation, we adopt BrowserGym v0.14.2 [2]. To access large language models, we employ LiteLLM 1.74.3 [3] as the serving interface.

D.3 Dataset Statistics

We include the test sets of MUStARD, URFunny, WebArena-Lite, and StableToolBench for evaluation. Table 5 shows their statistics.

[2] https://github.com/ServiceNow/BrowserGym
[3] https://litellm.ai

MUStARD [10] is a multimodal sarcasm detection dataset collected from TV shows, where each instance consists of an utterance with its conversational context and a binary sarcasm label. We evaluate on its 356-instance test split.

UR-Funny [19] is a multimodal humor detection dataset built from TED talks, where the task is to predict whether a punchline is humorous given its textual, visual, and acoustic context. We evaluate on its 992-instance test split.

WebArena-Lite [29] is a human-verified subset of WebArena [52] comprising tasks across five self-hosted websites: Reddit, GitLab, CMS, Map, and OneStopShop (OSS). Each task requires an agent to complete a natural-language instruction by interacting with real web pages, and is evaluated by programmatic success checkers.

StableToolBench [17] extends ToolBench [37] with a stabilized virtual API server and a solvability-filtered query set for tool-use evaluation. We evaluate on the I2-Inst, I2-Cat, and I3-Inst subsets using MirrorAPI-Cache [18] as the tool environment, which fine-tunes a specialized LLM on StableToolBench's cached API traces to stably mirror real API behaviors. As noted by [18], some queries in the original test set reference APIs that are no longer available; such queries may fail during evaluation regardless of the agent's behavior. We report results over queries with valid APIs in each subset.

E Experimental Details

In this section, we provide more implementation details related to the algorithm that we proposed based on CTM-AIWe also include the prompting details to explain how we adapt CTM-AI architecture to different types of tasks.

E.1 Details of Backbone Models

We select Gemini-2.5-flash-lite as our base model to make most of the processors. It is mainly because Gemini-2.5-flash-lite is relatively small-scale and supports audio, vision, and text as input for inference. When querying the Gemini API, we adopt a deterministic decoding configuration with temperature fixed at 0.1, top-n set to 1, and a maximum token limit of 4096.

E.2 Details of Unified Model Baselines

We evaluate unified model architectures across four distinct tasks, encompassing both fine-tuned models and prompting-based foundation models.

Sarcasm Detection (MUStARD). For fine-tuned baselines, we evaluate ALBEF [25] (209.5M parameters), BLIP2 [26] (2.7B parameters), MMoE [51], and our BaseModel (Gemini-2.5-flash-lite). For prompting-based baselines, we evaluate Qwen3-VL-8B-Instruct, Qwen3-VL-8B-thinking, Qwen3-Omni-flash, and Gemini-2.5-flash-lite using identical zero-shot/few-shot prompts.

Humor Detection (URFUNNY). To assess multimodal affective understanding, we evaluate this task using the same comprehensive suite of unified models as the MUStARD task, including both the fine-tuned architectures (ALBEF, BLIP2, MMoE, Gemini BaseModel) and prompting-based foundation models (Qwen3 variants, Gemini-2.5-flash-lite).

API Tool Calling (StableToolBench). We evaluate ToolLLaMA v2 [37] as the fine-tuned baseline, which is explicitly trained on the benchmark's train set. For prompting and search-based models, we compare GPT-4o-mini, GPT-4o, Qwen3-8B, Qwen2-8B-think, and Gemini-2.5-flash-lite, utilizing Chain-of-Thought (CoT) [47] and Depth-First Search (DFS) strategies via MirrorAPI-Cache.

Web Navigation (WebArena-Lite). We evaluate standalone foundation models in a direct agentic setting without multi-agent orchestration, specifically comparing the unassisted generation capabilities of GPT-4o and Gemini-2.5-flash-lite. We all use React as an agentic strategy that interleaves reasoning and acting to solve complex tasks. We use this as the primary baseline framework for the WebArena-Lite environment, driven by GPT-4o and Gemini-2.5-flash-lite backbones.

Table 5: **Dataset statistics**. Number of test instances for each evaluation dataset/subset. For StableToolBench, we chose a subset since part of the queries may fail under the MirrorAPI-Cache simulation.

Affective		**WebArena-Lite**					**StableToolBench**		
MUStARD	URFunny	Reddit	GitLab	CMS	Map	OSS	I2-Inst	I2-Cat	I3-Inst
356	992	24	34	36	31	46	106	124	61

E.3 Details of Multi-Agent Baselines

To assess the advantages of CTM-AI's dynamics against existing multi-agent paradigms, we compare it against the following frameworks. Unless otherwise specified, these baselines utilize Gemini-2.5-flash-lite as the backbone model and maintain consistent hyperparameters ($T = 0.2$, layers $N = 3$).

Multi-Agent ensemble (Ensemble). We conduct inference for all processors in the CTM-AI (video, audio, and text processor). Each processor outputs one answer, and we directly conduct majority voting on all of them to make the final answer.

Multi-agent debate (Debate). A collaborative framework where multiple agents argue from different viewpoints over successive rounds. At each round, every agent observes the previous round's responses from all other agents and is asked to either defend its position or revise its answer; a separately prompted judge model then aggregates the final round into a single decision. We instantiate three debaters (one per modality: video, audio, text) plus one judge, and strictly cap the total debate depth at 10 API calls per example.

Multi-agent centralized orchestra (Orchestra). A structured three-stage workflow inspired by planner–executor–aggregator designs. A *controller* first decomposes the input task into a set of self-contained sub-queries; an *executor* pool answers each sub-query independently and in parallel; finally, a *summarizer* aggregates the sub-answers and produces the final prediction. Only the executor stage is parallelizable; the controller and summarizer stages run sequentially.

Mixture-of-Agents (MoA). [45] A layered multi-agent architecture with $N = 3$ sequential layers, each consisting of one video, one audio, and one text agent. Every agent in layer ℓ conditions on all outputs from layer $\ell-1$ as auxiliary context when generating its own. Layer 1 produces initial modality-specific predictions, layers 2–3 iteratively refine them by cross-referencing peer outputs across modalities, and the final answer is produced by a last-layer aggregator.

MetaGPT. [20] A framework that encodes Standardized Operating Procedures (SOPs) as structured prompt sequences, with each agent assigned a fixed role and output schema. We instantiate MetaGPT with three domain-expert agents corresponding to the three modalities (video, audio, text), each following a role-specific SOP that requires it to extract modality-specific evidence, cross-check for inconsistencies with the other agents' outputs, and report a confidence-weighted verdict. The structured outputs are then passed to a final aggregator agent for the decision.

AutoGen. [31] Microsoft's general-purpose framework for multi-agent orchestration. We instantiate a `RoundRobinGroupChat` of three modality-expert agents (video, audio, text), each equipped with a modality-specific analysis tool, plus one judge agent that aggregates their verdicts. The chat terminates when the judge outputs a final answer or after $N = 3$ rounds.

E.4 Details of Evaluation

After receiving the conscious output from the STM, an additional parsing step is required to formulate the final answer for evaluation. We utilize Gemini-2.5-flash-lite to execute this final generation as a "conscious action", alongside an associated confidence score. Because the output spaces vary across tasks, the specific evaluation prompts for this final stage are tailored accordingly. The evaluation parsing prompt for UR-FUNNY has additional explanation on the final mapping of yes and no because we find that models can be biased towards outputting one type of answer and need to correct it.

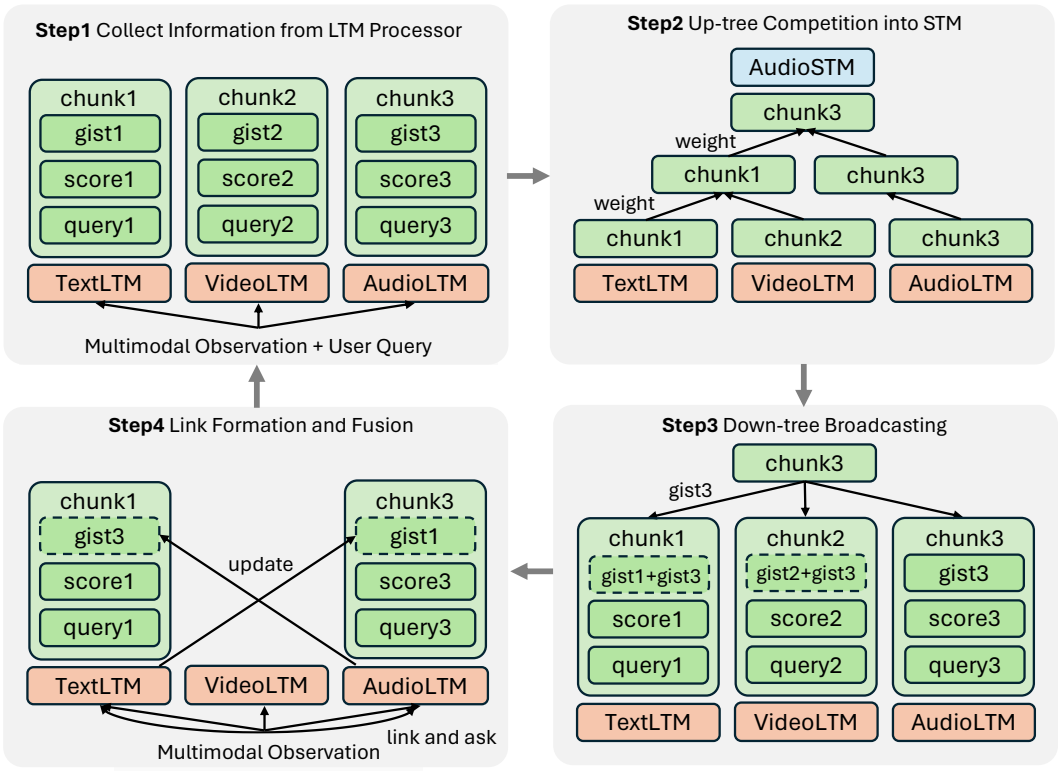


Figure 9: **Detailed dynamics of CTM-AI**. We decompose each chunk into three distinct components: a gist, a score, and a query, and describe the overall 4 stages with more details compared with Figure 2.

MUStARD evaluation parsing prompt.

```
You are a sarcasm detection expert. Based solely on the analysis provided below, determine if the person is being sarcastic. Your answer MUST start with either "Yes" (if sarcastic) or "No" (if not sarcastic), followed by a brief explanation. IMPORTANT: If the analysis expresses uncertainty, is inconclusive, or lacks sufficient evidence, you should answer "No". Analysis: {answer}
```

URFUNNY evaluation parsing prompt.

```
You are a humor detection expert. Based solely on the analysis provided below, determine if the punchline is humorous. Rules: - Answer "Yes" ONLY if the analysis identifies a SPECIFIC humor technique (self-deprecation, ironic reveal, absurd comparison, wordplay, incongruity, misdirection, deadpan understatement) with confidence >= 0.6 AND provides concrete evidence (specific words, phrases, or audience reactions).- If the analysis says humor is "possible" or "ambiguous" without strong evidence, your answer should be "No". - If the analysis concludes the content is NOT humorous, your answer should be "No". - If the analysis mentions audience laughter as evidence, that is strong evidence for "Yes". - A serious or calm delivery does NOT mean the content is not humorous — deadpan delivery is common. Your answer MUST start with either "Yes" or "No", followed by a brief explanation. Analysis: {answer}
```

StableToolBench evaluation parsing prompt.

```
You are an expert in tool use; you should answer the task based solely on the analysis provided below. Your answer should be comprehensive and concise. Task: {query} Analysis: {answer}
```

WebArena-Lite Evaluation Prompt.

```
You are an expert UI assistant. Summarize the current step. This summary will
```

```
be passed to future steps as context, so it MUST preserve all key factual evidence.  Task: {query} Action history:{action_history} Winning processor reasoning: {reasoning} Chosen action: answer Write a step summary in 2-4 plain text sentences that includes: 1. All key facts discovered (exact prices, product names, quantities, IDs, URLs, usernames, dates, error messages, etc.) 2. The reasoning behind the chosen action 3. The action taken 4. What remains to be done: CRITICAL RULES: - You MUST include every specific data point (numbers, names, IDs) from the reasoning. These facts will NOT be available later if you omit them. - NEVER claim an action succeeded or that a task is complete. You are only recording WHAT ACTION WAS ISSUED, not its outcome. The result will only be visible in the NEXT step's page state. For example, write "Issued click on Add to Cart button" NOT "Added the product to the cart".- Output ONLY plain text sentences. Do NOT output any JSON, code blocks, function calls, or structured data. No "'json"', no send_msg_to_user(), no curly braces.
```

F CTM-AI Implementation Details

In this section, we provide a more detailed description of the implementation details of CTM-AI. Figure 9 shows a more detailed description of the iterative inference process in CTM-AI.

F.1 LTM Processor Implementation

While the theoretical CTM architecture can scale to a virtually unlimited number of processors, evaluating such a massive system introduces compounding variables that obscure direct baseline comparisons. To ensure a fair and controlled empirical evaluation, we heuristically select a compact, task-specific subset of LTM processors for each benchmark. This deliberate scoping isolates the core benefits of our proposed mechanisms (e.g., the Up-Tree competition) while keeping the playing field level with existing baselines. Below, we detail the exact processor configurations deployed for each task.

MUStARD and URFUNNY. For these multimodal affective tasks, we deploy three modality-specific experts. Each processor receives the user query alongside its isolated modality stream:

- **Video processor:** Observes only the muted video.
- **Audio processor:** Observes only the audio track.
- **Text processor:** Observes only the textual transcript.

StableToolBench. In this environment, each available tool (API) acts as an independent LTM processor. These tool-processors are dynamically populated using the benchmark's native retrieval model, resulting in an average of 5.94 processors per task. Each processor is powered by a lightweight LLM (Gemini-2.5-flash-lite) that is strictly constrained to utilize only its assigned API.

WebArena-Lite. For web navigation, all processors share a common temporal context (the user's objective, action space, action history, and previous action). However, they perceive the current page state through distinct representational modalities:

- **HTML processor:** Parses the raw HTML DOM of the current page.
- **Accessibility tree processor:** Parses the accessibility tree structure of the current page.
- **Screenshot processor:** Processes a visual screenshot of the current page augmented with Set-of-Mark (SoM) annotations.

F.2 Chunk Inference Implementation Details

As defined in Equation 2, the function $\text{CTM}_{\text{collect}}(\cdot)$ processes the multimodal observation o_t and the user query q_t to generate multiple chunks. Formally, each chunk is represented as $\langle \text{addr}(p_i),\, t,\, h_t^i,\, q_t^i,\, s_t^i \rangle$. In practice, when a processor is queried, it returns a JSON object containing three primary elements: a gist h_t^i (*e.g.*, "the woman is smiling"), an additional internal question q_t^i to guide further processing (*e.g.*, "What is she speaking about?"), and a composite score s_t^i. This score is a linear combination of weight, intensity, and mood, using a ratio of $1 : 1 : 0.2$ to prioritize weight and intensity. This structured decomposition of a single confidence score into interpretable

sub-components is in line with recent findings that structured deliberation over decision attributes leads to more reliable LLM judgments [12].

To adapt chunk inference across benchmarks, the prompt template for generating scores remains fixed, while task-specific definitions are appended. Crucially, we do not assign specialized personas to different processors; all are instructed to directly answer the query, conditioned strictly on their partial multimodal observations. These conditional instructions are framed as properties of the task (the query) and the modality (the observation):

MUStARD and URFUNNY. The video, audio, and text processors share the identical prompt template but receive different input modalities. Appendix §I details the prompts responsible for score generation, gists, and additional queries.

StableToolBench. Processors across all available tools share a uniform prompt to extract information. Appendix §I details the weight, gist, and query generation prompts tailored for the API environment.

WebArena-Lite. While the weight generation prompt remains identical, the processors require modality-specific explanations to parse unique inputs, such as accessibility trees and SoM screenshots (detailed in Appendix §I).

F.3 Up-Tree Implementation Details

As formalized in Equation 4, the $\text{CTM}_{\text{up}}(\cdot)$ function evaluates the generated chunks and selects a single winning chunk to become the system's conscious content (STM).

F.4 Down-Tree Implementation Details

During the down-tree propagation phase, the winning chunk globally broadcasts its generated answer. In our implementation, each LTM processor maintains an internal Python list, `winner_answer`, which serves as a persistent record of the conscious sequence. The winning answer is appended to every processor's list.

In subsequent iterations, when a new query is issued, the system provides this accumulated memory as contextual guidance using the following prefix:
`"There are previous responses to the same query. Please reason further based on the following answer(s): {winner_answers}."`

F.5 Link Formation Implementation Details

To determine whether an unconscious link should form between two LTMs, the STM queries each LTM using its generated additional questions (q_t^i). This querying procedure mirrors the primary user query format (described in Prompt 1). We maintain a `adjacency_list` to track these dynamic connections.

The key distinction lies in the scoring criterion: link formation relies solely on the *weight* sub-score. If an answering processor yields a weight > 0.8, a directed link is established between the winning LTM and the answering LTM.

F.6 Link Fusion Implementation Details

Each LTM maintains a `fuse_history` list. When a link exists between two LTMs, they cross-evaluate each other's additional questions, and the resulting responses are appended to their respective histories. During the main query inference, a processor's context is augmented with its linked neighbors' insights using the following prefix:
`"There is extra information from other processors: [processor_name]: [answers]."`

F.7 Overall Inference Algorithm

We provide the complete inference algorithm for CTM-AI. To clarify the mechanics of chunk generation, up-tree competition, down-tree broadcast, and link formation, we explicitly decompose the generic chunk_t^i into its fine-grained components (gist, query, and weight) within the pseudocode.

Table 6: **Selection-score analyses**. Comparison of minimum-score selection, random selection, and CTM-AI's weighted sampling on MUStARD and UR-FUNNY.

	Minimum		**Random**		**CTM-AI**	
Dataset	Acc↑	F1↑	Acc↑	F1↑	Acc↑	F1↑
MUStARD	65.45	62.92	67.42	66.76	72.28	72.23
URFUNNY	57.56	53.07	66.40	66.07	72.13	72.13

Cost analysis. Assuming K processors and L established links in the processor graph, a single iteration requires $2(K + L)$ processor calls: K for initial chunk inference, K for evaluating the winning chunk's link formation, and $2L$ for bidirectional multimodal fusion. Because cross-processor links form selectively, L is typically much smaller than K ($L \ll K$). Most tasks resolve within 1 to 3 iterations.

Efficiency analysis. The system's temporal bottleneck lies in the API calls required for chunk inference, link formation, and link fusion. Because these three stages are executed in parallel, the wall-clock time per iteration is approximately $3T + \epsilon$, where T is the latency of a single API call and ϵ represents the negligible local computation time for the up-tree and down-tree routing.

G Additional analyses of self-reported scores

To assess whether LTM processors provide reliable scores without external feedback, we compare CTM-AI's weighted sampling against two ablations: *Minimum* (selecting the chunk with the lowest score) and *Random* (uniformly random selection). Results are shown in Table 6. CTM-AI consistently outperforms both baselines across MUStARD and UR-FUNNY, suggesting that self-reported scores provide useful signals for guiding chunk selection.

We further investigate whether alternative scoring signals can improve chunk selection on StableToolBench. We compare our structured self-reported scores against an external LLM judge, sequence-normalized log-probability, predictive entropy, and self-prompted Yes/No logits. Our structured scores achieve the highest downstream task pass rate (62.9%), outperforming the external LLM judge (60.4%), log-probability (58.7%), predictive entropy (53.1%), and Yes/No logits (55.0%). Interestingly, Yes/No logits achieve better discrimination (AUROC: 0.886 vs. 0.849) and calibration (ECE: 0.097 vs. 0.199), but yield lower task performance. This suggests that better confidence calibration does not necessarily translate into more effective chunk selection. Overall, our structured scores provide the strongest downstream selection performance among the evaluated scoring methods.

H Additional Case Study

Full iteration case study. Based on Figure 10, we analyze a multimodal perception case for identifying *sarcasm*. In the **first iteration**, all three processors are initially *uncertain* due to partial observations and limited cross-modal context. The audio processor reports: *"It doesn't exhibit the specific vocal patterns that typically indicate sarcasm."* The video processor notes: *"Challenging to determine, but the expressions and body language do suggest a possible level of irony."* The text processor states: *"The text alone is only a simple command; I need more context to determine the exact answer."* The text processor wins the up-tree competition and broadcasts its partial understanding to all processors, explicitly requesting more context. This broadcast prompts the video processor to respond with relevant visual cues, forming a link for sharing information about the scene and dialogue. In the **second iteration**, the video and text processors engage in unconscious communication via their newly formed link. The video processor responds to the text processor's query with: *"Monica has a shocked face, and Joey is shirtless in the kitchen."* Integrating these contextual cues with its own visual frames, the video processor infers that the speaker is likely being sarcastic. However, it remains uncertain and asks for accompanying audio to reach a more comprehensive judgment. In the **third iteration**, the video processor queries the audio processor and receives prosodic and tonal cues. With this enriched multimodal evidence, it refines its judgment and concludes that the speaker is *not* sarcastic, but instead expresses genuine concern with a shocked and somewhat exaggerated facial expression. Through repeated broadcasting and mutual communication, the processors progressively link their evidence, fuse perspectives, and converge on the correct answer.

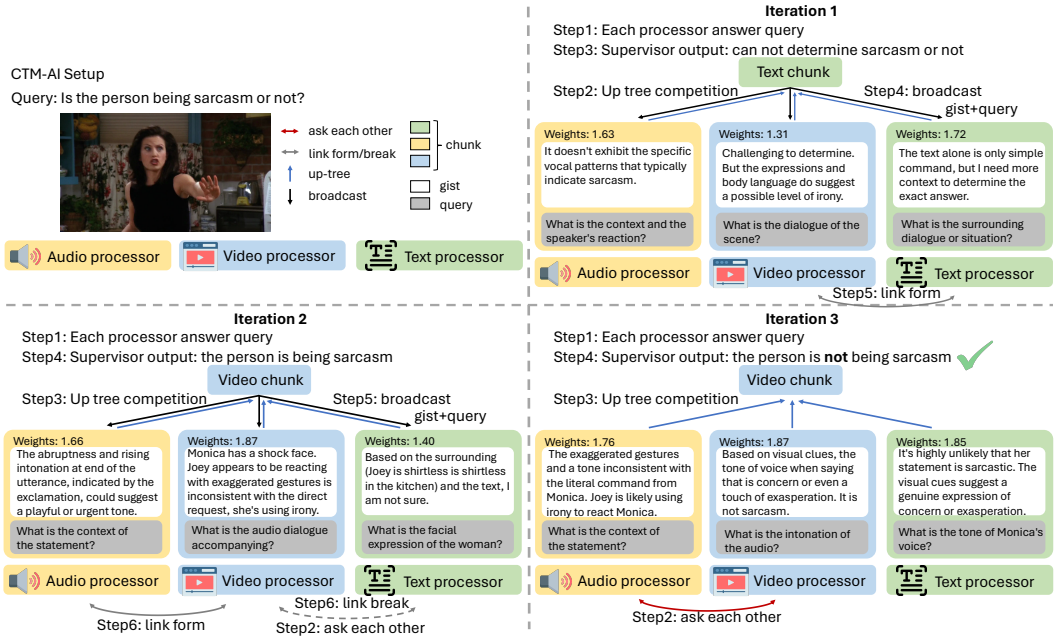


Figure 10: **Case study of CTM-AI dynamics.** We show three iterations of CTM-AI for sarcasm detection. Through multiple rounds of structured interaction, the system progressively integrates multimodal cues and converges on the correct interpretation.

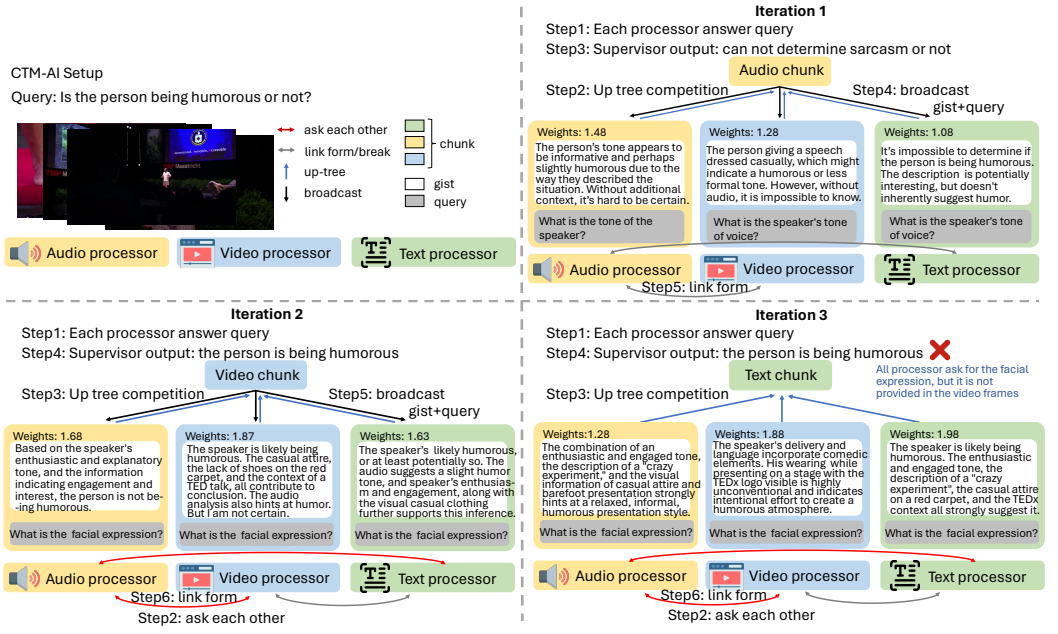


Figure 11: **Failure mode in affective computing (vision-only misleads).** The failure case is caused by incomplete observation of the video processor; all the LTMs have the same question beginning in the second iteration: *"What is the facial expression?"* But due to the lack of facial expression in the input video frames, too many links are formed to get the missing information, and the LTMs can not have correct answers.

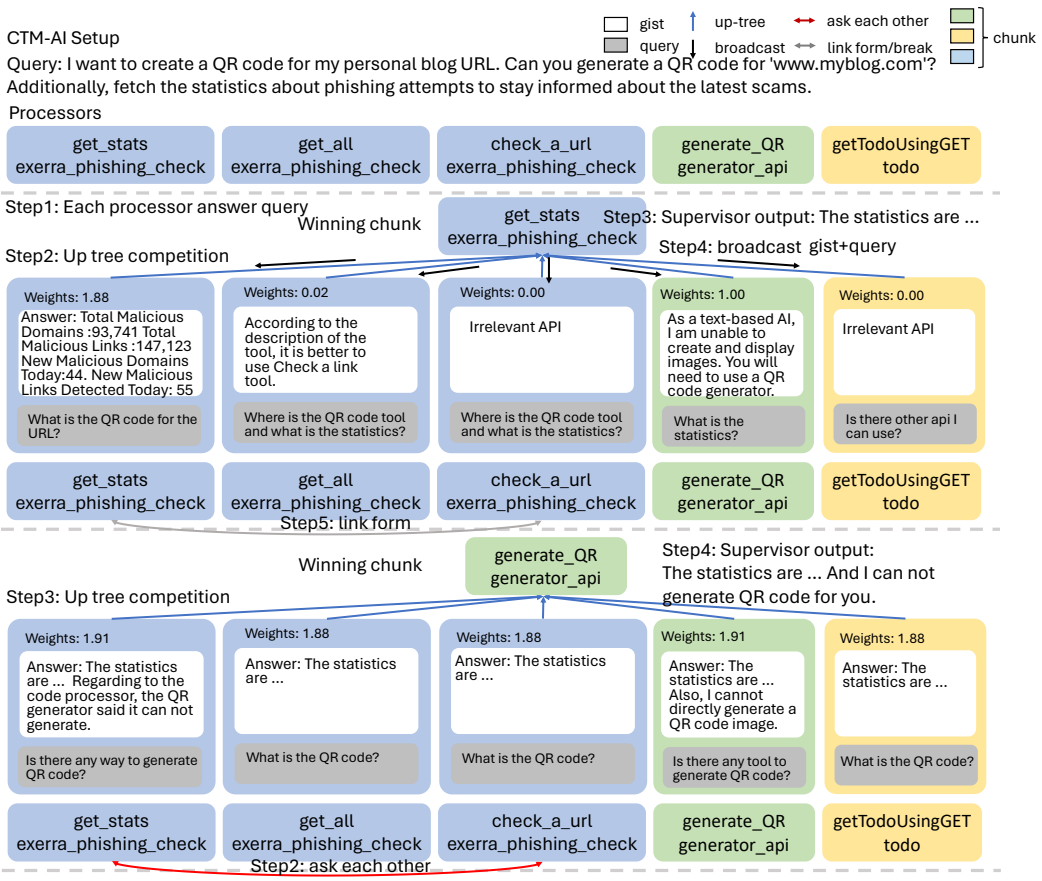


Figure 12: **Failure mode in StableToolBench (tool mishandle).** This failure occurred because the processor assigned to QR-code generation did not issue the required API call. Instead, it produced a premature judgment stating that it was unable to generate the QR code, without interacting with the tool.

Failure case study. Additionally, we also conduct analysis for failure cases. We present two detailed example of CTM-AI in URFunny (Figure. 11) and StableToolBench (Figure. 12). The failure observed in URFunny is caused by a vision-only misleading effect, which is caused by the incomplete visual observations available to the video processor. Beginning from the second iteration, all LTMs repeatedly generated the same additional question: *"What is the facial expression?"*, but the input video frames did not contain the necessary facial-expression information. As a result, the system created an excessive number of links in an attempt to acquire the missing information, ultimately preventing the LTMs from producing correct answers. The failure in StableToolBench is attributed to tool mishandling. Specifically, the processor responsible for QR-code generation failed to invoke its designated API. Instead of issuing the required tool call, it prematurely concluded that it was unable to generate the QR code, thereby producing an incorrect outcome without interacting with the tool.

I Detailed Prompts

We provide the detailed prompts used in our experiments, including (i) the self-reported score prompts that elicit processor confidence estimates, (ii) the system prompts for MUStARD, UR-FUNNY, StableToolBench, and WebArena-Lite, which provide processors with the basic task context, and (iii) the additional question-generation instructions, which ask each processor to propose one or more follow-up queries $q_t \in \mathcal{Q}$ to other processors whenever answering them could improve the final answer. For WebArena-Lite, we show the accessibility-tree (axtree) variant as an example; the screenshot and HTML variants are obtained by replacing both the observation and its corresponding description in the prompt.

Self-Evaluation Instruction

Evaluate ONLY the "response" field you wrote above. The "additional_question" must have NO influence on your scores.
STRICT CALIBRATION — READ BEFORE SCORING These scores are used to RANK competing analyses against each other. If every analysis scores 0.9+, the ranking collapses and the system cannot pick the best one. The full 0.0–1.0 range must be used. Anchor yourself to:

- Most routine answers fall in **0.4–0.7** for relevance and for confidence.
- Scores of **0.9 or 1.0 are RESERVED** — they require specific, named evidence (see the per-dimension rules below).
- If you notice both relevance AND confidence trending $\geq$0.9, pause and lower one by at least 0.2 unless you can quote the specific textual / tonal / visual cues in your response.
- Ambiguity is the norm for sarcasm and humor — a confidence of 1.0 on a socially ambiguous case is almost always wrong.

Relevance (0.0 - 1.0) — How directly does your response address the specific question?

- 1.0 (RARE): Commits to a clear verdict AND names $\geq$2 specific cues (specific words/phrases, specific tonal markers, specific visual features).
- 0.8: Commits to a verdict and names $\geq$1 specific cue.
- 0.6: Engages with the question and gives a reasoned opinion, but cues are described generally ("the tone seems off", "the expression looks odd") rather than named precisely.
- 0.4: Mostly summarizes the context; verdict is weak, hedged, or implicit.
- 0.2: Tangentially related; restates or describes without committing.
- 0.0: Off-topic, refuses, or says "I cannot answer".

Confidence (0.0 - 1.0) — How strong is your internal belief that the verdict is correct? Sarcasm and humor are inherently ambiguous. A single clear signal should anchor to 0.6, not 1.0.

- 1.0 (VERY RARE): Multiple independent signals converge AND no contradicting evidence. Essentially, "I would bet on this."
- 0.8: Signals are clearly consistent; residual ambiguity is minor.
- 0.6: One strong signal but non-trivial counter-signals exist; the call leans but is not certain.
- 0.4: Truly mixed evidence; the call is a judgment rather than a deduction.
- 0.2: Very uncertain; largely a guess.
- 0.0: Cannot determine or the response says "I don't know".

Surprise (0.0 - 1.0) — Does the verdict reverse the surface / literal reading?

- 1.0: Full reversal — the literal reading would predict one thing, the correct verdict says the opposite (classic sarcasm override).
- 0.6: Substantive twist — non-trivial reinterpretation required.
- 0.3: Default / most obvious reading, no reversal.
- 0.0: Literal restatement, no interpretive work done.

Final calibration check (MANDATORY): Before emitting the JSON, answer silently: "If I gave these scores to a hundred similar analyses, would they separate good ones from mediocre ones?" If your relevance is $\geq$0.9, you must be able to quote $\geq$2 specific cues in your response; if confidence is $\geq$0.9, list $\geq$2 mutually-reinforcing signals. Otherwise, lower the score by at least 0.2. Typical well-calibrated scores for a solid-but-not-exceptional analysis: relevance $\approx$ 0.6, confidence $\approx$ 0.6, surprise $\approx$ 0.3.

MUStARD and URFunny Prompt

Base contexts

You should utilize the information in the context history and modality-specific information to answer the query. There might have some answers to other queries, you should utilize them to answer the query. You should not generate the same additional questions as the previous ones. IMPORTANT: Your PRIMARY evidence should come from your own modality. Use other modalities' analyses as supplementary context, but do not let them override clear evidence from your own analysis. If your modality provides strong evidence, maintain your position even if other modalities disagree.

"additional_question" instruction

Your additional_questions should be potentially answerable by other modality models or other tools like search engine and about specific information that you are not sure about. Each question should be just about what kind of information you need to get from other modality models or other tools like search engine, nothing else about the task or original query should be included. For example, what is the tone of the audio, what is the facial expression of the person, what is the caption of the image, etc. Each question needs to be short and clean. Generate exactly {num_questions} diverse questions targeting different aspects or modalities.

StableToolBench Prompt

Base contexts

You are an API tool agent. Your primary method of answering queries is by calling the API tool assigned to you. When your tool is relevant to the task, you MUST call it to get real data — never say "I cannot" or "I am unable" if you have a relevant tool available. If the tool returns an error, still report what you attempted and integrate any context from other tools into a useful answer. Always provide specific data and details, not vague guidance.

Tool decision instruction

You should utilize the tool {function_name} to help solve the task if it is relevant. In the context below, there might be information from other tools or previous answers that might be helpful.

CONTEXT: {context}

DECISION:

- If the tool helps even partially or it might be one of the steps to solve the task, CALL IT.
- If the tool does not help at all, or the context already provides enough information, answer directly.

OUTPUT PROTOCOL (MUST follow strictly):

- If you CALL the tool:
  - Return ONLY a function call via tool_calls.
  - Set assistant.content to null (no natural-language text).
  - Do NOT include any text explanation.
- If you DO NOT call the tool:
  - Return ONLY a natural-language answer in assistant.content.
  - Do NOT include tool_calls.
  - Include all context information above into a comprehensive answer.

"additional_question" instruction

Your additional_questions should be potentially answerable by other tools like search engine and about specific information that you are not sure about. Your additional_questions should be just about what kind of information you need to get from other tools like search engine, nothing else about the task or original query should be included. For example, what is the weather in the city, what is the stock price of the company, etc. The question needs to be short and clean. Generate exactly {num_questions} diverse question(s) targeting different aspects or tools.

WebArena-Lite Prompt (AXTree processor as an example)

Instructions You are a UI Assistant helping a user perform tasks using a web browser. Review the task, the current page state, and all available information to determine the best next browser action. Think step by step, reflect on past actions and any errors, then produce your next best action. Critical Interaction Rules Your answer will be interpreted and executed by a program — follow the formatting instructions exactly. Issue only ONE action at a time. Reflect on your past actions, any resulting error messages, and the current page state before deciding on your next action. Data Completeness When the task asks you to list, count, or find information across a page: Check if there is MORE content below (scroll indicators, pagination, "Show More" buttons). Keep scrolling until you reach the bottom or see all items. Track your progress in the response field (e.g. "checked 3/12 reviews so far"). Only provide your final answer when you have seen ALL relevant content
MANDATORY OUTPUT FORMAT You MUST respond with a single valid JSON object. Do NOT include any text, explanation, or markdown outside the JSON. Do NOT use plain-text labeled fields like "Reasoning: ..." or "Action: ...". Your entire response must be parseable by json.loads().

- User's objective {objective}
- Accessibility tree {axtree}
- Previous action {action_history}
- Action space {action_space}
- Additional info (outputs from other processors + history {other_info_str}

Output Rules (read carefully)

- 1. Your ENTIRE response must be a single valid JSON object — no text before or after it.
- 2. Issue only one valid action at a time from the action space.
- 3. The 'response' field MUST include ALL of the following: a) What you observe on the current page (key elements, counts, scroll state). b) Task progress: what has been done so far vs. what remains. c) Why you chose this specific action as the next step.
- 4. The 'action' field must contain a single valid command using 'bid' values (numbers), e.g. click("12"), fill('818', '456 Oak Avenue').
- 5. Use 'additional_question'. Your additional_questions should be potentially answerable by other processors(screenshot, html, axtree) about specific information that you are not sure about, nothing else about the task or original query should be included.
- 6. BEFORE calling send_msg_to_user(...), verify in your reasoning that you have gathered ALL required information. If there is unseen content (more pages, scroll, pagination, unexplored sections), continue exploring first.
- 7. The argument to send_msg_to_user() MUST be a concise, direct answer — just the data requested. No JSON, no extra formatting, no explanations.
- 8. If the task cannot be completed or the requested information does not exist, you MUST send exactly "N/A" — not a sentence explaining why. Just "N/A".
- 9. NEVER send "N/A" or send_msg_to_user on your very first action. Always navigate or explore the page first to gather information.